Algoritmos Genéticos – Optimización Multi-Objetivo

Objetivos

En esta sesión se pretende que el alumno continúe resolviendo problemas reales en los que la aplicación de algoritmos de computación evolutiva necesitan de algunas adaptaciones adicionales para su utilización.

En este caso, se pretende que el alumno refuerce el conocimiento adquirido a cerca del planteamiento de las evaluaciones de las soluciones. En concreto se centrará en las opciones que de nos proporciona la librería escogida para el tratamiento de restricciones sobre el problema, así como otras representaciones de los individuos

Descripción del Problema

El problema propuesto consiste en generar un programa en python que permita resolver el problema de la asignación de vehículos a pasajeros, incluyendo ahora una extensión que tiene en cuenta los vehículos adaptados a personas con problemas de movilidad. El problema se describe en el documento adjunto al enunciado (ver al final del documento).

Para realizar este programa, el alumno deberá emplear la librería *Distributed Evolutionary Algorithms in Python* (DEAP) para ayudarse en las partes más genéricas del algoritmo genético.

Se dispone de una ayuda completa de la librería en el siguiente enlace. Se puede consultar el tutorial más básico a cerca del funcionamiento de la librería en el siguiente enlace.

Trabajo del Alumno

Se solicita implementar algunas variaciones al algoritmo pre-definido que se empleó en anteriores sesiones.

Se han visto en teoría diferentes opciones a la hora de trabajar con problemas en los que se busca una solución que no solo sea óptima en cuanto a una definición en el problema, sino a varias definiciones a la vez (probablemente encontrando un balance entre ellas):

- En el tema 9 se estudiaron formas de incluir restricciones en la definición de la evaluación de las soluciones.

- El tema 10 se dedicó íntegramente al estudio del planteamiento de estas características como problemas multi-objetivo.

En esta práctica se pide al alumno que explore ambas opciones, incorporándolas a la solución que se aporta al problema.

Se solicita también al alumno que incorpore un estudio comparativo de las dos variantes de solución, para comprobar si existen diferencias apreciables.

Implementación

Se pide al alumno implementar dos nuevas funciones que permitan realizar la evolución que se ha solicitado:

1.- Evaluación de Restricciones

La primera variación consiste en incorporar el concepto de restricción en la evaluación de las soluciones.

Para ello, basta con ampliar la función de evaluación.

En el caso del problema planteado, la función se deberá re-definir para que se busque la **solución óptima** que maximice el valor de los viajes realizados, respetando las restricciones de tiempo que se adjuntan a cada problema.

Si bien este concepto se tenía que incluir de manera implícita dentro de la evaluación, se considera por lo general aproximaciones más interesantes el tratarlas por separado. Se puede encontrar el material y ejemplos para comprobar cómo se implementa este concepto en DEAP en el siguiente enlace. Es obligatorio en este caso tanto la **decisión sobre de la validez** de un individuo como la de la **función de distancia** entre el individuo no válido y la región válida. Se puede consultar también información en la Referencia de la Biblioteca.

2.- Optimización Multi-Objetivo

Se solicita modificar, por separado, el modelado del problema para definirlo de forma que se intenten optimizar dos objetivos (inversamente relacionados) como:

El tiempo dedicado a completar los viajes asignados a cada vehículo y el número de viajes que se realizan en un vehículo que no está adaptado.

Para ello se deberán realizar las siguientes modificaciones:

a) Modificar la definición de los individuos para que se les asocie 2 o más funciones de adaptación, en lugar de una.

Comprobar cómo se realiza en el tutorial del enlace, en la sección *"A First Individual"*. Se debe tener en cuenta solo la parte en que define el fitness, ya que el genotipo del individuo deberá adaptarse al problema concreto.

b) Modificar la función de evaluación, para que en lugar de proporcionar un único valor de adaptación para cada individuo, ahora proporcione 2 valores.

   Comprobar cómo se realiza en el tutorial del enlace, en la sección *"Evaluation"*. Se necesita modificarla para que se devuelva una tupla de 2 valores.

c) Modificar la configuración de la evolución de la especie para que pueda trabajar con funciones multi-objetivo (en teoría se han detallado los algoritmos SPEA-2 y NSGA-II).

   En la práctica original se disponían de varios mecanismos de selección de individuos para mono-objetivo (Torneo, Ruleta, Estocástico,…). Cuando se trabaja con multi-objetivo, la selección de los individuos se debe hacer atendiendo al concepto de dominancia de Pareto. Para ello, se disponen de diferentes algoritmos en el módulo deap/emo.py. Algunos de ellos son selNSGA2(), selTournamentDCD() o selSPEA2(). Se puede consultar también información en la Referencia de la Biblioteca.

d) Modificar la configuración de la recogida de estadísticas, para que recoja las de 2 o más funciones, en lugar de solo 1. Además, para comprobar el funcionamiento del algoritmo, se deberán dibujar los avances de cada de las funciones por separado. Ver el apartado "Multi-Objective Statistics" en el enlace.

   Se pide también en el informe gráficas que representen la situación de la población y el frente de pareto al final de la evolución.

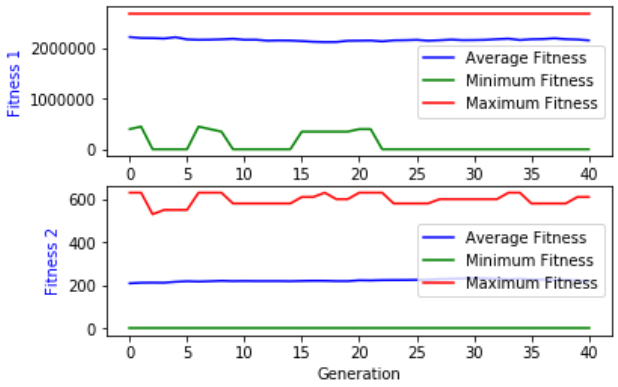

Ilustración 1: Representación de la evolución 2 funciones optimizadas a la vez (F1 maximizando y F2 minimizando)

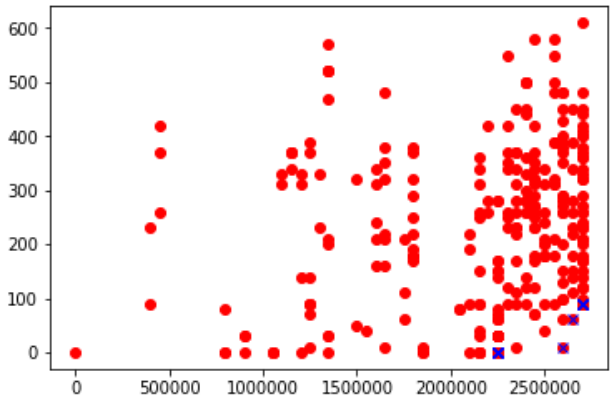

Ilustración 2: Representación de la población completa en función de 2 de las funciones de fitness. Se marca el frente de pareto en azul.

Informe

Incluye un fichero en formato PDF que incluya el contenido de los resultados obtenidos en los experimentos, que deberá incluir una comparativa de las variantes que se hayan probado en los algoritmos propuestos: optimización con restricciones y optimización multi-objetivo.

En cada variante, se deberán realizar diversos experimentos que permitan comprobar tendencias y se deberá tener en cuenta que existen varias situaciones del problema a resolver.

En el modelo de informe que se incluye, aparece una sección detallada de cómo se espera que se realicen los experimentos.

En el informe, esta comparativa deberá incluir:

- Descripción de los ensayos que se han realizado
- Resultados numéricos y/o gráficos obtenidos en cada uno
- Conclusiones de los alumnos sobre dichos resultados

Recursos Necesarios

Se adjunta al enunciado:

- Carpeta de “Material”, que incluye la estructura básica de la entrega que se pide al alumno. Dentro de los ficheros se pueden encontrar descripciones más completas de la entrega que se solicita.

  El alumno puede incluir ficheros fuentes y material adicional, pero es importante que en los ficheros fuente facilitados incluya las llamadas a los ficheros fuentes adicionales que se decida incluir. Para la corrección de los ejercicios, se ejecutará el fichero que se entregue con el nombre de `principal.py` en cada variante.

Se puede acceder al repositorio de la librería DEAP en el siguiente enlace. Se puede consultar las instrucciones de instalación en la página principal.

La Documentación y Tutoriales de la librería DEAP son accesibles en el siguiente enlace.

Consideraciones de Entrega

La entrega se empleará para obtener la evaluación final la primera práctica de la parte de “Computación Evolutiva” de la asignatura (Práctica 04).

La entrega se realizará en grupos de 1 o 2 alumnos.

Se incluirán en la entrega:

- Ficheros fuente en lenguaje python que resuelven el problema de la asignación de vídeos a las diferentes cachés.
- El fichero en PDF describiendo la solución propuesta.

Ambos ficheros se comprimirán en un .zip, cuyo nombre seguirá el formato: `ApellidoAlumno1_NombreAlumno1-ApellidoAlumno2_NombreAlumno2.zip`

La entrega se realizará solo por medio del correspondiente espacio de subida en plataforma UBUVirtual. No se admitirán entregas pasado el plazo establecido.

En caso de realizarse en pareja, ambos alumnos deberán realizar la entrega.

Problema de la Asignación de Vehículos

Descripción del Problema

El problema inicial planteado es el del problema de la asignación de vehículos a viajes solicitados previamente, incluido en el desafío HashCode'18.

Para este ejercicio, se plantea una extensión del mismo. Sin modificar las condiciones iniciales del problema, ahora se quiere introducir una nueva situación a tener en cuenta: si los viajes a realizar han sido solicitados por una persona con movilidad reducida. En este caso, cada viaje tendrá un nuevo identificador binario que informará sobre la condición del viajero: con movilidad reducida o no.

En cuanto a los vehículos, también tendremos ahora información adicional, ya que no serán todos iguales. En la flota de vehículos se disponen de ciertos vehículos adaptados, en un cierto porcentaje sobre el total. A la hora de asignar un viaje a un vehículo, se tendrá en cuenta esta condición.

Se considerará una solución más óptima aquella que asigne a viajes solicitados por personas con alguna discapacidad, vehículos adaptados. No se considera que esta asignación como obligatoria, ya que el conductor puede ayudar a subir y bajar a los pasajeros, pero sí se considera muy deseable para su mejor comodidad. Por ese motivo, se considerará esta condición como una restricción secundaria sobre el problema.

El alumno deberá valorar dos opciones para tener en cuenta esta situación adicional: modelarla como una restricción o establecer que se trata de algo más prioritario y tratarlo como un problema de dos objetivos

Ficheros de Datos

El formato del fichero de datos se respetará principalmente del que se facilita en el desafío HashCode'18. Solamente se añaden dos apartados nuevos:

1. En la información de cada viaje, aparece un último valor booleano, que indicará si la persona solicitante sufre de movilidad reducida: '0' indicará que no existe ningún problema, mientras que el '1', que sí existe.
2. La última linea del fichero describe a la flota de vehículos. Se trata de un array de valores booleanos con una longitud igual al número de vehículos que se incluyen en la flota. Se considerará que un '0' significa que el vehículo identificado con el valor de esa posición del array no está adaptado para personas de movilidad reducida, mientras que el '1' significa que sí se encuentra adaptado.

Ejemplo de fichero de entrada:

`3 4 2 3 2 10`	`3 fila, 4 columnas, 2 vehiculos, 3 viajes, 2 bonus y 10 pasos`
`0 0 1 3 2 9 0`	`viaje del [0, 0] al [1, 3], inicio más temprano 2, fin más tardio 9, persona con movilidad normal`
`1 2 1 0 0 9 1`	`viaje del [1, 2] al [1, 0], inicio más temprano 0, fin más tardio 9, persona con movilidad reducida`
`2 0 2 2 0 9 0`	`viaje del [2, 0] al [2, 2], inicio más temprano 0, fin más tardio 9, persona con movilidad normal`
`1 0 0`	`vehículo 0: adaptado, vehículo 1: no adaptado, vehículo 2: no adaptado`

Se considerará como salida un fichero con el mismo formato que se solicita en el enunciado del HashCode'18.

`1 0` `2 2 1`	`Al vehículo 0 se le asigna 1 viaje: [0]` `Al vehículo 1 se le asignan 2 viajes: [2, 1]`

En este caso, vemos que la puntuación original es de 10 (el primer viaje gana 6 puntos y los otros 2, 2 ptos. cada uno). Sin embargo, ahora tendríamos que tener en cuenta que al vehículo 1 se le asigna un viaje que lleva a un pasajero con movilidad reducida sin estar adaptado, contraviniendo una restricción del problema.