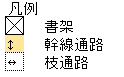
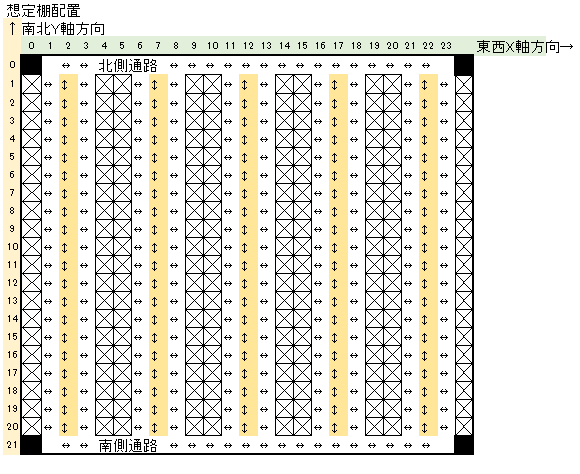
テーマ：通路制約を考慮した巡回セールスマン問題を遺伝的アルゴリズムで解いてみる

@matchbou

以下のような書架と通路の配置に於いて、ランダムに選択した書架座標を巡る経路の距離を最小化する。

手法は遺伝的アルゴリズムで行う。



仕様

幹線通路：Y軸方向は通路中央（上図の2,7,12,17,22列）を移動する。

枝通路：各書架には、幹線通路から左右（東西X軸方向）に一歩踏み出してアクセスする。

幹線通路間移動：書架をまたぐ幹線通路間の移動は、南北通路経由の近いほうを辿り行う。

（北回りor南回り）

実装方針

最短距離計算上は、各書架至近の幹線通路座標間の移動と捉える。

幹線通路・枝通路間の出入りは各々距離1で固定。

実装内容

「選択した書架座標データ」とそれを「至近幹線通路位置に変換するデータ」を入力データとして、

各書架座標データに対応する「至近幹線通路位置」を巡る巡回セールスマン問題として解く。

「至近幹線通路位置」間の「距離行列」はユークリッド距離等ではなく、上記表の通路制約を踏まえて算出したものを使用する。

Rコード

#working directory 設定-- 環境に合わせる

setwd("D:/\_User/\_SqlLocalDb")

#ライブラリ読込

library(tidyverse)

library(textshape)

library(GA)

#Y軸最大座標--棚の長さ次第で調整する。

maxY = 21

#立ち寄り書架座標データの読み込み---必要に応じ変える

df0 <- read\_csv("Data20180809EastSamp05.csv") %>%

dplyr::rename("x0"="x","y0"="y")

# 枝通路座標を幹線通路座標に変換するテーブルの作成

df1 = read\_csv("Data20180813West.csv")

df1a <- df1

df1a['x0'] <- df1a['x'] - 1

df1a['y0'] <- df1a['y']

df1b <- df1

df1b['x0'] <- df1b['x'] + 1

df1b['y0'] <- df1b['y']

df2 <- rbind(df1a, df1b)

#幹線通路座標に変えて重複を削除（相向かい合う棚は同じ幹線通路座標を共有する為、重複もあり得るのでユニーク化する）

df3 = dplyr::inner\_join(df0, df2, by=c("x0","y0"))

df4 <- df3 %>% dplyr::select("x","y") %>% distinct(x,y, .keep\_all = FALSE)

#X軸距離の算出

locX1Same <- df4 %>% dplyr::rename("x0"="x","y0"="y")

locX1Same2 <- locX1Same %>% rename("x1"="x0","y1"="y0")

locX1SameCross <- tbl\_df( merge(locX1Same,locX1Same2,all=TRUE) )

locX1SameCross['distXall'] <- sqrt((locX1SameCross['x0'] - locX1SameCross['x1'])^2)

#Y軸距離の算出-その1同一列

locY1Same <- df4 %>% dplyr::rename("x0"="x","y0"="y")

locY1Same2 <- locY1Same %>% rename("x1"="x0","y1"="y0")

locY1SameCross <- tbl\_df( merge(locY1Same,locY1Same2,all=TRUE) )

locY1SameCross['distYsame'] <- sqrt((locY1SameCross['y0'] - locY1SameCross['y1'])^2)

#Y軸距離の算出-その2相違列北回り

locY1Kita <- df4 %>% dplyr::rename("x0"="x","y0"="y")

locY1Kita2 <- locY1Kita %>% rename("x1"="x0","y1"="y0")

locY1KitaCross <- tbl\_df( merge(locY1Kita,locY1Kita2,all=TRUE) )

locY1KitaCross['distYkita'] <- locY1KitaCross['y0'] + locY1KitaCross['y1']

#Y軸距離の算出-その2相違列南回り

locY1Minami <- df4 %>% dplyr::rename("x0"="x","y0"="y")

locY1Minami['y0'] <- maxY - locY1Minami['y0']

locY1Minami2 <- locY1Minami %>% rename("x1"="x0","y1"="y0")

locY1MinamiCross <- tbl\_df( merge(locY1Minami,locY1Minami2,all=TRUE) )

locY1MinamiCross['distYminami'] <- locY1MinamiCross['y0'] + locY1MinamiCross['y1']

locY1MinamiCross['y0'] <- maxY - locY1MinamiCross['y0']

locY1MinamiCross['y1'] <- maxY - locY1MinamiCross['y1']

#距離中間テーブルの結合&派生変数処理

locXYcross01 <- dplyr::inner\_join( locX1SameCross, locY1SameCross, by=c("x0","y0","x1","y1") ) %>%

dplyr::inner\_join( locY1KitaCross, by=c("x0","y0","x1","y1") ) %>%

dplyr::inner\_join( locY1MinamiCross, by=c("x0","y0","x1","y1") ) %>%

dplyr::mutate(distYdiff = pmin(distYkita, distYminami)) %>%

dplyr::mutate(distYall = ifelse( x0 == x1, distYsame, distYdiff ) ) %>%

dplyr::mutate(distXYall = distXall+distYall ) %>%

dplyr::mutate(x0y0 = paste("x",x0,"y",y0)) %>%

dplyr::mutate(x1y1 = paste("x",x1,"y",y1))

locXY <- locXYcross01 %>% dplyr::select(x0y0,x1y1,distXYall)

distXY0 <- locXY %>% dplyr::arrange(x0y0,x1y1) %>%

tidyr::spread( key=x1y1, value=distXYall, fill=NA, convert=FALSE, drop=TRUE)

##距離行列完成!!

matdistXY <- as.matrix(column\_to\_rownames(distXY0,'x0y0') )

##距離行列をdistオブジェクトでも用意しておく

distobjXY <- as.dist(matdistXY)

####遺伝的アルゴリズム実行

##以下の部分の参考URL（ダウンロードPDF）

# https://www.jstatsoft.org/index.php/jss/article/view/v053i04/v53i04.pdf

#Function to calculate tour length

tourLength <- function(tour, distMatrix) {

tour <- c(tour, tour[1])

route <- embed(tour, 2)[,2:1]

sum(distMatrix[route])

}

#Firness function to be maximized --- Inverse of tour length ( Shorter is better!!! )

tspFitness <- function(tour, ...) 1/tourLength(tour, ...)

#Run GA

GA <- ga(type = "permutation", fitness = tspFitness, distMatrix = matdistXY,

min = 1, max = ( ncol(matdistXY) + nrow(matdistXY) )/2, popSize = 50, maxiter = 1000, #crossover = 0.6,

run = 5000, pmutation = 0.001)

#Show Summary

summary(GA)

#Visualization

mds <- cmdscale(distobjXY)

x <- mds[, 1]

y <- -mds[, 2]

plot(x, y, type = "n", asp = 1, xlab = "", ylab = "")

abline(h = pretty(range(x), 10), v = pretty(range(y), 10),

col = "light gray")

tour <- GA@solution[1, ]

tour <- c(tour, tour[1])

n <- length(tour)

arrows(x[tour[-n]], y[tour[-n]], x[tour[-1]], y[tour[-1]],

length = 0.15, angle = 25, col = "steelblue", lwd = 2)

text(x, y, labels(distobjXY), cex=0.8)

入力データ

「立ち寄り書架座標データ」プログラム中のData20180809EastSamp05.csv

「想定棚配置」図の枝線通路上の任意の座標を選択しリスト化。

|  |  |
| --- | --- |
| x | y |
| 13 | 12 |
| 1 | 7 |
| 21 | 1 |
| 11 | 20 |
| 13 | 6 |
| 18 | 5 |

「至近幹線通路位置に変換するデータ」Data20180813West.csv

以下のような100行2列の幹線通路座標位置データに対しx座標を±1した枝通路データを作成し、縦にスタックする事で幹線通路の左右にある枝通路座標と幹線通路座標の対応（変換）テーブルを作成する。

元の表（100行2列）

|  |  |
| --- | --- |
| x | y |
| 2 | 1 |
| 2 | 2 |
| 2 | 3 |
| 2 | 4 |
| 2 | 5 |
| 2 | 6 |
| 中略 | 中略 |
| 22 | 14 |
| 22 | 15 |
| 22 | 16 |
| 22 | 17 |
| 22 | 18 |
| 22 | 19 |
| 22 | 20 |

手順1.x0 = x -1, y0 = yとなる列を付与したデータを作成（100行4列）

|  |  |  |  |
| --- | --- | --- | --- |
| x | y | x0 | y0 |
| 2 | 1 | 1 | 1 |
| 2 | 2 | 1 | 2 |
| 2 | 3 | 1 | 3 |
| 2 | 4 | 1 | 4 |
| 2 | 5 | 1 | 5 |
| 2 | 6 | 1 | 6 |
| 中略 | 中略 | 中略 | 中略 |
| 22 | 14 | 21 | 14 |
| 22 | 15 | 21 | 15 |
| 22 | 16 | 21 | 16 |
| 22 | 17 | 21 | 17 |
| 22 | 18 | 21 | 18 |
| 22 | 19 | 21 | 19 |
| 22 | 20 | 21 | 20 |

手順2 .x0 = x +1, y0 = yとなる列を付与したデータを作成（100行4列）

|  |  |  |  |
| --- | --- | --- | --- |
| x | y | x0 | y0 |
| 2 | 1 | 3 | 1 |
| 2 | 2 | 3 | 2 |
| 2 | 3 | 3 | 3 |
| 2 | 4 | 3 | 4 |
| 2 | 5 | 3 | 5 |
| 2 | 6 | 3 | 6 |
| 中略 | 中略 | 中略 | 中略 |
| 22 | 14 | 23 | 14 |
| 22 | 15 | 23 | 15 |
| 22 | 16 | 23 | 16 |
| 22 | 17 | 23 | 17 |
| 22 | 18 | 23 | 18 |
| 22 | 19 | 23 | 19 |
| 22 | 20 | 23 | 20 |

手順3 手順1と手順2で作成したデータを縦積みしたデータを作成（200行4列）

|  |  |  |  |
| --- | --- | --- | --- |
| x | y | x0 | y0 |
| 2 | 1 | 1 | 1 |
| 2 | 2 | 1 | 2 |
| 2 | 3 | 1 | 3 |
| 2 | 4 | 1 | 4 |
| 2 | 5 | 1 | 5 |
| 2 | 6 | 1 | 6 |
| 中略 | 中略 | 中略 | 中略 |
| 22 | 14 | 23 | 14 |
| 22 | 15 | 23 | 15 |
| 22 | 16 | 23 | 16 |
| 22 | 17 | 23 | 17 |
| 22 | 18 | 23 | 18 |
| 22 | 19 | 23 | 19 |
| 22 | 20 | 23 | 20 |

この表を用いて「立ち寄り書架座標データ」の書架前の座標を対応する幹線通路上の座標に変換して巡回セールスマン問題を解く、

実行結果

+-----------------------------------+

| Genetic Algorithm |

+-----------------------------------+

GA settings:

Type = permutation

Population size = 50

Number of generations = 1000

Elitism = 2

Crossover probability = 0.8

Mutation probability = 0.001

GA results:

Iterations = 1000

Fitness function value = 0.0106383

Solutions =

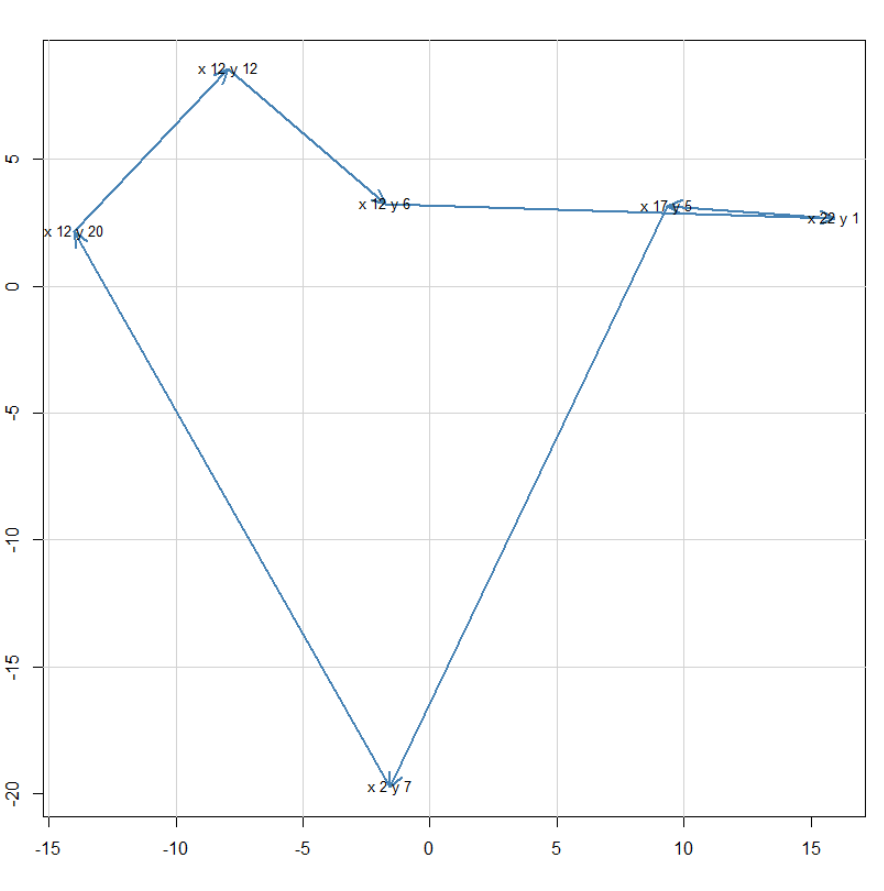
x1 x2 x3 x4 x5 x6

[1,] 3 6 4 5 2 1

[2,] 3 4 6 5 2 1

上記「Solutions」[1,]の3,6,4,5,2,1順に「立寄り書架座標データ」の行を巡るのが今回の試行で算出した最良解。

「Solutions」[2,]も同等解。図では経路が直線描画されているが、実際は冒頭の棚配置を考慮した距離で経路探索。



[考察]

パラメータの最適化は今後の課題だが、書架配置などを変更すると最適パラメータも変わるハズ。

maxiterの回数変えたりして何回か実行してみたが、始点・終点、方向は変わるが順番は基本崩れなかった。