Tareas Semana III

Jose A. Lozano, Josu Ceberio

22 de octubre de 2019

El objetivo de esta práctica es familiarizarse con los algoritmos de optimización basados en poblaciones. Para ello habrá que implementar un algoritmo basado en poblaciones para cada uno de los problemas trabajados en la Práctica I.

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.

1. PROBLEMA DE ASIGNACION CUADRÁTICA

El programa se llamará: QAPEA. Los operadores del algoritmo evolutivo (i.e, método de selección, operadores de cruce 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.

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

EJEMPLO DE LLAMADA AL OPTIMIZADOR EN PYTHON:

$$best\_sol, evals = QAPEA('../Instances/QAP/QAP.qap.n10{,}1', 500, 120)$$

Este programa ejecutaría el algoritmo poblacional con una población de tamaño 500 por 120 generaciones. La salida sería, por un lado un vector de tamaño 120 (conteniendo en el lugar k-ésimo el mejor valor objetivo encontrado hasta dicha generación k), y por otro la permutación con el mejor valor de función objetivo encontrado durante la búsqueda completa.

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.

2. BIPARTICIÓN DEL GRAFO

El programa se llamará: BipEA.

Los operadores del algoritmo evolutivo (i.e, método de selección, operadores de cruce 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:
$best\_sol, evals = BipEA('../Instances/BIPART/Cebe.bip.n10{,}1', 100, 12)$
Este programa ejecutaría el algoritmo poblacional con una población de tamaño 100 por 12 generaciones.
EJEMPLO HIPOTÉTICO DE SALIDA DEL PROGRAMA:
$best\_sol = [0, 1, 1, 1, 1, 0, 0, 0, 0, 1]$
$evals = [90, 100, 130, 130, 218, 218, 218, 320, 320, 413, 523, 523]$

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.

3. A TENER EN CUENTA

- Para la definición de los algoritmos evolutivos se pueden utilizar las librerías disponibles en Python y R que implementan algoritmos evolutivos.
- Cualquiera sea la aproximación utilizada la solución final devuelta por el algoritmo debe ser factible.