23-1-2017

Módulo Python de Algoritmos Genéticos

Juan Pedro Torres Muñoz e Iván Barbecho Delgado

1 TABLA DE CONTENIDO

2 INTRODUCCIÓN

Para este proyecto se ha querido desarrollar una librería para la facilitar la resolución de problemas usando tecnicas de algoritmos geneticos. Para ello hemos hemos implementado nuestra solución en Python, debido a la facilidad que nos brinda este lenguaje para trabajar con las estructuras de datos.

El proyecto finalizado nos brindará varias facilidades para generar, configurar el algoritmo y resolver el problema, siendo un modulo muy generico por lo que el usuario deberá modelar su problema de forma que se ajuste al estilo de resolución de problemas del modulo.

3 ESTRUCTURA DEL MODULO

Este módulo estará formado por dos sub-módulos diferenciados, un paquete que contiene las características adicionales para poder trabajar con interfaz gráfica, con los métodos listos para configurar el algoritmo mediante una interfaz gráfica y otro paquete que incluye todas las clases necesarias para realizar el mismo funcionamiento en un script de python.

3.1 MÓDULO INTERFAZ GRÁFICA

El paquete usado para dar soporte a la interfaz gráfica está formado por las siguientes clases:

- Genetic Algorithm gui
- Gui

Estos ficheros son los encargados de generar la interfaz gráfica y obtener los datos de ella para poder hacer uso de los datos introducidos. Este paquete se apoya en la librería PySide, la cual es una versión para python que nos brinda todas las funcionalidades de Qt. Esto nos proporciona una capa de abstracción del módulo general y nos permite invocar a este paquete directamente, encargándose dicho paquete después de realizar las llamadas pertinentes al módulo.

3.2 MÓDULO PRINCIPAL

Para la implementación de la solución se ha dividido el proyecto en tres clases diferenciadas que tienen una función específica.

El modulo está finalmente compuesto por las siguientes clases:

- Chromosome
- Genetic algorithm attr
- Genetic algorithm

Con esta división se intenta conseguir que se pueda trabajar de la forma más genérica posible, trabajando siempre con elementos genéricos sin necesidad de conocer de que están formados.

La clase “Chromosome” contendrá la información relativa a un individuo y a los cromosomas de dicho individuo, dándonos así la genericidad en el código a la hora de trabajar con los individuos más adelante desde otras clases.

La clase “Genetic algorithm attr” es la clase encargada de generar la configuración usada para lanzar el algoritmo genético con una configuración dada, seleccionando en ella el tipo de reproducción de los individuos, el tipo de selección de dichos individuos y de controlar que los valores con los que se quiere lanzar el algoritmos son totalmente válidos, elevando una excepción en caso contrario.

La clase “Genetic algorithm” contiene la implementación de las diferentes variaciones en las técnicas de resolución que hemos añadido a la librería, desde los métodos de reproducción de individuos al bucle encargado de actualizar y regenerar todos los individuos entre generaciones. También incluye varios métodos para establecer el pool inicial con el propósito de realizar pruebas de rendimiento, de funcionamiento y que en caso de parar el trabajo poder guardar y volver a relanzarlo con el estado del pool al parar.

Todas estas clases disponen del comando ‘help’ de python, proporcionando la información necesaria para su uso.

4 FUNCIONAMIENTO

A continuación habrá una explicación de cómo hacer uso de este módulo de python.

4.1 INSTALACIÓN

Para poder utilizar nuestra librería necesitaremos tener la versión de Python 2.7 y un sistema operativo Linux. Una vez tengamos esto solo tenemos que descargar el contenido de la rama “Master” del repositorio. Una vez hecho esto ejecutaremos el script de instalación disponible.

4.2 EJEMPLO DE USO

Dentro del mismo repositorio, existen diferentes ejemplos, ahora se explicará el funcionamiento del más simple de ellos, el que no tiene interfaz gráfica.

Tras importar la librería, crearemos un objeto de la clase chromosoma, que tiene los siguientes atributos:

- N_chromosomes
- Maxvalue
- Minvalue
- Is_float

N_chromosomes hace referencia al número de cromosomas que tendrá cada individuo. Dichos valores se encontraran en el intervalo [minvalue, maxValue).

Is_float es una bandera para saber si los datos de los cromosomas son de tipo float, en vez de enteros.

A partir de ahí, se crearan los atributos con los que funcionará el algoritmo. Al crear dicho atributo se usaran las siguientes opciones.

- Función fitness
- Chromosomes
- Itmax
- Child type
- Selection type
- High low
- Size pool
- Percent elitism
- Percent mute
- Can repeat chro

Deberemos pasar la función que calculará el fitness de cada individuo, dando la posibilidad de configuración de la librería, ya que la función fitness es implementada por el usuario de la librería y solo tiene que pasársela a dicha librería para que la use.

También recibiremos el objeto de clase cromosomas creado antes, el valor de itmax, representa el número de iteraciones hasta que se termine la ejecución del algoritmo.

Child type es el tipo de reproducción que usará el algoritmo, a seleccionar entre child Split, child flip, etc.

Con selection type se especificará el método de selección con el que se generaran los siguientes hijos para rellenar la nueva generación, pudiendo seleccionarse el método de ruleta o el método de llenar la siguiente generación.

High low es una bandera para la ordenación del pool una vez calculado el fitness de todos los individuos.

Size pool es el tamaño del pool, en caso de no especificar nada tiene un valor por defecto de 1000.

Porcent elitism es el porcentaje de elitismo sobre dicho pool, siendo porcent mute la probabilidad de mutación de cada individuo una vez generado el siguiente hijo.

Can repeat chro es una bandera para indicar que los valores el valor de un cromosoma puede repetirse en el individuo.

Una vez tenemos el objeto de la clase atributo creado, se le asigna el atributo al algoritmo y este queda listo para ser ejecutado.

5 COMPARATIVA DE TIEMPOS

Se ha realizado una batería de pruebas con el mismo pool inicial, con los valores de elitismo y mutación fijados, con todo esto se han hecho diferentes ejecuciones con varias configuraciones, diferentes tamaños de individuos y diferentes métodos de selección y reproducción.

Obteniendo algunos de los siguientes resultados:

10 chro	child flip, fill next_gen , size 1000, elitism 10 mute 3		
iteraciones	time	resultado	Ejecución
1000	56.48	304	1
1000	56.48	304	2
1000	56.22	304	3
1000	56.53	304	4
1000	55.93	304	5
1000	56.06	304	6
1000	56.06	304	7
1000	56.64	304	8

1000	56.45	304	9
1000	56.19	304	10
	56.304	304	

Ahora podemos mirar el resultado en el caso de 20 cromosomas, y podremos ver como el incremento del tamaño del individuo no aumenta el tiempo significativamente.

	20 chro	child flip, fill next_gen , size 1000, elitism 10 mute 3	
Ejecución	Iteración	time	resultado
1	1000	57.47	243
2	1000	56.7	248
3	1000	57.53	247
4	1000	56.8	248
5	1000	57.29	239
6	1000	61.73	244
7	1000	56.72	243
8	1000	56.91	239
9	1000	59.34	239
10	1000	62.75	243
		58.324	243.3

Dándonos un incremento medio de 2s en la duración de las 1000 iteraciones, eso sí, cabe destacar que con tan pocas iteraciones el algoritmo NO llega a converger.

En cambio, la misma configuración, con 10 cromosomas y solo cambiando el método de selección a ruleta, obtenemos los siguientes resultados.

10 chro	child flip, roulette , size 1000, elitism 10 mute 3		
iteracion	time	resultado	Ejecucion
1000	156.01	304	1
1000	149.48	304	2
1000	153.94	304	3
1000	148.68	304	4
1000	150.95	304	5
1000	153.38	304	6
1000	157.45	304	7

1000	151.06	304	8
1000	158.53	304	9
1000	149.41	304	10
	152.889	304	

Podemos ver que con la misma configuración que en la primera tabla, solo modificando el método de selección a ruleta el tiempo de ejecución aumenta casi 100 segundos en cada 1000 iteraciones, debido a todos los cálculos realizados por ese método de selección.

Se han realizado ejecuciones con más de 1000 iteraciones máximas, para comprobar si los resultados con 15 y 20 cromosomas eran problema de convergencia o de que no daba tiempo a converger con tan pocas iteraciones, obteniendo la siguiente tabla de resultados

	1000	5000
10 Cromosomas	**304**	**304**
15 Cromosomas	**263**	**255**
20 Cromosomas	**243.3**	**236**

Se puede ver, que para un tamaño de 10 cromosomas converge por debajo de 1000, pero el resto llegan a converger entre 1000 y 5000 iteraciones, solucionándonos la duda de si era problema de que no convergiera o que faltaban iteraciones.

También se han hecho pruebas con los diferentes tipos de selección, combinados con los diferentes tipos de reproducción. Las tablas de los resultados pueden encontrarse en el siguiente enlace:
https://docs.google.com/spreadsheets/d/1aCaMWlvVRPWewUprRELVQxb2Y7rCV0sROCTPpQxHXhk/edit?usp=sharing

De todos los resultados también se puede obtener el dato de que si se usa una selección por torneo determinista es necesario incluir un poco de elitismo, o confiar en que si se produce una mutación en el primer individuo, otro ocupará su lugar sin importar que la solución no mejore.

6 CONCLUSIONES

Tras la realización de la práctica se ha publicado el código en Github para el uso de cualquiera que lo necesite, y para que cualquiera que esté dispuesto contribuya al proyecto.

Se han intentado añadir diferentes técnicas de reproducción para intentar hacer más genérica la librería, aparte de leer la función fitness del usuario directamente. Una mejora a corto plazo que podría realizarse es implementar diferentes métodos de mutación.