Ingeniería de la Salud
Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

Bioinformática

Práctica 3. Aproximación Diofántica

Arrabalí Cañete, Carmen Lucía

---

1. Introducción y objetivos

En la teoría de números, el estudio de la aproximación diofántica se encarga de aproximar números reales por medio de números racionales.

El primer problema consistía en saber hasta qué punto un número real puede ser aproximado por números racionales. Para este problema, un número racional $\frac{p}{q}$ es una "buena" aproximación de un número real α si el valor absoluto de la diferencia entre $\frac{p}{q}$ y α no puede disminuir si $\frac{p}{q}$ se sustituye por otro número racional con un denominador menor. Este problema se resolvió en el siglo XVIII mediante las fracciones continuas.

1.1. Enunciado matemático del método

Sea $x \in \mathbb{R}$ un número real y $\epsilon \in \mathbb{R}^+$ una pequeña constante positiva, se desea encontrar p y $q \in \mathbb{Z}$ tal que $\frac{p}{q}$ $\left|x - \frac{p}{q}\right| < \epsilon$ y, además, p y q son números relativamente primos, por lo que $\frac{p}{q}$ está en términos mínimos.

Al haber infinitos números racionales $\frac{p}{q}$ que se pueden aproximar a x dentro de una tolerancia ϵ, se dice que la mejor aproximación diofántica $\frac{p}{q}$ es $\left|x - \frac{p}{q}\right| < \left|x - \frac{p'}{q'}\right|$ teniendo en cuenta que para cada número racional $\frac{p'}{q'}$ diferente de $\frac{p}{q}$ debe cumplir que $0 < q' \leqslant q$.

2. Configuración del equipo

Se ha realizado la implementación en un equipo con un sistema operativo Windows 10 Home 64 bits, con un procesador Intel(R) Core (TM) i7-6500U CPU @ 2.50GHz 2.59 GHz (4 CPUs) y un disco SSD de 480GB y 12GB de RAM. La versión Java corresponde a la número 17.

3. Implemetación

En el primer método de la implementación, *protected RationalNumber getMiddlePoint* de la clase *MediantApproximation* (ver apartado 3.1) se encarga de calcular el punto medio entre dos números racionales L y R. En este caso, se calcula como la suma de los numeradores de ambos números y la suma de los denominadores de los mismos.

$$\text{Dado que L} = \frac{a}{c} \text{ y R} = \frac{b}{d}\text{, la mediana de ambos corresponde a M} = \frac{a+b}{c+d}$$

Por otro lado, se tiene el método *protected RationalNumber _approximate* de la clase *BinarySearchDiophantineApproximator* (ver apartado 3.2) que se encarga de realizar una búsqueda binaria entre dos números racionales L y R.

3.1. Método *protected RationalNumber getMiddlePoint* de la clase *MediantApproximation*

```
protected RationalNumber getMiddlePoint(RationalNumber l, RationalNumber r) {
  RationalNumber M = new  RationalNumber(l.numerator() + r.numerator(), l.denominator() +
  r.denominator());
  return M;
}
```

3.2. Método *protected RationalNumber _approximate* de la clase *BinarySearchDiophantineApproximator*

```
protected RationalNumber _approximate(double x, double epsilon) {
  RationalNumber L = zero, R = one; // rational numbers that enclose x
  RationalNumber M;          // the middle-ish point (a rational number between L and R).
  double Mfp;              // floating-point value of the middle-ish point
  double diff;             // difference between the middle-ish point and |x|

  if (x < epsilon) {
    M = zero;
  }else if (x > (1-epsilon)) {
    M = one;
  }else {
    // Perform binary search between L and R. Use
    // method getMiddlePoint to obtain a rational
    // number in-between L and R.
    do {
      M = getMiddlePoint(L, R);
      Mfp = M.asDouble();
      diff = Math.abs(Mfp - x);
      if (M.asDouble() <= x) {
        L = M;
      } else {
        R = M;
      }
    } while (diff >= epsilon);

  }
  return M;
}

```

4. Resultados y Conclusiones

Para poder analizar correctamente los resultados, el método main recibe en los argumentos de entrada, varios modos de ejecución entre los que se encuentra $-r$ <*max_digits*> <*max_num*> el cual genera un archivo con <max_num> números aleatorios con una longitud <max_digits> y calcula las aproximaciones con distintos valores de tolerancia, desde 0.1 hasta 10^{-16}.

Al utilizar ese modo de ejecución, se genera un archivo con los diferentes tiempos de ejecución, el cual se llama un archivo *Mediant-Approximation-stats.txt*.

Por otro lado, entre los recursos aportados en el Campus Virtual se tiene un archivo llamado *analyze.R* que contiene un script el cual se encarga de generar una gráfica haciendo uso del archivo .txt generado anteriormente con el modo de ejecución −r.

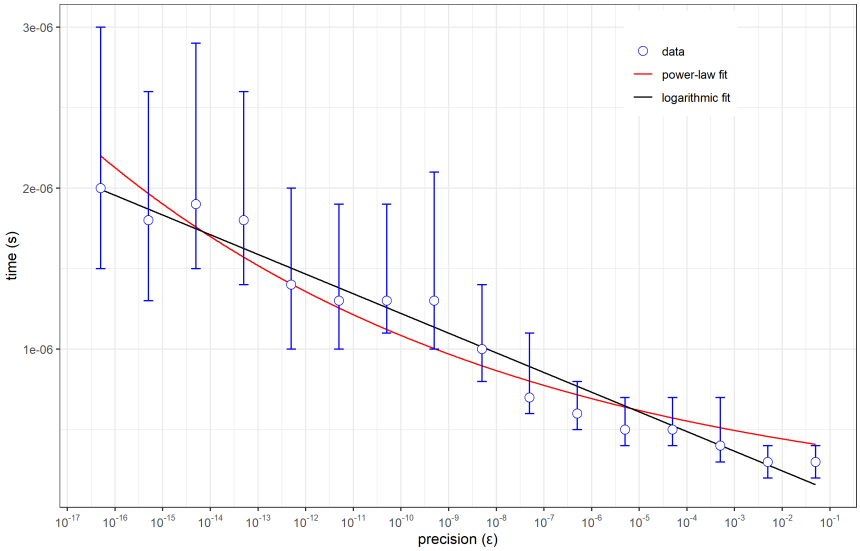

Figura 1: Resultado de la ejecución del algoritmo con los parámetros utilizados.

Para poder obtener un buen conjunto de datos se han utilizado 50000 números con una longitud máxima de 16 caracteres. Como se puede observar en la figura 1, cuanta mayor sea la precisión con la que se quiere obtener la aproximación diofántica, el tiempo que tarda el algoritmo en obtener el resultado también es mayor. Se ajusta a la definición de la fórmula inicial, ϵ cuanto más pequeño es, menos diferencia tiene que haber con el número original.