Introducción al paquete metaheuR

El paquete "metaheuR" ha sido creado con fines didácticos por Borja Calvo, Usue Mori, Ander Carreño y Josu Ceberio (BCAM-UPV/EHU) .

Instalación

Descargar el archivo "metaheuR_0.3.tar.gz" desde e-gela (NO ABRIRLO !!!) en una carpeta "C:\\.....\\carpeta"

Instalarlo desde la consola de R, o desde un archivo script:

```
setwd("C:\\.....\\carpeta")
install.packages("metaheuR\_0.3.tar.gz", repos=NULL, type="source")
```

NOTA: previamente hay que instalar la librería "ggplot2"

install.packages("ggplot2“)

- library(metaheuR)

Problemas implementados

- **GCP:** Graph Coloring Problem (problema de coloreado de grafos)
- **TSP:** Travel Salesman Problem (problema del agente viajero)
- **KSP:** Knap Sack Problem (problema de la mochila)
- **LOP:** Linear Ordering Problem
- **MIS:** Maximal Independent Set
- **MDS:** Minimum Dominating Set
- **QAP:** Quadratic assignment problem
- **rosenbrockProblem**: Rosenbrock function to evaluate numeric algorithms

Problema del agente viajero (Travel Salesman Problem)

tspProblem (matriz, coordenadas)

Entrada:

- matriz nxn con los pesos de las aristas
- coordenadas de los vértices, como una matriz de n filas y 2 columnas.

Estos últimos solo son necesarios en el caso de que se quiera la representación gráfica.

tspProblem

tspProblem (matriz, coordenadas) -> lista

El resultado es una lista con tres campos. Para saber cuales son los campos, teclear.

names(lista)

En el caso del TSP, los campos son

- lista$evaluate: función que evalúa la función objetivo a partir de una solución.:
- lista$size: número de vértices
- lista$plotSolution: función que dibuja la solución.

Tened en cuenta que en R las funciones son objetos, y por tanto pueden ser elementos de una lista.

Soluciones

Las soluciones del TSP son permutaciones del número de vértices:

```
sol <- randomPermutation(lista$size)
```

Esta solución se puede evaluar o dibujar:

```
lista$evaluate( sol)
lista$plotSolution(sol)
```

Problema de coloreado de grafos (Graph Coloring Problem)

La función graphColoringProblem genera el problema a partir de un grafo.

Previamente generamos el grafo con la función random.graph.game de la librería igraph

```
grafo <- random.graph.game(n, p)
```

Si 0<p<1 es la proporción de vertices adyacentes; si es un entero es el grado d elso vértices

```
colores <- graphColoringProblem(grafo)
```

graphColoringProblem

El resultado es una lista

- lista$evaluate(sol) : nro. de colores en el grafo
- lista$valid(sol) : FALSO si hay vértices adyacentes con el mismo color
- lista$correct(sol) : corrige la solución, si no es válida.
- lista$plot(sol) : dibuja la solución

Soluciones

Definimos tantos colores como vértices en el grafo.

```
colores <- paste("c", 1:n, sep="")
```

A partir del número, k, de colores, se asignan aleatoriamente k colores a los vértices del grafo.

```
solucion <- factor(sample(colores[1:k], size=n,
    replace=TRUE), levels=colores)
```

"sample" toma combinaciones con repetición de k colores tomados de n en n

"factor" genera a partir de este vector la estructura tipo factor de R, necesaria para las funciones de la librería metaheuR

Problema de la mochila (KSP)

knapsackProblem(peso , valor , total)

El resultado es una lista:

lista$evaluate(solucion)

lista$valid(solucion)

lista$correct(solucion)

Soluciones

Las soluciones son vectores binarios TRUE/FALSE.

Se generan aleatoriamente como:

```
sample (c (TRUE,FALSE) , n, replace = TRUE)
```