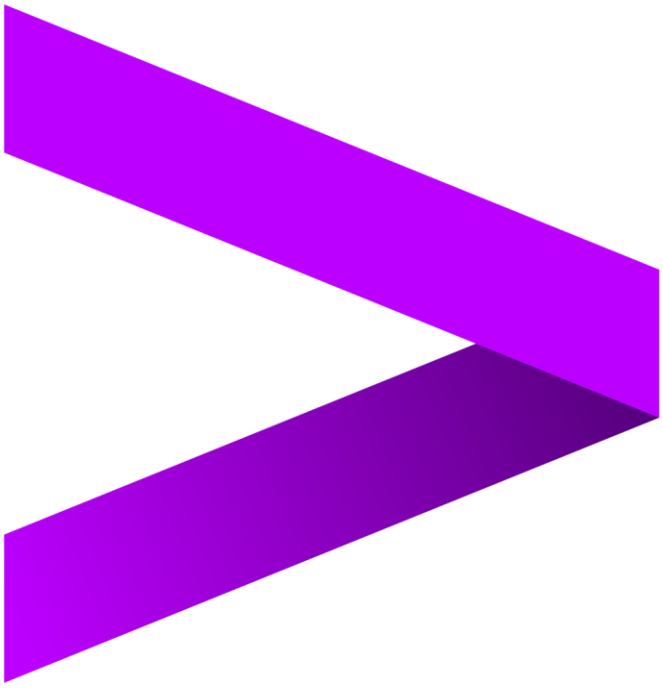
accenture

RETO ACCENTURE

Algoritmos de Optimización

Octubre 2019



1. EL RETO

El desarrollo de la Inteligencia Artificial para los juegos de mesa se viene produciendo con cada vez más naturalidad desde los últimos 20 años. En el año 1997, Gary Kasparov, entonces campeón mundial de ajedrez, fue derrotado por Deep Blue, programa de inteligencia artificial desarrollado por IBM. También fue el caso de AlphaGo, un algoritmo de DeepMind que en 2016 venció a Lee Sedol, campeón de go, un juego altamente estratégico que requiere de una capacidad de computación mucho mayor que el ajedrez. En 2017, un nuevo algoritmo de DeepMind, AlphaGo Zero, se convirtió en el mejor jugador de go del mundo al derrotar a AlphaGo. Pero, a diferencia de todos los campeones hasta la fecha, AlphaGo Zero alcanzó su maestría sin aprender de nadie. Partiendo solo de las reglas del juego, se entrenó jugando partidas contra sí mismo y en tan solo tres horas estuvo preparado jugar contra AlphaGo, al que venció por cien a cero en cien partidas. Más recientemente, en mayo de 2019, se ha establecido un nuevo hito histórico, de nuevo los investigadores de DeepMind han conseguido que sus agentes de IA sean capaces de jugar y ganar a los humanos en un juego pero, en este caso, Quake III Arena Capture the Flag, un videojuego multijugador en línea, algo imposible hasta la fecha, y que implica cooperar y competir con otros jugadores.

Bajo este contexto, se propone una versión del famoso Cifras y Letras en el que se pone la Inteligencia Artificial en general y los algoritmos de optimización, en particular, al servicio de este juego. ¿Serán capaces los participantes de desarrollar un algoritmo que sea mejor que ellos mismos? ¿Quién será el mejor de todos ellos?

La finalidad será obtener un número objetivo a partir de la combinación de los números primos, a excepción de uno que no se podrá utilizar, premiando aquellas soluciones que utilicen el menor número de factores primos posibles.

2. INTRODUCCIÓN A LOS ALGORITMOS DE OPTIMIZACIÓN

Un problema de optimización consiste en maximizar o minimizar una función real eligiendo sistemáticamente valores de entrada (tomados de un conjunto permitido) y computando el valor de la función. La generalización de la teoría de la optimización y técnicas para otras formulaciones comprende un área grande de las matemáticas aplicadas. Los puntos fundamentales que debe cumplir un proceso para ser considerado un verdadero algoritmo:

- Debe ser preciso: es por ello que se debe indicar el orden exacto de ejecución de cada paso implicado en el proceso.
- Debe estar perfectamente definido. Esto significa que en el caso de ejecutarse el mismo más de dos veces, siempre se debe obtener el mismo resultado independientemente de la cantidad de veces que se siga.
- Debe ser finito, es decir que el algoritmo debe culminar en algún momento de su ejecución, expresado en otras palabras, debe tener un número de pasos bien determinados hasta concluir con su tarea.
- Debe ser legible. Esto significa que el texto que describe debe ser claro y conciso, de una manera tal que permita su comprensión inmediata, es decir sin procedimientos rebuscados o poco claros.
- Debe estar definido en tres partes fundamentales, las cuales son: Entrada, Proceso y Salida. Si quieres saber más sobre este tema, más adelante en este mismo post encontrarás información al respecto.

Ante un problema de optimización, la primera pregunta que se debe responder es si es fácil o difícil de resolver. Aunque parece una pregunta simple, sólo a partir de la década de los setenta los investigadores abordaron este tema, introduciendo un nuevo campo de investigación: la complejidad computacional, la cual, entre otros usos, determina si un problema es fácil o no de acuerdo con los algoritmos conocidos para resolverlo. Dentro de los algoritmos de optimización cuantitativos podemos encontrar dos tipos fundamentales:

- La programación matemática, que es la rama de la investigación operativa que trata de identificar un punto extremo (máximo o mínimo) de una función f dentro de un conjunto S. La podemos dividir en dos grandes campos programación lineal y programación no linear dependiendo si la función f

es lineal y el conjunto S se puede describir como un conjunto de desigualdades lineales o, por el contrario, esto no es posible. Algunos de los algoritmos más apropiados para resolver el tipo de problemas que en este reto se plantean son:

  - La ramificación y poda interpreta el problema como un árbol de soluciones, donde cada rama nos lleva a una posible solución posterior a la actual. La característica de esta técnica con respecto a otras anteriores (y a la que debe su nombre) es que el algoritmo se encarga de detectar en qué ramificación las soluciones dadas ya no están siendo óptimas, para «podar» esa rama del árbol y no continuar malgastando recursos y procesos en casos que se alejan de la solución óptima.
  - La programación dinámica es un método para reducir el tiempo de ejecución de un algoritmo mediante la utilización de subproblemas superpuestos y subestructuras óptimas, como se describe a continuación. Contemplar un problema como una secuencia de decisiones equivale a dividirlo en problemas más pequeños y por lo tanto más fáciles de resolver como hacemos en Divide y Vencerás, técnica similar a la de programación dinámica.

- Por otro lado, los algoritmos metaheurísticos son métodos aproximados diseñados para resolver problemas de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Los metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos, combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos.

3. MECÁNICA DEL PROBLEMA

Tal y como se ha comentado, el objetivo es la combinación de los números primos mediante sumas, restas, divisiones y multiplicaciones para conseguir un resultado determinado. Se podrá utilizar varias veces el mismo número primo, pero no sé podrá utilizar un número primo en concreto. Tanto el resultado como el número primo a excluir será fijado por el jurado en la fase de evaluación.

Los números primos a utilizar podrán ser todos aquellos de una o dos cifras, incluido el uno.

El número objetivo será un número natural de 0 a 10^9.

El script únicamente podrá tener precalculados los números primos de una o dos cifras. No pudiendo contener ninguna otra operación matemática precalculada.

El script desarrollado por los participantes será probado 100 veces por el jurado con resultados objetivo y números primos a excluir diferentes. De este modo, se pretende conseguir una muestra significativa de cada algoritmo que permita identificar aquellos que son más eficientes.

El jurado evaluará el grado de optimización del algoritmo, así como la documentación presentada. Ambos entregables, algoritmo y documentación, tendrán que atenerse a las consideraciones y requisitos expuestos en los epígrafes siguientes.

3.1 Input del script

La evaluación del algoritmo se realizará a través de un script denominado TEST.txt, que contendrá una cabecera y estará delimitado a través de un pipe o |. Los campos que lo formen serán tres: el ID de la prueba, el resultado objetivo y el número primo que no se puede utilizar en dicha prueba. Por ejemplo:

ID|TARGET|PRIMO

1|345|3

2|568|7

…

100|289|2

El fichero de prueba se alojará en el mismo directorio que el script y será el mismo para todos los concursantes.

3.2 Ejecución y características del script

El algoritmo deberá ser desarrollado en lenguaje de programación Python (versión 3.7), con extensión ".py" y cuyo nombre será SCRIPT. Este script SCRIPT.py deberá ser autoinstalable en caso de disponer de alguna dependencia, pues deberá contener todo lo necesario para que el programa funcione en cualquier ordenador con conexión a internet. Las características del ordenador son las siguientes:

- OS Name Microsoft Windows 10 Enterprise
- System: Dell Inc. Precision 3520. x64-based PC
- Processor Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz, 2901 Mhz, 4 Core(s), 8 Logical Processor(s)
- Installed Physical Memory (RAM) 16,0 GB

El script no podrá contener procesamiento multihilo y, como ya se ha comentado, únicamente podrá tener precalculados los números primos potencialmente utilizables.

Tal y como ya se ha comentado, el algoritmo deberá leer las pruebas a realizar del fichero TEST.txt que estará almacenado en el mismo directorio que el SCRIPT.py. A partir de la combinación mediante sumas, restas, multiplicaciones y divisiones, deberá obtener el número objetivo sin utilizar el número primo a excluir en cada caso. Destacar que se identificarán como exitosos o ganadores aquellos algoritmos que en una determinada prueba consigan el resultado objetivo con la menor utilización de números. Por ejemplo, para el número objetivo 6 y con el número primo a excluir 3:

- Participante A: 2x2+2 = 6
- Participante B: 5+1 = 6
- Participante C: 7-1 = 6
- Participante D: 3x2 = 6

Tanto el participante B como el C serían ganadores para esta prueba por utilizar el menor número de números primos. El participante A no sería ganador por utilizar más elementos que B y C. Por último, el participante D no sería ganador por utilizar el número primo a excluir (3).

El tiempo total para las 100 pruebas será de 240 segundos, no contabilizándose los resultados de las pruebas que se hayan obtenido tras ese tiempo.

3.3 Salida del script

La ejecución del SCRIPT.py deberá generar en su misma carpeta un fichero SCORE.txt, delimitado por un pipe o |, sin cabecera, y que contendrá los siguientes campos:

- En la primera fila: un timestamp en formato YYYY-MM-DD-HH:MM:SS.SSSS con el momento de arranque del script.
- En las 100 filas siguientes:
  - Nombre del equipo utilizando únicamente caracteres alfanuméricos
  - ID de la prueba a la que pertenece el registro
  - Número de números primos utilizados. En el caso de utilizar un mismo número varias veces, computará tantas veces como haya sido utilizado.
  - Fórmula matemática obtenida para la obtención del target, pudiendo utilizar los símbolos: + - / * ( )
- En la última fila: un timestamp en formato YYYY-MM-DD-HH:MM:SS.SSSS con el momento de finalización del script.

Un ejemplo de fichero SCORE.txt, sería:

- 2019-10-15-17:38:23.1234
- EquipoA|1|5|3*2*(7+5)-3
- ...
- EquipoA|100|3|2*3*+1
- 2019-10-15-17:39:32.1356

La fórmula será utilizada para validar que el resultado obtenido a través del número indicado de números primos es correcto.

3.4 Entregables

Cada participante deberá hacer entrega del fichero SCRIPT.py y de un documento explicativo de lo realizado. Esta memoria, podrá ser una presentación o documento de texto. Deberá contener, al menos:

- Metodología y estrategia desarrollada para la resolución del problema
- Principales dificultades encontradas
- Conocimientos adquiridos
- Fuentes principales de consulta (webs de consulta, libros…)

3.5 Criterios de valoración

La valoración subjetiva se realizará a través de los siguientes elementos:

- Puntos obtenidos en cada una de las 100 pruebas en las que el participante consiga obtener la cifra target usando el menor número de números primos con respecto al resto de participantes. Si hubiera varios participantes que utilizasen el mismo número de elementos, se les daría un punto a todos ellos.
- Tiempo de ejecución empleado para resolver el ejercicio propuesto.
- Memoria sobre el trabajo realizado en la que se valorará la claridad, la calidad, la innovación, la originalidad y la metodología propuesta entre otros.

El jurado se reserva el derecho a desestimar las soluciones presentadas de aquellos participantes que no cumplan los requisitos exigidos en este documento.