Práctica 2. "Algoritmos probabilisticos y aleatorios"

Miriam Pescador-Rojas[1] and Carlos A. Coello Coello

[1] ESCOM, Instituto Politécnico Nacional, Ciudad de México 07738, México
[2] CINVESTAV-IPN (Evolutionary Computation Group) Computer Science Department, Ciudad de México 07360, México

Abstract. multi-objective evolutionary algorithms (MOEAs)

Key words: algoritmos aleatorios, método de monte Carlo, algoritmos de las Vegas

1 Introducción

Un algoritmo se dice aleatorizado si usa algún grado de aleatoriedad como parte de su lógica.

En este tipo de algoritmos el tiempo de ejecución o su salida se convierten en variables aleatorias.

Con base en el tiempo de ejecución y la salida de los algoritmos aleatorizados como variables aleatorias, podemos definir dos tipos de algoritmos:

- *Las Vegas:* algoritmos que siempre entregan el resultado correcto, pero cuyo tiempo de ejecución varía (incluso para la misma entrada de datos). Un algoritmo Las Vegas se dice eficiente si su tiempo esperado de ejecución es polinomial para cualquier entrada.
- *Monte Carlo:* algoritmos que pueden entregar resultados incorrectos con una probabilidad acotada. Podemos disminuir esta probabilidad de error repitiendo el algoritmo a expensas del tiempo de ejecución. Un algoritmo Monte Carlo se dice eficiente si su tiempo de ejecución en el peor caso es polinomial para cualquier entrada.

1.1 Método de Monte Carlo aplicado a búsquedas

Los Métodos de Monte Carlo se basan en la analogía entre probabilidad y volumen. Las matemáticas de las medidas formalizan la noción intuitiva de probabilidad, asociando un evento con un conjunto de resultados y definiendo que la probabilidad del evento será el volumen o medida relativa del universo de posibles resultados. Monte-Carlo usa esta identidad a la inversa, calculando el volumen de un conjunto interpretando el volumen como una probabilidad. En el caso más simple, esto significa muestrear aleatoriamente un universo de resultados posibles y tomar la fracción de muestras aleatorias que caen en un conjunto

dado como una estimación del volumen del conjunto. La ley de grandes números asegura que esta estimación converja al valor correcto a medida que aumenta el número de muestras. El teorema del límite central proporciona información sobre la magnitud del probable error en la estimación después de un número finito de muestras.

En un proceso estándar de Monte Carlo tenemos los siguientes pasos:

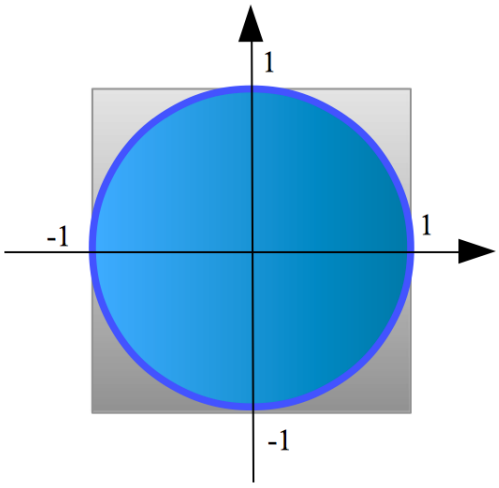
- Se generan un gran número de simulaciones aleatorias desde la posición del tablero para la que se desea encontrar el mejor movimiento siguiente.
- Se almacenan las estadísticas para cada movimiento posible a partir de este estado inicial.
- Se devuelve el movimiento con los mejores resultados generales.

Un ejemplo común de la aplicación de método Monte Carlo es la aproximación del valor de π, para ello tenemos que considerar un círculo unitario (radio=1) dentro de un cuadrado con los lados iguales a 2 (véase la figura 1). Sí escogemos un punto al azar (x, y) donde x y y están definidos en el rango $[-1, 1]$, la probabilidad de que este punto al azar se encuentre dentro del círculo unitario se da como la proporción entre el área el círculo unitario y el cuadrado:

$$P(x^2, y^2 \leq 1) = \frac{A_{circle}}{A_{square}} = \frac{\pi}{4} \tag{1}$$

Si escogemos puntos al azar N veces y M de esas veces el punto cae dentro del círculo unitario, la probabilidad de que un punto al azar caiga dentro de dicho círculo esta dado por:

$$P'(x^2, y^2 \leq 1) = \frac{M}{N} \tag{2}$$



Por lo tanto consideramos la siguiente ecuación para el calculo de π:

$$H(x,y) = \begin{cases} 1 \text{ si } x^2 + y^2 \leq 1 \\ 0 \text{ in other case} \end{cases} \tag{3}$$

1.2 Desarrollo (sección 1. Monte Carlo)

– Pruebe el código en python proporcionado y ejecute pruebas para $N = 100, 1000, 10000, 100000, 1000000$, muestre cuáles son los valores que obtiene para π en cada caso.

N	Valores	Error
100	3.28	4.22%
1000	3.152	0.33%
10000	3.1348	0.22%
100000	3.1414	0.01%
1000000	3.1433	0.05%

Fig. 1. se muestran los resultados de la experimentacin

– Implemente el método de Monte Carlo para resolver integrales. Utilice la siguiente formula:

$$\frac{b-a}{n} \sum_{i=1}^{n} f(u_i(b-a)+a) \tag{4}$$

donde a y b son los límites inferior y superior de una integral, u_i es un valor aleatorio generado entre $[0, 1]$ y f se refiere a la función a integrar.
Considere las siguientes integrales para probar su implementación:

$$f_1(x) = \int_0^1 (1-x^2)^{3/2} \cdot dx \tag{5}$$

$$f_2(x) = \int_{-1}^1 e^{x+x^2} \cdot dx \tag{6}$$

$$f_3(x) = \int_0^2 (1+x^2)^2 \cdot dx \tag{7}$$

$$f_4(x) = \int_0^{2\pi} \frac{1}{cos(x)+2} \cdot dx \tag{8}$$

$$f_5(x) = \int_0^2 log(x) \cdot dx \tag{9}$$

Deberá desarrollar una simulación donde se muestre paso a paso la generación de números aleatorios y su graficación en la función correspondiente. (Ver ejemplo en figura 1.2 y consulte la página web https://www.scratchapixel.com/lessons/mathematics-physics-for-computer-graphics/monte-carlo-methods-in-practice/monte-carlo-integration).

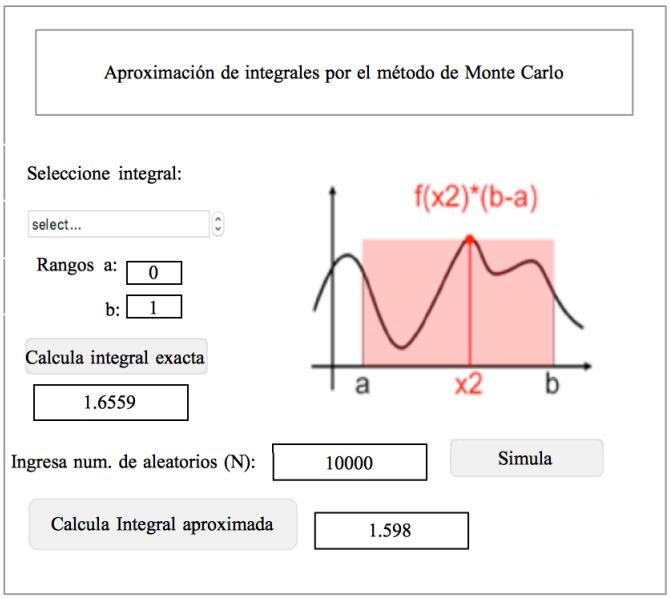


Fig. 2. Ejemplo de interfaz gráfica para simular aproximación a integrales

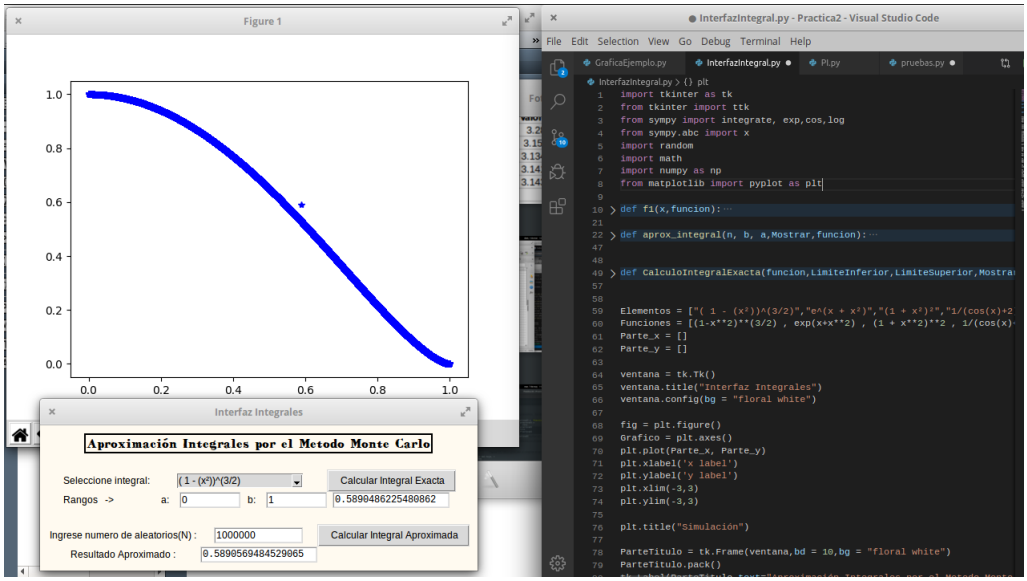


Fig. 3. Interfaz gráfica para simular aproximación a integrales que se realizó

1.3 Desarrollo (sección 2. Las Vegas)

Para poner en práctica el algoritmo las vegas se implementaran dos ejercicios:

1. Algoritmo de Quick-sort. Modifique el algoritmo de Quick sort implementado en la práctica 1. Por cada llamada recursiva seleccione al azar el pivote, compare si hay mejora respecto a su versión anterior para el caso promedio y reporte el tiempo de ejecución para los siguientes casos $n = 1000, 2000, 3000, \ldots, 10,000$ (n es el número de elementos en un arreglo).
2. El prolema de las 8 reinas. Se requiere generar una solución correcta al problema de las 8 reinas de acuerdo a las siguientes reglas.

El problema de las 8 reinas consiste en colocar las piezas en un tablero de ajedrez sin que se amenacen entre sí. En el juego de ajedrez la reina amenaza a aquellas piezas que se encuentren en su misma fila, columna o diagonal.

Para resolver este problema emplearemos un esquema de algoritmo Las vegas. Se supondrá que hay un cierto número de piezas colocadas en forma correcta (sin atacarse) y se generará el resto de las piezas al azar. Considera las siguientes configuraciones predefinidas e implemente un algoritmo que genere las otras reinas de manera aleatoria.

Reporte en una tabla el porcentaje de éxitos del algoritmo de acuerdo a los casos presentados en la tabla 1.3

La práctica puede realizarse en equipos de 2 personas. La entrega de la práctica es por correo electrónico para el próximo viernes 1 de marzo (11:59 pm). Deberá enviar el código fuente y un reporte en formato latex (2).

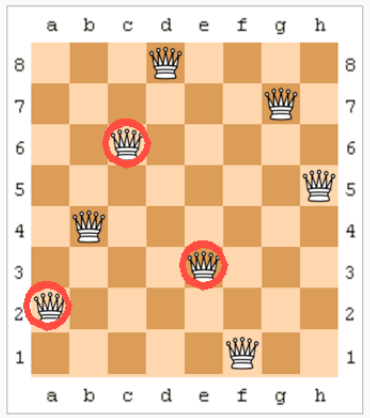


Fig. 4. Tablero con 3 reinas preestablecidas.

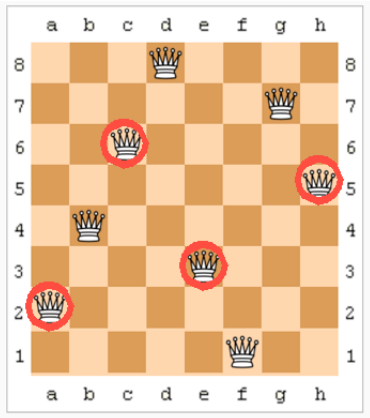


Fig. 5. Tablero con 4 reinas preestablecidas.

Table 1. Casos para el problema de las 8 reinas

	Reinas aleatorias			
Soluciones aleatorias	8	5	4	3
1,000				
10,000				
100,000				
1,000,000				

[2] [1] [3] [4]

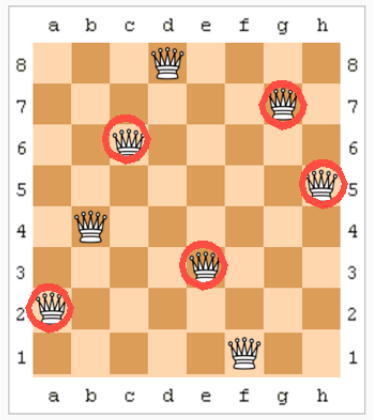


Fig. 6. Tablero con 5 reinas preestablecidas.

References

1. Deb K., M. K. A Review of Nadir Point Estimation Procedures Using Evolutionary Approaches: A Tale of Dimensionality Reduction. Tech. Rep. KanGAL Report 2008004, Indian Institute of Technology, 2008.
2. Marler, R., and Arora, J. Survey of Multi-objective Optimization Methods for Engineering. *Structural and Multidisciplinary Optimization 26*, 6 (2004), 369–395.
3. Miettinen, K. *Nonlinear Multiobjective Optimization*, vol. 12. International Series in Operations Research & Management Science, 1998.
4. Zhang, Q., and Li, H. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation 11*, 6 (December 2007), 712–731.