Tareas Semana II

Jose A. Lozano, Josu Ceberio

15 de octubre de 2019

El objetivo de esta práctica es familiarizarse con los algoritmos de búsqueda local. Para ello habrá que implementar una búsqueda local básica para cada uno de los problemas trabajados en la Práctica 1. Se utilizará un sistema de vecinos dado y, como base, las funciones de evaluación implementadas en la tarea de la Práctica 1.

Los algoritmos de optimización podrán ser implementados en R o Python. Para realizar la revisión de los algoritmos implementados de manera automática, los programas, y sus parámetros de entrada y salida tendrán nombres predeterminados. A pesar de que los ejercicios de consolidación están hechos en Python, tened en cuenta que las prácticas, es decir, los programas se pueden entregar en R.

0.1. PROBLEMA DE ASIGNACION CUADRÁTICA

El programa se llamará: QAPLocalSearch. El sistema de vecinos está definido de la siguiente manera: dada una solución, son soluciones vecinas todas aquellas que se generan intercambiando de lugar dos posiciones de la permutación. Por ejemplo, la vecindad de $x = [1, 2, 3, 4]$ es
$\{[2, 1, 3, 4], [3, 2, 1, 4], [4, 2, 3, 1], [1, 3, 2, 4], [1, 4, 3, 2], [1, 2, 4, 3]\}$

El algoritmo de búsqueda local debe recorrer todo el vecindario, eligiendo a cada paso la mejor solución vecina, para continuar con el vecindario de la solución actual terminando en un óptimo local.

En este caso recordad que el problema es de minimización.

Parámetros de entrada:

- Nombre del fichero con la instancia
- Solución inicial (Permutación)

Parámetros de salida:

- Solución final
- Valor de de la solución final

El formato de los ficheros conteniendo las instancias será similar al de la Práctica 1 (no os olvidéis que las instancias que se encuentran dentro de los ejercicios de consolidación están en un formato diferente).

EJEMPLO DE LLAMADA AL OPTIMIZADOR EN PYTHON:
$best\_sol, best\_val = QAPLocalSearch('../Instances/QAP/QAP.qap.n10{,}1', [1, 2, 3, 4, 5, 6, 7, 8, 9, 10])$

En el caso de R el programa se llamará igual y tendrá los mismos parámetros de entrada y salida

0.2. BIPARTICIÓN DEL GRAFO

El programa se llamará: BipLocalSearch. El sistema de vecinos está definido de la siguiente manera. Dada una solución, son soluciones vecinas todas aquellas que se generan intercambiando dos posiciones tales que una tenga valor 1 y la otra 0. Por ejemplo, la vecindad de $x = [1, 1, 0, 0]$ es

$\{[0, 1, 1, 0], [0, 1, 0, 1], [1, 0, 1, 0], [1, 0, 0, 1]\}$

El algoritmo de búsqueda local debe recorrer todo el vecindario, eligiendo a cada paso la mejor solución vecina.

Recordad que el algoritmo de búsqueda local no termina en un paso, sino que el criterio de parada es alcanzar un óptimo local. Además no os olvidéis de que en este caso el problema que se quiere resolver es uno de maximización.

Parámetros de entrada:

- Nombre del fichero con el grafo
- Solución inicial (vector con un número balanceado de unos y ceros)

Parámetros de salida:

- Solución final
- Valor de la solución final

El formato de los ficheros conteniendo las instancias será similar al de la Práctica 1.

EJEMPLO DE LLAMADA AL OPTIMIZADOR EN PYTHON:
$best\_sol, best\_val = BipLocalSearch('../Instances/BIPART/Cebe.bip.n10{,}1', [1, 1, 1, 1, 1, 0, 0, 0, 0, 0])$
EJEMPLO HIPOTÉTICO DE SALIDA DEL PROGRAMA:
$best\_sol = [0, 1, 1, 1, 1, 0, 0, 0, 0, 1]$
$best\_val = 34$

En el caso de R el programa se llamará igual y tendrá los mismos parámetros de entrada y salida