Metaheurísticas. Examen final. Julio de 2019

1. Explica cómo resolverías el problema de las cuatro reinas mediante una metaheurística basada en trayectorias. Debes poner énfasis en los siguientes aspectos: (a) Cómo serían la codificación de los estados (b) Cuáles serían los estados inicial y final (c) Cuáles serían las reglas de producción (d) Qué metaheurística utilizarías y por qué (e) breve descripción del funcionamiento

   Hay dos formas de representar el espacio de estados para resolver el problema de las cuatro reinas. Una consistiría en colocar de una vez las cuatro reinas al azar (estado inicial), y definir una serie de movimientos de las piezas a lo largo del tablero. Otra forma sería partir de un tablero vacío e ir añadiendo reinas en posiciones donde no sean atacadas. En cada uno de los casos habría que definir una heurística que condujera la búsqueda. Por ejemplo, una heurística podría ser el número de posiciones libres de amenaza que quedan después de colocar la reina en la nueva posición (o de añadir una nueva reina en la otra codificación).

   Una vez planteados el espacio de estados (indicando las ED que se utilizarían para codificarlo), los estados inicial y final, y las reglas para cambiar de un estado a otro, sería sencillo explicar una metaheurística para resolver el problema. Personalmente hubiera elegido una búsqueda tabú, por aquello de evitar repetir posiciones que nos llevan a ciclos dentro del grafo de estados. La explicación del algoritmo en cuestión la omito por razones obvias.

2. Basándote en lo expuesto anteriormente, ¿consideras que el problema de las cuatro reinas se podría resolver bien mediante el algoritmo ACS? ¿Cómo plantearías su resolución?

   Si el problema se representa como un recorrido a través de un grafo (como la segunda codificación que se ha presentado en el ejercicio 1 (o incluso la otra) es factible resolverlo con ACS. Una vez establecido este punto, no es complicado explicar cómo resolver el problema con ACS.

3. Imagina que queremos hallar el máximo la función $f(x) = x^2 - 4x + 4$ en el intervalo (0,4) con algoritmo genético con codificación binaria. Explica cómo llevarías a cabo dicha codificación y cuáles serían los parámetros de diseño más importantes a definir para encontrar la solución a dicho problema.

   Para resolver un problema de optimización numérica con un algoritmo genético binario, hay que codificar los valores numéricos en binario. Una forma sería usar el estándar IEEE para representar los números, con su mantisa y su exponente, pero NO ES OPERATIVO. Lo más sencillo, como se planteó en clase de problemas, es dividir el intervalo 0,4 en subintervalos de tamaño tal, que permitieran una precisión de 2 cifras decimales. Dicho de otro modo, $(4-0) / 2^n <= 0.01$, es decir, $400 <= 2^n$. Y despejando n se obtiene el número mínimo de bits para codificar las soluciones $n >= \log_2(400) = 8.64 \approx 9$ bits.

   De este modo, (0,0,0,0,0,0,0,0,0) representará el valor de x=0 y (1,1,1,1,1,1,1,1,1) representará el valor $4\frac{2^9-1}{2^9}$. Ya tenemos genotipo y fenotipo. Para calcular el *fitness* sólo hay que calcular el valor de f en el punto obtenido.

   ¿Qué faltaría? Explicar el algoritmo que va a usarse. Yo utilizaría un generacional convencional, como operadores cruce en un punto (pc=0.9) y mutación de un gen (pm=0.1 o menor), selección por torneo de tamaño 2 y poco más, tal vez indicar la dinámica del algoritmo.

4. Formula un problema que requiera de un algoritmo de nichos para su resolución. Una vez planteado, indica cómo lo resolverías y explica brevemente la dinámica de funcionamiento del algoritmo que has elegido para resolverlo.

   Cualquier problema en el que existan varias soluciones posibles (óptimos globales o locales) y todas de interés es un problema en el que hay que aplicar nichos. NO CONFUNDIR CON PROBLEMAS MULTI-OBJETIVO.

   Si a alguien no se le ocurre un problema, pues que dibuje una gráfica diciendo que la función de aptitud presenta dos o más óptimos. Aunque no lo valore como respuesta perfecta, al menos se está demostrando que

se sabe lo que es un problema multimodal y cómo se resuelve. Con respecto al algoritmo elegido, ya cada uno puede decidir entre nichos espaciales o temporales y explicar los fundamentos de cada uno.

5. Imagina que pretendes resolver el problema de la mochila con un algoritmo genético binario, pero por la complejidad del problema planteado, es fácil que el método quede atrapado en un óptimo local. ¿Qué estrategias se te ocurren para salir del mismo?

Cuando un algoritmo se queda atrapado en un óptimo local, es porque la población ha perdido diversidad. La única forma de solucionar este problema es aumentar la diversidad de la población. Clásicamente, esto se hace **reiniciando** la población. El reinicio se puede llevar a cabo de varias formas:

- La más sencilla es crear otra vez la población al azar.
- Otras estrategias utilizan como **semillas** individuos prometedores y se someten a mutación (variaciones para no copiar directamente estos individuos).
- En otros casos se "vetan" las regiones ya exploradas y se crean los nuevos individuos de las nuevas regiones sin explorar.

Hay compañeros que en la respuesta han hecho referencia a estrategias orientadas a mantener la diversidad de la población. Esto no es correcto en este caso, porque ya se habla de que el método ha quedado atrapado en un óptimo local. Sin embargo, me ha parecido interesante su aportación y la he dado como válida.

6. Imagina que quieres llevar a cabo la función de la función de Ackley (d=5)

$$f(\mathbf{x}) = -a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)$$

con un algoritmo genético que utiliza codificación real. La búsqueda se realizará en el intervalo [-5.12, 5.12] extendido a las 5 dimensiones del problema. Propón una estrategia de inicialización de la población que facilite la búsqueda del óptimo global. NOTA: La función tiene un único mínimo en $\boldsymbol{x} = (0,0,\dots,0)$.

La respuesta más obvia, pero que no aporta mucho, es decir que para cada individuo de la población se generan cinco valores aleatorios en el intervalo [-5.12, 5.12] con un generador aleatorio uniforme (la función rand() de toda la vida). Sin embargo, a menos que la población sea muy grande, no tenemos garantías de que los individuos que se generen estén distribuidos uniformemente. Una solución es definir una estrategia de rejilla, definiendo subintervalos dentro del rango anterior. Por ejemplo [-5.12, -2.56); [-2.56, 0); [0,2.56) y [2.56, 5.12]. Establecidos estos subrangos, construimos una rejilla en la que cada celda será una combinación de valores, y dentro de esa celda generamos una serie de individuos al azar. Ilustrémoslo con un ejemplo en dos dimensiones

Como puede verse, en dos dimensiones tenemos una rejilla con 16 celdas (4x4 intervalos). Si tenemos una población de 160 individuos, obligaré a que cada celda contenga 10 individuos. Por tanto, generaré 10 individuos con valores [-5.12, -2.56) y [-5.12, -2.56) para los dos genes, [-2.56, 0) y [-5.12, -2.56) y así sucesivamente. De este modo, es mucho más probable que obtengamos individuos cerca de la solución buscada que de la otra forma.

	[-5.12, -2.56)	[-2.56, 0)	[0,2.56)	[2.56, 5.12]
[-5.12, -2.56)				
[-2.56, 0)		X	X	
[0.00,2.56)		X	X	
[2.56, 5.12]				

7. Explica las ventajas e inconvenientes de resolver un problema de optimización multiobjetivo mediante un algoritmo de agregación y mediante un algoritmo que se basa en el uso del Frente de Pareto. En los segundos, explica el papel que desempeña el uso de una población élite dentro del algoritmo.

   Ya se ha indicado en alguna ocasión que utilizar agregación de objetivos tiene la ventaja de que estamos convirtiendo el problema multi en mono objetivo, de forma que se pueden utilizar múltiples metodologías para intentar resolverlo. Sin embargo, a menos que tengamos clara qué importancia relativa tiene cada uno de los objetivos (pesos de agregación) es mucho más interesante disponer del frente de Pareto completo. Cuando lo tengamos, podemos probar las distintas soluciones y ver cuál es la mejor para nuestras circunstancias, o incluso se podrían usar varias.

   Con respecto a la población élite, ya explicamos en la revisión del otro examen por qué es importante.

8. El problema de la paridad par consiste en tomar un conjunto de bits e indicar si el número de unos es par. Por ejemplo, paridad_par(1,0,0,0,1,0,1,0) = FALSO. Deseamos encontrar una función

   Paridad(B0, B1, B2, …, B9)

   que recibirá como argumentos los 10 bits del conjunto y devolverá TRUE o FALSE en función que se cumpla la condición planteada. ¿Cómo resolverías dicho problema mediante programación genética **canónica**?

   Si vamos a usar GP canónica, habrá que definir los nodos terminales (B0 a B9), los nodos función (AND, OR, NOT es suficiente, aunque otros podrían plantear usar otras funciones como XOR). Luego habría que definir otros aspectos del algoritmo como la profundidad máxima de árbol (ha de ser suficiente para albergan la función correcta – a priori no se cuál es), el esquema de creación de la población inicial, qué operadores de cruce y mutación se utilizarán y con qué probabilidad, y qué esquema algorítmico se utilizará.

   Con respecto a la evaluación de los individuos, este es un problema muy parecido al de la regresión simbólica. La diferencia es que, en lugar de obtener un valor real se obtiene un valor binario: 1 (TRUE) o 0 (FALSE).