



PEC 3 – Optimización

Presentación

En esta PEC diseñaremos un proceso basado en algoritmos genéticos para optimizar la ubicación de una serie de generadores eléctricos a lo largo de una línea de transmisión.

Competencias

En este enunciado se trabajan en un determinado grado las siguientes competencias general de máster:

- Capacidad para proyectar, calcular y diseñar productos, procesos e instalaciones en todos los ámbitos de la ingeniería en informática.
- Capacidad para el modelado matemático, cálculo y simulación en centros tecnológicos y de ingeniería de empresa, particularmente en tareas de investigación, desarrollo e innovación en todos los ámbitos relacionados con la ingeniería en informática
- Capacidad para la aplicación de los conocimientos adquiridos y de solucionar problemas en entornos nuevos o poco conocidos dentro de contextos más amplios y multidisciplinares, siendo capaces de integrar estos conocimientos.
- Poseer habilidades para el aprendizaje continuado, autodirigido y autónomo.
- Capacidad para modelar, diseñar, definir la arquitectura, implantar, gestionar, operar, administrar y mantener aplicaciones, redes, sistemas, servicios y contenidos informáticos.
- Capacidad para asegurar, gestionar, auditar y certificar la calidad de los desarrollos, procesos, sistemas, servicios, aplicaciones y productos informáticos.



Las competencias específicas de esta asignatura que se trabajan son:

- Entender que es el aprendizaje automático en el contexto de la Inteligencia Artificial
- Distinguir entre los diferentes tipos y métodos de aprendizaje
- Aplicar las técnicas estudiadas en un caso concreto

Objetivos

En esta PEC se practicarán los conceptos del temario relacionado con optimización, en una vertiente práctica con datos sintéticos.

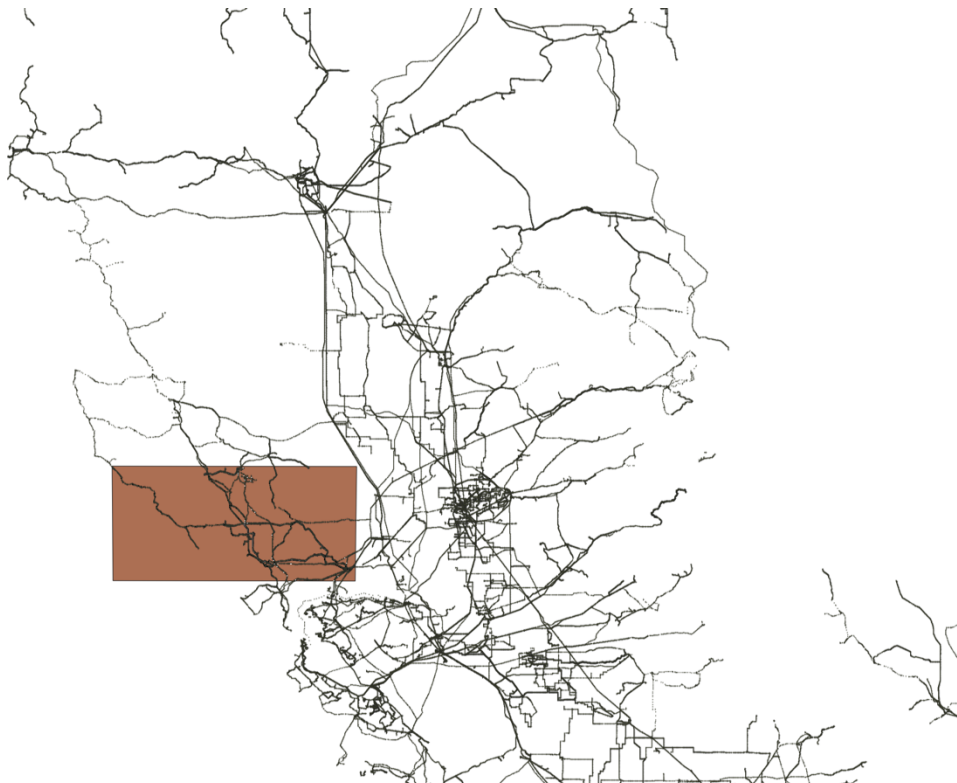
Descripción de la PEC a realizar

Contexto

Las últimas oleadas de incendios en California han llevado a una de las mayores compañías eléctricas de USA a la bancarrota, debido a haber sido sentenciada como culpable por negligencias en el mantenimiento de sus líneas, lo cual se demostró ser la causa de los incendios más importantes (<https://edition.cnn.com/2019/01/29/business/pgc-bankruptcy-fires/index.html>).

Debido a esto, el gobierno del estado licitó el servicio de suministro eléctrico en todos sus condados. En el caso de los condados de Sonoma y Napa, este servicio cae en manos de la empresa ElectrIAA, especializada en la aplicación de Inteligencia Artificial Avanzada para la optimización de todos sus procesos.

El panorama que se encuentra esta joven, aunque ambiciosa, empresa al inspeccionar el estado de las anteriores instalaciones es desolador, dado que todas las subestaciones generadoras de la línea principal de suministro necesitan ser reemplazadas por motivos de seguridad. Tan solo se trata de 20 estaciones, pero son las encargadas de dar suministro a 1000 urbanizaciones, repartidas por todo el mapa de estos condados.



A modo ilustrativo, la figura de arriba muestra la red eléctrica de alta potencia en la mitad norte de California. El rectángulo indica el área del contexto de esta PEC.

Para reemplazar las subestaciones, no hace falta que éstas tengan exactamente la misma ubicación que las anteriores, sino que pueden estar ubicadas en diferentes localizaciones a lo largo del tendido eléctrico que se ve en la figura. El fichero adjunto *transmissionLines_SonomaNapa_1000.txt* es una representación *raster* simplificada del mapa de posibles ubicaciones. Se trata de una grilla de 1000x1000 celdas, donde los valores 1 indican los lugares que reúnen las condiciones para poder instalar una nueva subestación.



Dada esta circunstancia, ElectrIAA decide echar mano de su experiencia en algoritmos evolutivos para optimizar la ubicación de las nuevas subestaciones, tratando de minimizar la distancia promedio entre las nuevas subestaciones y las 1000 urbanizaciones que han de abastecer.

Así pues, como expertos que sois, se os encarga diseñar e implementar unos algoritmos genéticos que cumplan de manera óptima con los requisitos del problema.

Como resultado de otros proyectos similares, contáis con unos unos códigos Python en los ficheros *P3.py* y *utils.py*, que se ajustan bastante a lo que necesitáis hacer, aunque están inacabados. Es importante que dediquéis tiempo a entender bien el código existente para realizar correctamente las actividades de a continuación. Consultad la documentación oficial de las librerías de todas las funciones que se utilizan, sobre todo las de *numpy* y *DEAP*. En este último caso, estudiad bien el manual proporcionado en el aula.

Actividades

La solución a esta PEC se diseñará y ejecutará con la librería DEAP de Python. Hay que justificar las decisiones tomadas y analizar los resultados obtenidos.

Actividad 1 (3 puntos)

Después de analizar brevemente los códigos y entender bien en qué consisten las variables `rand_urbanization_map` y `stations_locations`, así como las porciones de código destinadas a configurar el algoritmo genético y funciones de soporte, se pide:

a) 1.5 puntos. Explicar en qué consiste un individuo en el marco de este problema, cómo está representado, y rellenar los huecos existentes en las llamadas a las funciones de configuración (ver comentarios



#Exercise 1a: ...). Para la fase de selección, se realizarán torneos entre individuos, cogidos de 3 en 3.

Revisad bien qué se hace con el argumento *individual* en las funciones de *utils.py* para llegar a vuestras conclusiones, y tened en cuenta que los individuos se generan mediante la función `deap.tools.initIterate` (consultad la documentación de la librería).

b) 0.5 puntos. Configura el algoritmo genético con las diferentes combinaciones de los siguientes valores para las variables `N_POP`, `N_GEN`, `cx_prob` y `mut_prob`:

`N_POP`: 10, 25, 50

`N_GEN`: 10, 30

`cx_prob`: 0.2, 0.5

`mut_prob`: 0.2, 0.55

Mostrad las gráficas generadas que consideréis más relevantes para su discusión en el apartado siguiente.

c) 1 punto. Discutir el efecto de cada una de las variables (o la combinación entre ellas), según lo observado tras las ejecuciones del apartado anterior.

Actividad 2 (2 puntos)

Después de comprobar que, efectivamente, se puede optimizar la ubicación de nuestras subestaciones según el criterio de minimizar la distancia promedio, queremos ir un paso más allá y, en vez de definir un único objetivo, queremos optimizar nuestros resultados en base a un objetivo múltiple. En este caso se establecen 2 criterios extra:

- Que, por cuestiones de eficiencia en el mantenimiento de toda la red, además de minimizar el promedio de todas las distancias, se minimice también la desviación estándar.



- Que, por una cuestión logística, se minimice también la distancia promedio de todas las subestaciones a la estación principal de la compañía, que estará ubicada en la posición (0,0) del mapa (fuera de las ubicaciones posibles para el resto de subestaciones).

a) 1 punto. Completar las funciones `stdev_distance` y `avg_distance_to_main` en el fichero `utils.py` (fijáos en qué y cómo lo hace la función `avg_distance` que se os da ya implementada). Modificad también la función `fitness` para que devuelva los valores correspondientes a los 3 criterios (el inicial más los 2 nuevos).

b) 1 punto. Modificad las instrucciones necesarias para que el algoritmo genético ahora trabaje con individuos con una función multiobjetivo. Otorgad un peso de -1, -0.8 y -0.5 a los 3 diferentes criterios (`avg_distance`, `stdev_distance` y `avg_distance_to_main`, respectivamente), y comprobad que el algoritmo genético multiobjetivo funciona correctamente. ¿Por qué estos valores son negativos?

Actividad 3 (2.5 puntos)

Responded de manera argumentada a las siguientes cuestiones (cada una tiene un valor de 0.5 puntos)

a) En el código base (fichero `P3.py`) encontramos estas dos líneas:

```
seq = [i for i in range (0,999999)]
rand_urbanization_map = sample(seq,1000)
```

¿Qué implicaciones tienen, a la hora de evaluar los resultados de diferentes ejecuciones? ¿Sería útil contar siempre con una misma distribución de urbanizaciones en el mapa?

b) Si ejecutamos el código varias veces fijando una misma configuración de las del apartado 1b), enseguida vemos que no hay una varianza destacable entre ejecuciones, en cuanto a la velocidad de convergencia del algoritmo y valores alcanzados. Razonad qué nos indica este hecho.



c) Imaginad que, en vez de la codificación escogida para representar a los individuos, se escogiese una codificación binaria, consistente solo en valores 0 y 1. ¿Qué ventajas e inconvenientes tendría esto?

d) Como se puede observar, en nuestro caso la función registrada `'individual'` utiliza la función `toolbox.initIterate` de DEAP. ¿Qué diferencia hay entre ésta y la función `toolbox.initRepeat` que podemos ver en otros muchos ejemplos? ¿Por qué `initRepeat` no sería adecuada en nuestro caso?

e) En el código base (fichero P3.py) encontramos estas dos líneas de código de decoración, aplicado a las operaciones de cruzamiento y mutación:

```
# GA decoration
toolbox.decorate('mate', fulfill_constraints())
toolbox.decorate('mutate', fulfill_constraints())
```

¿Por qué no es necesario aplicar la decoración en la operación `'population'`, encargada de generar la población de individuos? Se recomienda consultar el manual de DEAP para entender el procedimiento de decoración.

Actividad 4 (2.5 puntos)

En el código proporcionado, la función de decoración `fulfill_constraints` tan solo consiste en un esqueleto que, al fin y al cabo, no hace nada. En este ejercicio vamos a completarla para que, efectivamente, decore a los individuos que se generan a partir de las operaciones de cruzamiento y mutación.

a) (0.5 puntos) Antes de ponernos a decorar, obviamente, nos tenemos que preguntar por qué es necesario hacerlo. Razonad a continuación qué aspectos de nuestros individuos hacen necesaria la decoración para asegurarnos de que el algoritmo genético funciona correctamente y nos dará resultados útiles.



b) (2 puntos) Completad el código de la función `fulfill_constraints`, de acuerdo a vuestra respuesta del apartado anterior.

Recursos

Tutorial de DEAP adjuntado en el aula. Video de la asignatura sobre optimización.

Ficheros de código adjuntos.

Complementarios

Manual de teoria de la asignatura.

Documentación de DEAP:

<http://deap.readthedocs.org/en/master/>

Criterios de valoración

La valoración asociada a cada ejercicio está indicada en el propio enunciado de cada uno.

Razonar la respuesta en todos los ejercicios. Las respuestas sin justificación no recibirán puntuación.

Formato y fecha de entrega

La PEC se ha de entregar antes del próximo **17 de mayo (inclusive)**.

La solución a entregar consiste en un informe en formato PDF usando este enunciado como plantilla más los archivos de código (*.py) que ha creado para resolver la prueba. Todos estos archivos se deben comprimir en un archivo ZIP.

Adjuntar el archivo a un mensaje en el apartado de Entrega y Registro de EC. El nombre del archivo debe ser ApellidosNombre_IAA_PEC3 con la extensión .zip.

Para dudas y aclaraciones sobre el enunciado, dirigirse al consultor responsable de su aula.



Nota: Propiedad intelectual

A menudo es inevitable, al producir una obra multimedia, hacer uso de recursos creados por terceras personas. Es por tanto comprensible hacerlo en el marco de una práctica de los estudios del Máster en Informática, siempre y esto se documente claramente y no suponga plagio en la práctica.

Por lo tanto, al presentar una práctica que haga uso de recursos ajenos, se presentará junto con ella un documento en el que se detallen todos ellos, especificando el nombre de cada recurso, su autor, el lugar donde se obtuvo y el su estatus legal: si la obra está protegida por copyright o se acoge a alguna otra licencia de uso (Creative Commons, GNU, GPL ...). El estudiante deberá asegurarse de que la licencia que sea no impide específicamente su uso en el marco de la práctica. En caso de no encontrar la información correspondiente deberá asumir que la obra está protegida por copyright.

Deberán, además, adjuntar los archivos originales cuando las obras utilizadas sean digitales, y su código fuente si corresponde.