BÚSQUEDA LOCAL Y ALGORITMOS EVOLUTIVOS

SISTEMAS INTELIGENTES

EJERCICIO 1

Considerando el problema 3-SAT para la expresión lógica contenida en el archivo 'uf20.py', definida con 20 variables y 91 cláusulas que se sabe que es satisfacible, se intenta resolverlo usando Hill Climbing. Se pide que:

1. Proponga una representación para las posibles soluciones y determine el tamaño del espacio de búsqueda asociado a esa representación.
2. Proponga una función de evaluación.
3. Defina la manera en que se obtienen las soluciones vecinas a la solución actual.

Intente resolverlo ahora empleando un Algoritmo Evolutivo. Se pide que:

4. Proponga una representación para las posibles soluciones y una función de evaluación. Describa las diferencias, si las hay, con las propuestas anteriormente.
5. Defina que operadores van a usarse durante el proceso evolutivo.

Implemente y pruebe lo definido en los puntos anteriores. Evalúe los resultados obtenidos. Indique los valores de los parámetros usados durante la ejecución de ambos algoritmos.

EJERCICIO 2

Intente resolver un problema de optimización numérica usando Simulated Annealing y un Algoritmo Evolutivo. La función a minimizar es la llamada Rastrigin y su definición es:

$$R(\bar{x}) = 10N \sum_{i=1}^{N} x_i^2 - 10\cos(2\pi x_i), \bar{x} = (x_1, x_2, \ldots, x_N)$$

a) Proponga una representación adecuada para las soluciones al problema cuando N=10, sabiendo que el rango de las variables va de -5 a 5 y se requiere trabajar al menos con 2 dígitos decimales de precisión.

b) Proponga una función de evaluación que le permita calificar que tan buena es una solución.

c) Para Simulated Annealing defina como y cuantas soluciones vecinas se van a generar en cada paso de la búsqueda mientras que para el Algoritmo Evolutivo defina que operadores van a usarse durante el proceso evolutivo.

d) Implemente y pruebe lo definido en los puntos anteriores. Evalúe los resultados obtenidos.

EJERCICIO 3

Intente resolver el problema de las N-Reinas usando Hill Climbing y un Algoritmo Evolutivo.

a) Proponga una representación adecuada para las soluciones.

b) Proponga una función de evaluación que le permita calificar que tan buena es una solución.

c) Defina de que manera se obtendrán las soluciones vecinas para Hill Climbing y que operadores van a usarse con el Algoritmo Evolutivo.

d) Implemente y pruebe lo definido en los puntos anteriores para el caso donde N=16. Evalúe los resultados obtenidos. Detalle todos los parámetros usados. Justifique.

EJERCICIO 4

Intente resolver dos casos diferentes del problema TSP usando Simulated Annealing y un Algoritmo Evolutivo. El primer problema es de 29 ciudades y el segundo, de 101 ciudades.

Las ubicaciones de las ciudades se encuentran dentro de una matriz contenida en el archivo con extensión 'cities.npy' mientras que los costos de ir de una ciudad a otra están contenidos en otra matriz almacenada en el archivo con extensión 'distances.npy'. Para ambos casos, se conocen las soluciones óptimas, las cuales se encuentran en los archivos con extensión 'opt.tour.npy'.

Se pide que:

a) Proponga una representación adecuada para las soluciones.
b) Proponga una función de evaluación que le permita calificar que tan buena es una solución.
c) Defina como se generarán las soluciones vecinas al usar Simulated Annealing y cuales serán los operadores empleados en el Algoritmo Evolutivo.
d) Implemente y pruebe lo definido en los puntos anteriores. Evalúe los resultados obtenidos. Detalle todos los parámetros usados durante la ejecución de ambos algoritmos. Justifique.

EJERCICIO 5

Implemente un Algoritmo Evolutivo que, dadas las siguientes funciones, busque cumplir con el objetivo propuesto para cada una de ellas.

a) Para $f_1(x, y) = 0.5x^2 + y^2 - xy - 2x - 6y$, con $-5 \leqslant x \leqslant 5,\ -10 \leqslant y \leqslant 10$, buscar donde se encuentra el mínimo de esta función.

b) Para $f_2(\mathbf{x}) = \sum_{i=1}^{n} |x_i|^{(i+1)}$, con $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ y $n = 4$, buscar $\mathbf{x}$ tal que $f_2(\mathbf{x}) = 0$, sabiendo que $-1 \leqslant x_1 \leqslant 1$, $-0{,}8 \leqslant x_2 \leqslant 1{,}2$, $-0{,}5 \leqslant x_3 \leqslant 0{,}75$ y $-1{,}4 \leqslant x_4 \leqslant 0{,}5$.

En cada uno de los puntos, detalle:

i. El tipo y forma del cromosoma.
ii. Los operadores genéticos usados.
iii. Los parámetros empleados para la ejecución del algoritmo.

Para asegurarse de haber obtenido resultados consistentes, realice 10 ejecuciones independientes e informe los mejores resultados encontrados en cada ejecución junto con su fitness y la cantidad de generaciones que se necesitaron para obtenerlos.

EJERCICIO 6

Una manera de conseguir una búsqueda más efectiva cuando se emplea un Algoritmo Evolutivo es dotarlo de operadores con más “inteligencia” que los operadores sencillos de cruza y mutación.

Resuelva nuevamente el ejercicio 5 incorporando ahora un nuevo operador de mutación a los operadores usados originalmente, que tome un cromosoma y lo someta a la ejecución de un algoritmo de Hill Climbing durante 25 iteraciones, devolviendo como resultado de la “mutación” el mejor valor encontrado.

Informe los nuevos resultados obtenidos y compárelos con los del ejercicio 5, realizando la misma cantidad de ejecuciones independientes.

EJERCICIO 7

Un ladrón intenta decidir que cosas llevarse tras haber conseguido entrar a una tienda de electrodomésticos. El problema es que debe elegir entre 10 electrodomésticos, todos de distinto precio y peso, pero solo trajo 2 bolsas para llevarlos, con 51 y 78 kilogramos de capacidad. Como son productos que están en exposición, solo hay uno de cada tipo y, por lo tanto, no podría llevarse dos o más productos iguales.

Describa un método que le ayude a este malhechor a elegir la combinación óptima de productos (o una que esté lo más cercano posible a la óptima), de forma de obtener la mayor ganancia y no excederse de la capacidad de sus bolsas.

Detalle y justifique todas las decisiones tomadas con respecto al método, parámetros empleados, representación, etc. Implemente el método propuesto y úselo para encontrar e informar la mejor solución que pueda obtener.

El precio (en pesos) y el peso (en kilogramos) de cada producto se encuentra disponible en el archivo '`ladron.py`'.

Para asegurarse de haber obtenido resultados consistentes, realice 10 ejecuciones independientes e informe el mejor resultado encontrado en cada ejecución junto con su evaluación y la cantidad de generaciones que se necesitaron para obtenerlo.