CIENTIFICA
TechNoLogica
ISSN:3119-7906, Vol.1Núm.2(2026): Abril-Junio
https://doi.org/10.17268/tech1.2.2026.03

Métodos Algorítmicos: Fuerza Bruta Como Estrategia de Solución

Algorithmic Methods: Brute Force as a Solution Strategy

Becerra Gil Carlos Bruno[1], Cotrina Lezama James Larry[1], De la Cruz Lara Alva Maythe[1], Quiroz Herrera Yubitza Clarita, Rivera Chirinos James Kevin Josimar[1]

[1] Escuela de Ingeniería Informática, Universidad Nacional de Trujillo, La Libertad, Perú

*Autor correspondiente: cbbecerrag@unitru.edu.pe (Bruno. B)

RESUMEN

Este artículo busca entender cómo funcionan los algoritmos de fuerza bruta para resolver problemas en computadoras. Se trata de explicar qué son, cómo funcionan, qué ventajas y limitaciones tienen, y cómo se usan en algunos problemas conocidos. Estos problemas incluyen encontrar divisores de un número, el problema del agente viajero y buscar patrones en textos. La forma en que se hace esto es poniendo en práctica estos algoritmos y analizando cómo afectan el tiempo y el espacio que necesita la computadora. Los resultados muestran que, aunque estos algoritmos siempre encuentran la mejor solución probando todas las posibilidades, no son muy eficientes cuando el problema es muy grande. En resumen, los algoritmos de fuerza bruta son útiles cuando el problema no es demasiado grande o cuando no hay métodos mejores. Sin embargo, no son prácticos para problemas muy complejos.
Palabras clave: Algoritmos de fuerza bruta, complejidad computacional, búsqueda exhaustiva, problema del agente viajero, búsqueda de patrones.

ABSTRACT

This article aims to understand how brute-force algorithms work to solve problems on computers. It explains what they are, how they function, their advantages and limitations, and how they are used in some well-known problems. These problems include finding divisors of a number, the traveling salesman problem, and searching for patterns in texts. This is done by implementing these algorithms and analyzing their impact on the computer's time and resource requirements. The results show that while these algorithms always find the best solution by testing all possibilities, they are not very efficient when the problem is very large. In short, brute-force algorithms are useful when the problem is not too large or when there are no better methods. However, they are not practical for very complex problems.
Keywords: *Brute-force algorithms, computational complexity, exhaustive search, traveling salesman problem, pattern search.*

I. INTRODUCCIÓN

En informática y ingeniería de software, diseñar y analizar algoritmos es fundamental para crear soluciones computacionales eficientes. Una de las estrategias más directas para resolver problemas es el algoritmo de fuerza bruta. Este enfoque se caracteriza por su simplicidad y por garantizar la solución óptima al explorar todas las posibilidades.

Los algoritmos de fuerza bruta, también llamados exhaustivos, consisten en probar todas las soluciones posibles para encontrar la que cumpla con ciertos criterios. Aunque es sencillo, este método tiene implicaciones importantes en eficiencia, especialmente con problemas grandes.

La importancia de estos algoritmos radica en que se pueden aplicar en cualquier situación y sirven como punto de partida para soluciones más complejas. Son útiles cuando el espacio de búsqueda es pequeño, no hay un método mejor disponible o se necesita una solución de referencia.

Este artículo se divide en cinco secciones. La primera es esta introducción. La sección 2 describe los materiales y métodos usados, incluyendo los problemas estudiados y los algoritmos implementados. La sección 3 presenta los resultados obtenidos. La sección 4 analiza estos resultados en relación con otros estudios. La sección 5 resume las conclusiones y propone trabajos futuros.

II. MATERIALES Y METODOS

A. Definición y Características Fundamentales

Un algoritmo de fuerza bruta es una técnica de resolución de problemas que consiste en probar sistemáticamente todas las posibles soluciones para encontrar la óptima. Esta estrategia se asemeja a buscar una aguja en un pajar revisando cada brizna de paja una por una, garantizando que, si la aguja existe, será encontrada. Las características principales de los algoritmos de fuerza bruta son:

- Exhaustividad: Exploran todo el espacio de búsqueda sin omitir ninguna solución posible.
- Simplicidad: Son fáciles de entender y de implementar por su naturaleza intuitiva.
- Optimalidad garantizada: Siempre encuentran la solución óptima si esta existe dentro del espacio de búsqueda.
- Ineficiencia potencial: Pueden consumir mucho tiempo y recursos computacionales en problemas de gran tamaño. Además, hemos utilizado https://graphonline.top/es/ para realizar los grafos.

B. Problemas Seleccionados

Para analizar los algoritmos de fuerza bruta, se han elegido tres problemas clásicos que muestran diferentes partes de esta estrategia.

1. Búsqueda de divisores de un número natural: este problema trata de encontrar todos los divisores de un número entero positivo. La forma de fuerza bruta consiste en probar todos los números desde 1 hasta n para ver si dividen exactamente a n.
2. Problema del agente viajero: el problema del agente viajero consiste en encontrar el recorrido más corto que visita un conjunto de ciudades solo una vez y regresa a la ciudad de inicio. Para n ciudades, hay (n-1)1/2 recorridos posibles.
3. Búsqueda de patrones en texto: este problema implica encontrar todas las veces que aparece un patrón dentro de un texto. La forma de fuerza bruta compara el patrón con cada posición posible en el texto.

C. Metodología de Análisis

Para cada problema, se creó un algoritmo básico y se estudió su complejidad temporal. También se probaron los algoritmos con diferentes tamaños de datos para ver cómo crecía el tiempo de ejecución.

III. RESULTADOS

A. Búsqueda de Divisores

El método simple para encontrar los divisores de un número es probar desde 1 hasta el número mismo. Para ver si un número es divisor, se hace la siguiente comprobación: si el resto de dividir el número entre el posible divisor es cero, entonces es un divisor. En la tabla I se muestra un ejemplo con el número 3.

TABLA I

COMPROBACIÓN DE DIVISORES

Número de Corrida	i	3 % == 0	Resultado
1	1	Verdadero	1 es divisor de 3
2	2	Falso	No es divisor
3	3	Verdadero	3 es divisor de 3

B. Problema del Agente Viajero

Para el caso de 5 ciudades {A, B, C, D, E}, con la matriz de distancias mostrada en la tabla II, el número de ciclos hamiltonianos posibles es (5-1)!/2 = 12. Esto significa que existen 12 rutas distintas que visitan cada ciudad exactamente una vez y regresan al punto de partida, considerando que las rutas en sentido contrario se cuentan como iguales.

TABLA I

MATRIZ DE DISTANCIAS

	A	B	C	D	E
A	0	57	64	85	26
B	57	0	88	54	34
C	64	88	0	57	56
D	85	54	57	0	23
E	26	34	56	23	0

El análisis de los 12 posibles ciclos se presenta en la tabla III, donde se muestra la distancia total de cada recorrido.

TABLA II

DISTANCIAS TOTALES PARA CADA CICLO HAMILTONIANO

Ciclo	Recorrido	Distancia Total
1	A-B-C-D-E-A	251
2	A-B-C-E-D-A	232
3	A-B-D-C-E-A	250
4	A-B-D-E-C-A	254
5	A-B-E-D-C-A	235
6	A-C-B-D-E-A	255
7	A-C-B-E-D-A	217
8	A-C-D-B-E-A	235
9	A-D-C-B-E-A	213
10	A-D-C-E-B-A	212
11	A-D-B-E-C-A	216
12	A-D-B-B-E-A*	232

La distancia mínima encontrada es 212, correspondiente al ciclo A-D-C-E-B-A.

C. Búsqueda de Patrones en Texto

El algoritmo de fuerza bruta busca patrones en un texto comparando el patrón con cada parte del texto. Por ejemplo, si tenemos el texto “THISISATEST” y buscamos el patrón “TEST", el algoritmo hace comparaciones una tras otra hasta encontrar una coincidencia.

La complejidad temporal de este algoritmo es O(n*m). Aquí, n es la longitud del texto y m es la longitud del patrón.

IV. DISCUSIÓN

Los resultados que hemos obtenido confirman las características teóricas de los algoritmos de fuerza bruta. La búsqueda de divisores, con una complejidad de O(n), muestra que, para problemas con un espacio de búsqueda lineal, el enfoque de fuerza bruta es perfectamente viable. Sin embargo, el problema del agente viajero muestra claramente la principal limitación de estos algoritmos: su complejidad factorial O(n!).

Como se puede ver en la tabla III, para solo 5 ciudades, el número de ciclos a evaluar es 12. Para 10 ciudades, serían 181.440 ciclos, y para 20 ciudades, la cifra asciende a aproximadamente 1,2 × 10^17 ciclos. Este crecimiento exponencial hace que el algoritmo de fuerza bruta sea impracticable para problemas de tamaño moderado o grande.

En el caso de la búsqueda de patrones, la complejidad O(n*m) puede ser aceptable para textos cortos, pero se vuelve ineficiente cuando tanto el texto como el patrón son largos. El principal problema es que el algoritmo de fuerza bruta necesita retroceder en cada discrepancia, realizando muchas comparaciones redundantes.

A pesar de estas limitaciones, los algoritmos de fuerza bruta tienen aplicaciones prácticas en diversos dominios, incluyendo la criptografía, la optimización de rutas y la verificación de sistemas. La principal ventaja de los algoritmos de fuerza bruta radica en su simplicidad y en la garantía de encontrar la solución óptima.

V. CONCLUSIÓN

Los algoritmos de fuerza bruta son una estrategia básica en soluciones computacionales, caracterizados por su simplicidad y la garantía de encontrar la mejor solución mediante la exploración completa del espacio de búsqueda.

El análisis realizado muestra que estos algoritmos son adecuados para problemas con espacios de búsqueda pequeños o cuando no existen métodos más eficientes. Por ejemplo, la búsqueda de divisores es viable con esta estrategia. Sin embargo, en problemas complejos como el del agente viajero, su uso se vuelve impráctico para conjuntos de datos grandes debido a su alta complejidad.

Para trabajos futuros, se propone investigar formas de optimizar los algoritmos de fuerza bruta, incluyendo técnicas para reducir el espacio de búsqueda. Además, se recomienda comparar estos algoritmos con otros enfoques, como la programación dinámica, algoritmos voraces y metaheurísticas, para evaluar su desempeño y aplicabilidad.

VI. RECONOCIMIENTOS

Los autores agradecen a la Universidad Nacional de Trujillo por el apoyo institucional brindado para la realización de esta investigación. Asimismo, expresan su agradecimiento a los revisores anónimos por sus valiosos comentarios y sugerencias, los cuales contribuyeron a mejorar la calidad del presente artículo.

VI. REFERENCIAS BIBLIOGRÁFICAS

[1]. T. H. Cormen, C. E. Leiserson, R. L. Rivest y C. Stein, Introduction to Algorithms, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.

[2]. R. C. T. Lee, S. S. Tseng, R. C. Chang y Y. T. Tsai, Introducción al diseño y análisis de algoritmos: un enfoque estratégico. México: McGraw-Hill Interamericana, 2007.

[3]. R. Guerrero y A. Vallecillo, Técnicas de diseño de algoritmos. Málaga, España: Universidad de Málaga, 1999.

[4]. J. Kleinberg y E. Tardos, Algorithm Design. Boston, MA, USA: Addison-Wesley, 2006.

[5]. S. Dasgupta, C. Papadimitriou y U. Vazirani, Algorithms. New York, NY, USA: McGraw-Hill, 2008.

[6]. [M. T. Goodrich y R. Tamassia, Algorithm Design and Applications. Hoboken, NJ, USA: Wiley, 2015.

[7]. A. Levitin, Introduction to the Design and Analysis of Algorithms, 3rd ed. Boston, MA, USA: Pearson, 2012.

[8]. R. Sedgewick y K. Wayne, Algorithms, 4th ed. Boston, MA, USA: Addison-Wesley, 2011.

[9]. D. E. Knuth, The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd ed. Boston, MA, USA: Addison-Wesley, 1997.

[10]. E. Horowitz, S. Sahni y S. Rajasekaran, Computer Algorithms. Rockville, MD, USA: Computer Science Press, 1998.

[11]. G. Brassard y P. Bratley, Fundamentals of Algorithmics. Upper Saddle River, NJ, USA: Prentice Hall, 1996.

[12]. J. Hromkovic, Algorithmics for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics, 2nd ed. Berlin, Germany: Springer, 2004.

[13]. V. V. Vazirani, Approximation Algorithms. Berlin, Germany: Springer, 2001.

[14]. M. R. Garey y D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. New York, NY, USA: W. H. Freeman, 1979.

[15]. C. H. Papadimitriou y K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity. Mineola, NY, USA: Dover Publications, 1998.

[16]. B. Korte y J. Vygen, Combinatorial Optimization: Theory and Algorithms, 4th ed. Berlin, Germany: Springer, 2008.

[17]. D. P. Bertsekas, Dynamic Programming and Optimal Control, Vol. 1, 4th ed. Belmont, MA, USA: Athena Scientific, 2017.

[18]. Z. Michalewicz y D. B. Fogel, How to Solve It: Modern Heuristics, 2nd ed. Berlin, Germany: Springer, 2004.

[19]. S. Luke, Essentials of Metaheuristics, 2nd ed. Fairfax, VA, USA: Lulu Press, 2013.

[20]. K. A. De Jong, Evolutionary Computation: A Unified Approach. Cambridge, MA, USA: MIT Press, 2006.

Anexo

Para presentar los principales hallazgos del estudio, las tablas y los gráficos son claves, pues en ellos se ilustran claramente los resultados obtenidos

A. Formato de Ecuaciones

$$C(n) = \frac{(n-1)!}{2} \quad (1)$$

Donde:

$C(n)$ = Número de ciclos hamiltonianos para n ciudades

n = Números de ciudades

m = Longitud del patrón

B. Figuras

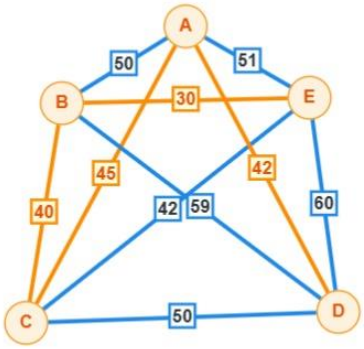

Fig. 1. *Grafo de generación de permutaciones*

El diagrama muestra el proceso de generación y evaluación de todas las permutaciones posibles de ciudades para encontrar el recorrido con la distancia mínima.