Tareas Semana IV

Jose A. Lozano, Roberto Santana

20 de marzo de 2018

La práctica final tiene como objetivo que el alumno pueda ejercitar los conocimientos aprendidos durante el curso. Para ello habrá que implementar un algoritmo de optimización local avanzado entre los mostrados durante el curso (GRASP, búsqueda de vecindad variable, etc..) y un algoritmo evolutivo para cada uno de los problemas trabajados en la Práctica I. Es decir en total serán 4 programas. Dos para cada problema.

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. Por favor, seguid las normas en cuanto a los nombres de forma estricta o sino no realizaremos la evaluación de vuestros programas.

Como novedad, en esta última tarea debéis escribir una documentación que describa los algoritmos implementados y los resultados experimentales obtenidos. Existe un fichero dentro del material de esta semana que describe como debe ser esta documentación. Por favor, leed en detalle dicho documento, ya que la documentación es la mitad de la nota de esta práctica.

1. BIPARTICIÓN DEL GRAFO

1.1. Búsqueda local avanzada

El programa se llamará:: BipAdvLocalSearch. 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 será algunos de los estudiados durante el curso (GRASP, VNS, enfriamiento estadístico, etc.) y diferente al implementado en la Tarea 2. Es decir, deberá contener un mecanismo para escapar de óptimos locales.

Parámetros de entrada:

- Nombre del fichero con el grafo
- Solución inicial (vector con un número balanceado de unos y ceros)
- Número máximo de evaluaciones permitidas al algoritmo (max_eval)
- Parámetro k que identifica cada cuántas evaluaciones es necesario calcular el valor del mejor individuo hasta el momento (Ver explicación parámetros de salida)

Parámetros de salida:

- Solución final

- Vector con el valor de la mejor solución encontrada cada k evaluaciones. Es un vector de tamaño max_eval/k, donde max_eval es un múltiplo de k. La componente i del vector contiene cuál ha sido el mejor valor encontrado por la búsqueda tras i*(max_eval/k) evaluaciones.

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

EJEMPLO DE LLAMADA AL OPTIMIZADOR EN PYTHON:
$best\_sol, best\_vals = BipAdvLocalSearch('../Instances/BIPART/Cebe.bip.n10{,}1', [1,1,1,1,1,0,0,0,0,0], 100, 10)$
EJEMPLO HIPOTÉTICO DE SALIDA DEL PROGRAMA:
$best\_sol = [0,1,1,1,1,0,0,0,0,1]$
$best\_val = [90, 100, 130, 195, 218, 218, 218, 320, 320, 413, 523, 523]$

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

1.2. Algoritmo evolutivo

El programa que implementa el algoritmo evolutivo se llamará: BipAdvEA.
Los operadores del algoritmo evolutivo (i.e, método de selección, operadores de cruzamiento y mutación, etc) serán de libre elección por parte de los estudiantes.
Parámetros de entrada:

- Nombre del fichero con el grafo.
- Tamaño de población.
- Número de generaciones.

Parámetros de salida:

- Vector donde cada posición k es el mejor valor de la función obtenido por el algoritmo después de k generaciones.
- Vector con la mejor solución encontrada durante toda la búsqueda.

La solución devuelta por el algoritmo tiene que cumplir con las restricciones del problema, i.e., mismo números de ceros y uno.
El formato de los ficheros conteniendo las instancias será similar al de la Práctica 1.

EJEMPLO DE LLAMADA AL OPTIMIZADOR EN PYTHON:
$best\_sol, evals = BipAdvEA('../Instances/BIPART/Cebe.bip.n10{,}1', 100, 12)$
Este programa ejecutará el algoritmo evolutivo con una población de tamaño 100 por 12 generaciones
EJEMPLO HIPOTÉTICO DE SALIDA DEL PROGRAMA:
$best\_sol = [0,1,1,1,1,0,0,0,0,1]$
$evals = [90, 100, 130, 195, 218, 218, 218, 320, 320, 413, 523, 523]$

Nótese como el vector evals tiene valores no decrecientes. Es decir, al aumentar el número de evaluaciones, el mejor valor que ha sido encontrado (teniendo en cuenta que maximizamos) puede mantenerse igual pero no empeorar.

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

2. PROBLEMA DE ASIGNACION CUADRÁTICA

2.1. Búsqueda local avanzada

El programa se llamará¡: QAPAdvLocalSearch. 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 será algunos de los estudiados durante el curso (GRASP, VNS, enfriamiento estadístico, etc.) y diferente al implementado en la Tarea 2. Es decir, deberá contener un mecanismo para escapar de óptimos locales.

Parámetros de entrada:

- Nombre del fichero con las matrices
- Solución inicial (Permutación)
- Número máximo de evaluaciones permitidas al algoritmo (max_eval)
- Parámetro k que identifica cada cuántas evaluaciones es necesario calcular el valor del mejor individuo hasta el momento (Ver explicación parámetros de salida)

Parámetros de salida:

- Solución final
- Vector con el valor de la mejor solución encontrada cada k evaluaciones. Es un vector de tamaño max_eval/k, donde max_eval es un múltiplo de k. La componente i del vector contiene cuál ha sido el mejor valor encontrado por la búsqueda tras i*(max_eval/k) evaluaciones.

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

EJEMPLO DE LLAMADA AL OPTIMIZADOR EN PYTHON:
$best\_sol, best\_vals = QAPAdvLocalSearch('../Instances/QAP/QAP.qap.n10,1', [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 100, 10)$

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

2.2. Algoritmo evolutivo

El programa se llamará: QAPAdvEA.

Los operadores del algoritmo evolutivo (i.e, método de selección, operadores de cruzamiento y mutación, etc) serán de libre elección por parte de los estudiantes.

Parámetros de entrada:

- Nombre del fichero con las matrices.
- Tamaño de población.
- Número de generaciones.

Parámetros de salida:

- Vector donde cada posición k es el mejor valor de la función obtenido por el algoritmo después de k generaciones.

- Vector con la mejor solución encontrada durante toda la búsqueda.

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

EJEMPLO DE LLAMADA AL OPTIMIZADOR EN PYTHON:
$best\_sol, evals = QAPAdvEA('../Instances/QAP/QAP.qap.n10{,}1', 500, 12)$
La solución devuelta por el algoritmo tiene que cumplir las restricciones del problema, i.e., la solución es una permutación.

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