Diplomatura de Especialización en Desarrollo de Aplicaciones con IA

Optimización Industrial con Computación Evolutiva

Ideas de proyectos finales

Idea 1: Resolviendo tableros Sudoku con Algoritmos Genéticos (3 integrante)

En este desafío se pide implementar un algoritmo genético para solucionar tableros de sudoku proporcionados. Para ello se les está dando un código base (Sudoku_HillClimbing) que contiene una implementación del algoritmo HillClimbing para solucionar un tablero dado. En dicho código ya se encuentra implementado una función para evaluar tableros (score_board())

5	3				7			
6			1	9	5			
	9	8					6	
8					6			3
4			8		3			1
7					2			6
	6					2	8	
			4	1	9			5
					8		7	9

Los individuos de la población inicial pueden ser creados con el método randomAssign()
Se sugiere representar cada individuo (tablero) con un cromosoma de 9 genes, cada gen representando un bloque diferente del tablero. Mantener siempre **alelos válidos** (bloques con números diferentes pero respetando las posiciones de los números iniciales), es decir, cada gen debería representar una permutación de 9 números.
El operadorde mutación debe alterar solo un gen del individuo (escogido aleatoriamente) y debe producir un nuevo alelo válido (respetando los números fijos del tablero original). Puede usar para ello la función make_neighborBoard()en el código dado

Se pide:

- Implementar al menos dos operadores de cruzamiento que produzcan individuos legales (todos los genes siendo permutaciones). Se le está proporcionando 5 tableros para hacer pruebas
- Elaborar experimentos para evaluar el AG con los diferentes operadores de cruzamiento implementados (deshabilitar mutación para ello) y luego con mutación habilitada a diferentes tasas de mutación. Se sugiere 10 individuos como tamaño de población. Para cada configuración a evaluar se debe hacer al menos 10 corridas para sacar conclusiones estadísticamente válidas. Como criterios de evaluación puede usar nroo de generaciones para llegar a convergencia, fitness máximo, nro de llamadas a la función Fitness para conv.
- Realizar el análisis de resultados y elaborar sus conclusionessobre el desempeño de los operadores y efectos de parámetros y cuál sería la configuración más adecuada para resolver tableros de sudoku.
- Informe de lo realizadoy notebookscon las implementaciones

Idea 2: Usando computación evolutiva para encontrar localizaciones optimas de una cadena de supermercados (3 integrantes)

Una conocida cadena de supermercados desea ingresar al mercado limeño. Usted, como experto en IA, ha sido contratado para proponer donde deben ser localizados los supermercados. Para ello, se le está proporcionando un mapa de la ciudad con 60 locales candidatos donde la cadena de supermercados podría localizar sus supermercados. Dicho mapa puede encontrarlo en: https://drive.google.com/open?id=1w7n77ByWK6TrX74lOBTwlD719LRj_UQz&usp=sharing

La cadena tiene presupuesto para instalar 10 supermercados. Se le está proporcionando también una tabla con las coordenadas de las localizaciones candidatas y la población estimada que hay en 500 metros a la redonda (Candidatos_supermercados.xlsx).

Su objetivo es escoger los supermercados de manera que se maximice la suma de la población que vive a 500m alrededor de los supermercados y la suma de las distancias entre los supermercados escogidos.

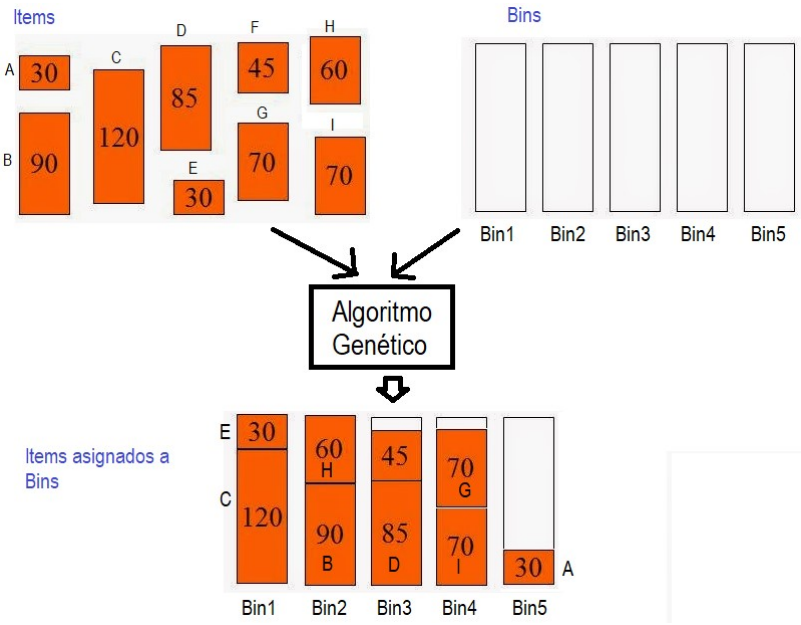
Para resolver dicho problema usted debe implementar:

1) **Un algoritmo genético mono-objetivo**. Cada individuo debe representar una colección de 10 localizaciones. Se sugiere que el cromosoma de un individuo sea un string de 60 bits, cada bit representando si una localización esta seleccionado o no. Para evaluar un individuo se puede usar como función *fitness* la suma de distancias entre todos los posibles pares de localizaciones escogidas en el individuo (45 pares distintos) más la suma de las poblaciones alrededor de cada supermercado escogido. Implemente operadores de cruzamiento, mutación y selección adecuados para garantizar que los individuos resultantes sean válidos (representen colecciones de 10 localizaciones). Experimente varias veces el algoritmo implementado con una población de 100 individuos y 500 generaciones (se sugiere ejecutar 10 veces) y registre en una tabla los mejores individuos de cada ejecución, sus *fitness* y los componentes del fitness desagregados (suma de distancias y suma de población).
2) **Un algoritmo genético multiobjetivo (NSGA-II).** La representación de los cromosomas seria la misma que el caso mono-objetivo. Los objetivos serian: a) suma de distancias entre todos los posibles pares de localizaciones escogidas en el individuo; y b) suma de las poblaciones alrededor de cada supermercado escogido en el individuo. Los operadores de cruzamiento, mutación pueden ser los mismos del caso mono-objetivo. Experimente varias veces el algoritmo implementado con una población de 100 individuos y 500 generaciones (se sugiere ejecutar 10 veces). En cada ejecución registre los individuos de la frontera de Pareto final, así como los valores de las funciones objetivos de dichos individuo. Realice Plots de las fronteras de Pareto encontradas, ubicando en dichos plots los puntos de los mejores individuos encontrados con el algoritmo mono-objetivo.

Se debe entregar notebooks con las implementaciones e Informe con descripción del algoritmo mono-objetivo y multi-objetivo implementado (representación de individuos, operadores de cruzamiento, mutación, etc). Tablas de resultados y análisis de los mismos para el algoritmo mono-objetivo.Plots de resultados y análisis de los mismos para el algoritmo multi-objetivo.Comparación de resultados presentados en b) y c) y sus conclusiones respecto a su idoneidad en el problema abordado.

Idea 3: Usando computación evolutiva para asignación de ítems a bins (3 integrantes)

En este proyecto se pide implementar un algoritmo evolutivo para asignar un conjunto de ítems de diferentes tamaños a un conjunto de Bins. Cada Bintiene un tamaño máximo pre-especificado. La idea es que se minimice la suma total del tamaño de los ítems que no pudieron ser asignados junto con la suma total de áreas no usadas en cada Bin



Se pide:

- Plantear un diseño del algoritmo genético, proponiendo una representación adecuada de los individuos y operadores genéticos. Al menos dos operadores de cruzamiento y dos de mutación deberían ser probados.
- Elaborar experimentos para evaluar el AG con los diferentes operadores de cruzamiento implementados (deshabilitar mutación para ello) y luego con la mutación habilitada a diferentes tasas de mutación. Se sugiere 10 individuos como tamaño de población. Para cada configuración a evaluar se debe hacer al menos 10 corridas para sacar conclusiones estadísticamente válidas. Probar bajo diferentes escenarios de cantidades de bins e ítems. La generación de ítems también puede ser hecha partiendo los bins en n pedazos y corriendo el AG para ver como reconstruye
- Realizar el análisis de resultados y elaborar sus conclusiones sobre el desempeño de los operadores y efectos de parámetros y cuál sería la configuración más adecuada para resolver tableros de sudoku.
- Informe de lo realizado y códigos con las implementaciones

Idea 4: Aprendiendo a pintar con computación evolutiva (3 integrantes)

Se tiene un algoritmo evolutivo que evoluye imágenes construidas con polígonos semitransparentes a fin de obtener una imagen semejante a una imagen de referencia. El código se encuentra en https://github.com/miezekatze/evolisa. Tal como está implementado el algoritmo la evolución sucede mediante operaciones de mutación sobre un único individuo.

Se pide:

- Implementar una versión poblacional de dicho algoritmo con al menos un operador de cruzamiento
- Elaborar experimentos para evaluar el AG con el operador(es) de cruzamiento implementado (deshabilitar mutación para ello) y luego con la mutación habilitada a diferentes tasas de mutación. Se sugiere 10 individuos como tamaño de población. Para cada configuración a evaluar se debe hacer al menos 10 corridas para sacar conclusiones estadísticamente válidas.
- Realizar el análisis de resultados y elaborar sus conclusiones sobre el desempeño de los operadores y efectos de parámetros y cuál sería la configuración más.
- Informe de lo realizado y códigos con las implementaciones

Otras Ideas:

- **Programación de horarios de cursos universitarios (3-4 integrantes)**

  - Aquí un código base: https://github.com/NDresevic/timetable-generator/ . Se puede proponer un algoritmo genético mono-objetivo, o multiobjetivo. En este caso los objetivos podrían ser: minimizar el tiempo de pausa entre clases para los grupos de alumnos y minimizar el tiempo de pausa entre clases de los profesores

- **Selección de Atributos en Aprendizaje de Máquina (2 integrante)**:

  - How To Use Genetic Algorithms As A Tool For Feature Selection In Machine Learning(https://analyticsindiamag.com/how-to-use-genetic-algorithms-as-a-tool-for-feature-selection-in-machine-learning/)

- **Optimización continua en simuladores biomecánicos:(3-4 integrantes)**

  - Learning to walk with genetic algorithms- It used the OpenSim environment (https://github.com/stanfordnmbl/osim-rlfor biomechanical simulations, modified for the NIPS 2017
  - "Learning to Run" competition. Here you can find their repository with a detailed introduction https://github.com/normandipalo/learn-to-walk-with-genetic-algs/

- **Neuroevolución para crear agentes de videojuegos :(3 integrantes)**

  - Creando agentes jugadores de Mario Bros con NEAT (https://github.com/vivek3141/super-mario-neat, https://github.com/rae0924/pymar-i-o,https://github.com/gustavohb/super-mario-neat)
  - Creando Agentes para el juego FlappyBird con neuroevolucionhttps://github.com/ashmeet13/Flappy-Bird-Genetic-Algorithm-
  - Agente de programación genética para el juego del Snake: https://github.com/Kripperoo/Genetic-Programming-for--Snake-

- **Música evolutiva:(3 integrantes)**

  - Programacion genética y redes neuronales para crear música: https://github.com/cagatay/Evolution-9