Metaheurísticas. Examen final. Junio de 2019

1. Describe cómo resolverías el problema del viajante de comercio mediante una búsqueda tabú. Explica cómo utilizarías la estructura de datos de nodos visitados para (1) fomentar la exploración de nodos no explorados y para (2) potenciar la explotación de los nodos más prometedores.

   Para responder a esta respuesta correctamente tendríais que abordar los siguientes puntos:
   - Dar una breve reseña indicando qué es una búsqueda tabú. 2-3 líneas.
   - Explicar cómo es la estrategia tabú para nodos visitados.
     - Explicar cómo un nodo se convierte en tabú y que tiene que estar varias iteraciones sin ser explorado.
     - Explicar también que la parte inferior de la matriz puede usarse en la selección de nodos/caminos más prometedores.

   Y todas estas explicaciones centrarlas en el problema del TSP.

2. Explica cómo resolverías un problema MO, por ejemplo, intentar optimizar la velocidad media y el consumo de un transporte público, con un algoritmo de *simulated annealing* o cualquier otro de los algoritmos que conoces de búsqueda basada en trayectorias.

   La única forma de modelar el problema de forma razonable es usar **agregación**. Usar dominancia de Pareto no tiene sentido porque, como solo nos quedamos una solución, todas las soluciones no dominadas estarían al mismo nivel y el algoritmo se detendría una vez encontrada una. Por lo demás, una vez definido $fitness = \sum_{i=1}^{N} \alpha_i fitness_i$, el problema es trivial.

3. Presenta un problema que **no** sea fácil de resolver mediante un algoritmo de ACO, y justifica por qué no es apropiado para este tipo de algoritmos.

   Se trata de explicar que ACO, al igual que otras heurísticas basadas en trayectorias, favorecen los problemas relacionados con validación de un camino, como el del viajante de comercio. Otros problemas, donde lo normal es tener todos los elementos de una vez, como el de la mochila, se resuelven, pero de un modo menos natural. Cualquier otro ejemplo en el que sea más sencillo más sencillo generar la solución de una vez que hacerlo por etapas/estados es peor con ACO que otros que consistan en ir de un estado a otro.

4. Imagina que quisiéramos encontrar los valores máximos de la función

$$f(x) = 3e^{-(x-4)^2} + 4e^{-(x-9)^2}$$

   En el intervalo (0,12). Propón la forma más apropiada de resolver el problema de manera que obtengamos todos los máximos (absoluto y relativo) en este intervalo. **PISTA**: En este intervalo hay dos máximos.

   La clave aquí es identificar este problema como multimodal y decir que se pueden usar para resolverlo algoritmos tales como el *fitness-sharing*, el *clearing* o cualquier otro para resolverlo. El resto es sencillo: codificación real (la más sencilla) y operadores típicos de optimización real (BLX-α y cualquier mutación real).

5. ¿Consideras importante el uso de una población auxiliar (élite) en los algoritmos multiobjetivo? ¿Por qué? ¿Qué algoritmos conoces que hagan uso de ella?

   La población élite es **crucial** en los algoritmos multiobjetivo. Tanto es así que, cuando NSGA se modifica añadiéndole elitismo, iguala en rendimiento a SPEA. La población auxiliar es una gran ventaja porque almacena soluciones no dominadas, permitiendo que estas participen en la evolución, pero que no ocupen espacio en la población normal para no quitar diversidad.

6. Considera un algoritmo genético con codificación real en el que sólo utilizáramos un operador de cruce (no mutaciones). Justifica las ventajas de utilizar un cruce BLX-α frente al cruce aritmético.

   El hecho de quitar la mutación significa que se pierde el operador que realiza **exploración**. La única forma de que un algoritmo genético pueda obtener soluciones a un problema complejo en estas condiciones es que el operador de cruce pueda también explorar el espacio de estados, no solo explotar. El cruce aritmético es incapaz de explorar dado que los hijos son promedio de los padres. Sin embargo, el cruce BLX-α sí que realiza una exploración fuera del rango

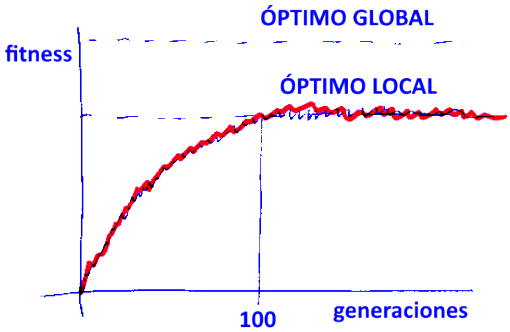
de valores de los padres. De hecho, α modela ese cambio. Por tanto, BLX-α es mucho mejor y podría alcanzar soluciones con más facilidad que el cruce aritmético.

7. Considera un algoritmo de programación genética. ¿Crees posible encontrar una solución a un problema dado, por ejemplo, el de la hormiga artificial o el de la regresión simbólica, sólo con operaciones de cruce? ¿Y sólo con mutaciones?

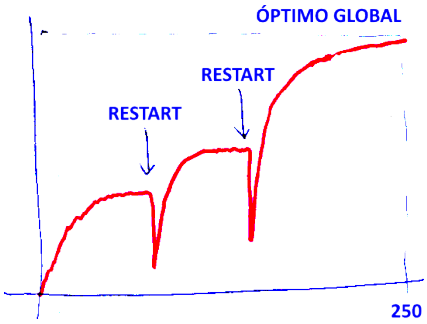
   En programación genética los cruces son muy diferentes de los de los algoritmos genéticos, producen hijos que pueden ser muy diferentes a los padres (de hechos, hay autores que lo consideran otra forma de mutación). Por tanto, si la población contiene los bloques constructivos (funciones y terminales) suficientes para alcanzar la solución al problema, es posible alcanzarla solo con cruces. Con respecto a la mutación, por supuesto que también será posible, lo que pasa es que no aprovechamos la "ventaja" de combinar 2 soluciones en otra nueva. Pero sí, podría obtener soluciones solo con mutación.

8. Dibuja gráficas que representan el fitness promedio de la población a lo largo de las distintas generaciones en las siguientes circunstancias (justifica tus respuestas):

   a. Un algoritmo que queda atrapado en un óptimo local y cuyo fitness no progresa después de la generación 100.
   b. Un algoritmo que necesita ser reiniciado en las generaciones 50 y 100 porque parece quedar atrapado en óptimos locales, y que alcanza el óptimo global en la generación 250.



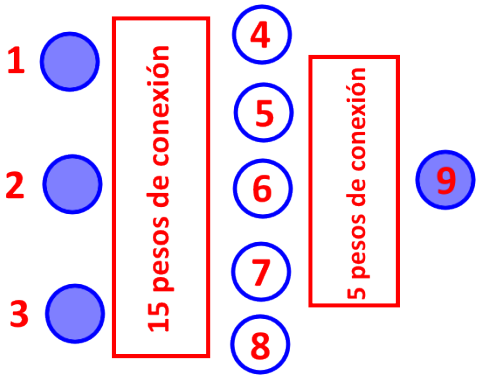
(a)



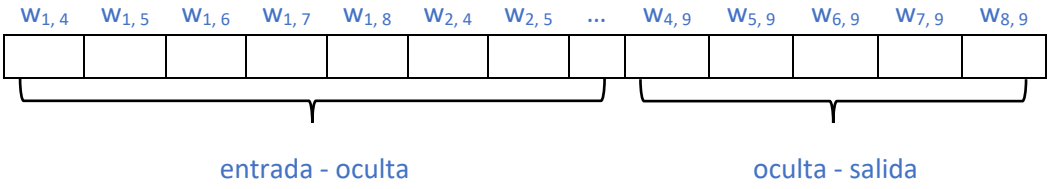
(b)

9. Imagina que quieres codificar la arquitectura de una red neuronal artificial para optimizarla mediante una metaheurística. Haz la propuesta que te parezca más apropiada (sólo neuronas en las capas ocultas, neuronas y conexiones, neuronas, conexiones y valores de pesos). Una vez explicada la codificación, explica con qué algoritmo llevarías a cabo dicha optimización.

   Vamos a suponer una red sencilla con 3 entradas y 1 salida. Si queremos representar arquitecturas de entre 1 y 5 neuronas en la capa oculta (con una única capa oculta), considerando que el número máximo de conexiones es 15 (3x5) + 5 (5x1) = 20, podría usar un vector de valores reales para representar estos pesos de conexión.



   El esquema de abajo muestra el cromosoma que utilizaríamos para representar esta red neuronal. Con este vector tenemos representada la red con 5 neuronas ocultas. Si os fijáis, el hecho de que todos los pesos que entran en una neurona oculta sean nulos significará que esa neurona no va a recibir entrada y será como anulada. Por otra parte, si los 5 pesos que llegan a la neurona de salida fueran nulos, la red tendría salida nula y sería trivial.



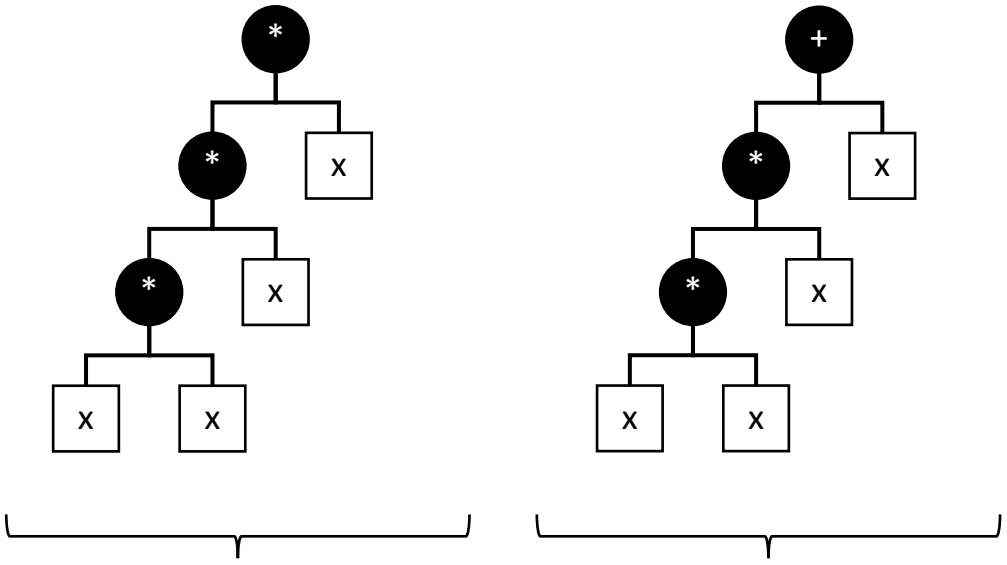
Esta representación tiene muchas pegas, pero es muy sencilla. Se puede complicar añadiendo una codificación binaria que sirva para quitar neuronas (equivalente a aumentar el valor del peso). Hay muchas más representaciones posibles. Lo que es fundamental es que represente algo coherente y que tenga un mínimo de sentido. Y la representación de esto con programación genética es muy complicada: no la recomiendo.

10. Imagina que quieres representar expresiones polinómicas de dos variables (x e y) y grado 3 como máximo. Es decir:

Expresiones válidas	Expresiones inválidas
• x^3 • xy^2 • $xy + y^2$ • $x + y + x^2 + y^2 + x^2y$	• x^4 • x^2y^2 • $x + y + xy^3$

¿Qué paradigma consideras más adecuado para representar estas expresiones? Razona tu respuesta y explica cómo llevarías a cabo dicha codificación.

El problema de codificar expresiones polinómicas es más complicado de lo que parece a simple vista. Usar solo la profundidad máxima de un árbol y codificarlo con programación genética no es buena opción. Veamos este ejemplo:



¡Y las 2 tienen profundidad máxima 3! Con programación genética basada en gramáticas el problema es parecido, porque para codificar estas expresiones no se puede usar una gramática de contexto libre. Bueno, sí puedes, pero no se puede controlar la forma de las expresiones. Quiero decir que habría que una gramática demasiado restrictiva que diferenciara monomios de grado 1, 2 y 3 por separado en lugar de solo monomios.

La forma más sencilla, curiosamente, es usar una codificación lineal:

$$P^3(x, y) = a_0 + a_1x + a_2y + a_3x^2 + a_4xy + a_5y^2 + a_6x^3 + a_7x^2y + a_8xy^2 + a_9y^3$$

Por tanto, para representar la expresión…

a_0	a_1	a_2	…	a_9

Con un vector de valores reales es suficiente para representar cualquier polinomio. Se podría también definir una doble codificación binaria/real donde el valor binario indicaría la presencia/ausencia del término correspondiente, pero esto complicaría los operadores genéticos. Y con respecto al algoritmo…un algoritmo genético con codificación real convencional es suficiente para resolver el problema de optimización.