

Diseño y Desarrollo de un Resolvedor para Cubos de Rubik con Bloqueos

Trabajo de Fin de Grado, Grado en Ingeniería Informática

Alejandro Soriano Compta

20 junio de 2025



POLITÉCNICA

UNIVERSIDAD
POLITÉCNICA
DE MADRID



ESCUOLA TÉCNICA
SUPERIOR DE INGENIEROS
INFORMÁTICOS

Introducción

Rubik



Ernő Rubik

Introducción

Rubik



Ernő Rubik

NOTES ON

RUBIK'S MAGIC CUBE

by

DAVID SINGMASTER

Lecturer in Mathematical Sciences and Computing
Polytechnic of the South Bank
London, England

Introducción

Algoritmo de Korf (1997)

Finding Optimal Solutions to Rubik's Cube Using Pattern Databases

Richard E. Korf

Computer Science Department
University of California, Los Angeles
Los Angeles, Ca. 90095
Korf@cs.ucla.edu

Abstract

We have found the first optimal solutions to random instances of Rubik's Cube. The median optimal solution length appears to be 18 moves. The algorithm used is iterative-deepening-A* (IDA*), with a lower-bound heuristic function based on large memory-based lookup tables, or "pattern databases" (Culberson and Schaeffer 1996). These tables store the exact number of moves required to solve various subgoals of the problem, in this case subsets of the individual movable cubies. We characterize the effectiveness of an admissible heuristic function by its expected value, and hypothesize that the overall performance of the program obeys a relation in which the product of the time and space used equals the size of the state space. Thus, the speed of the program increases linearly with the amount of memory available. As computer memories become larger and cheaper, we believe that this

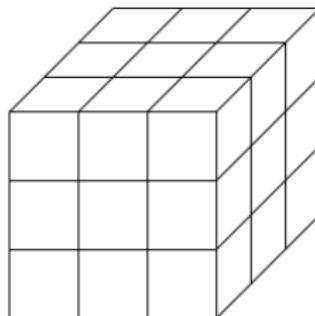
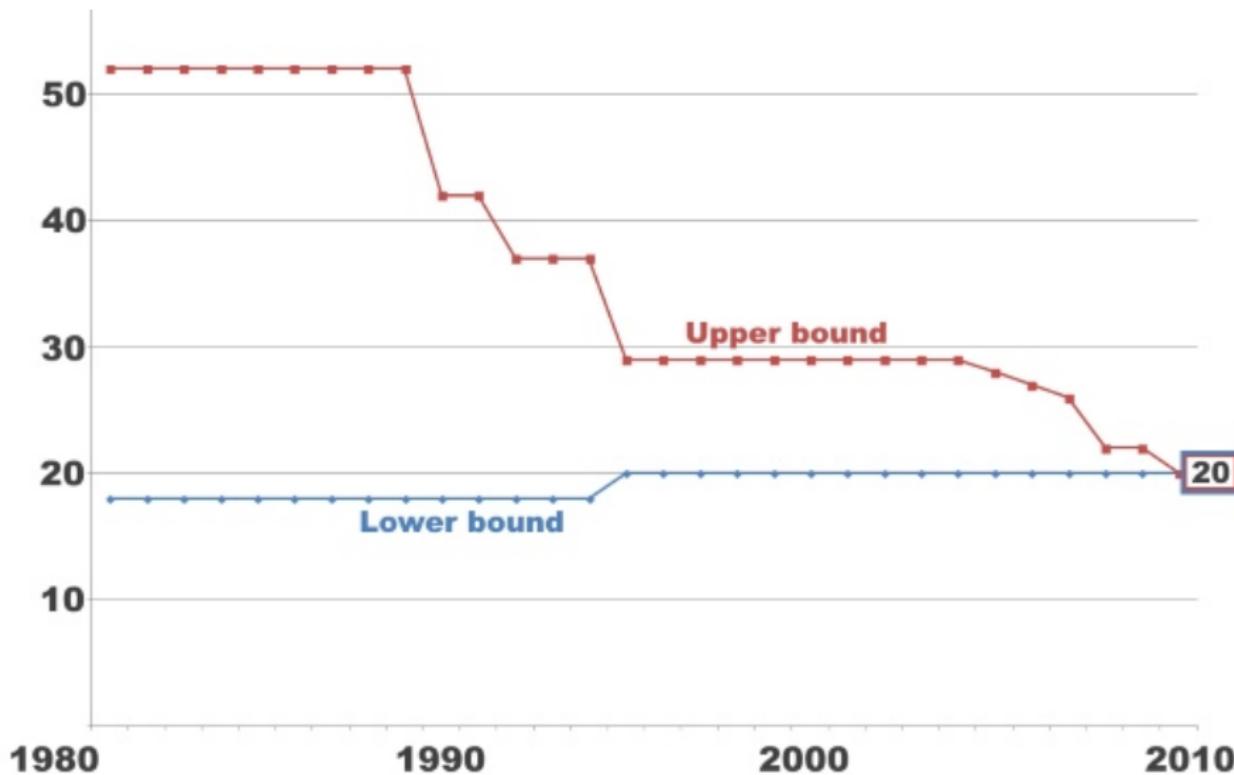


Figure 1: Rubik's Cube

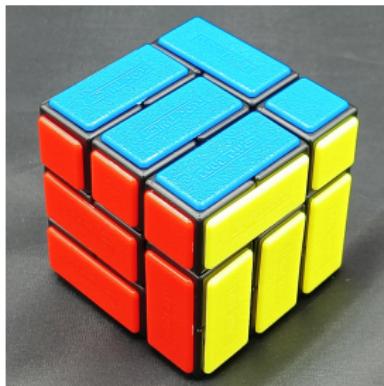
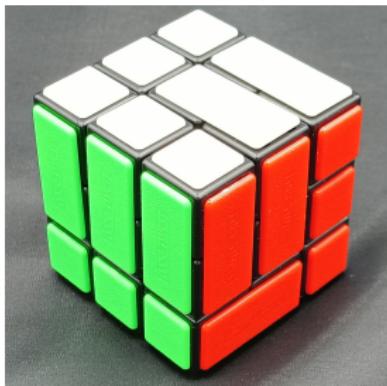
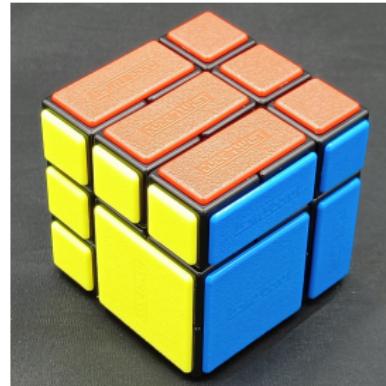
Introducción

Número de Dios



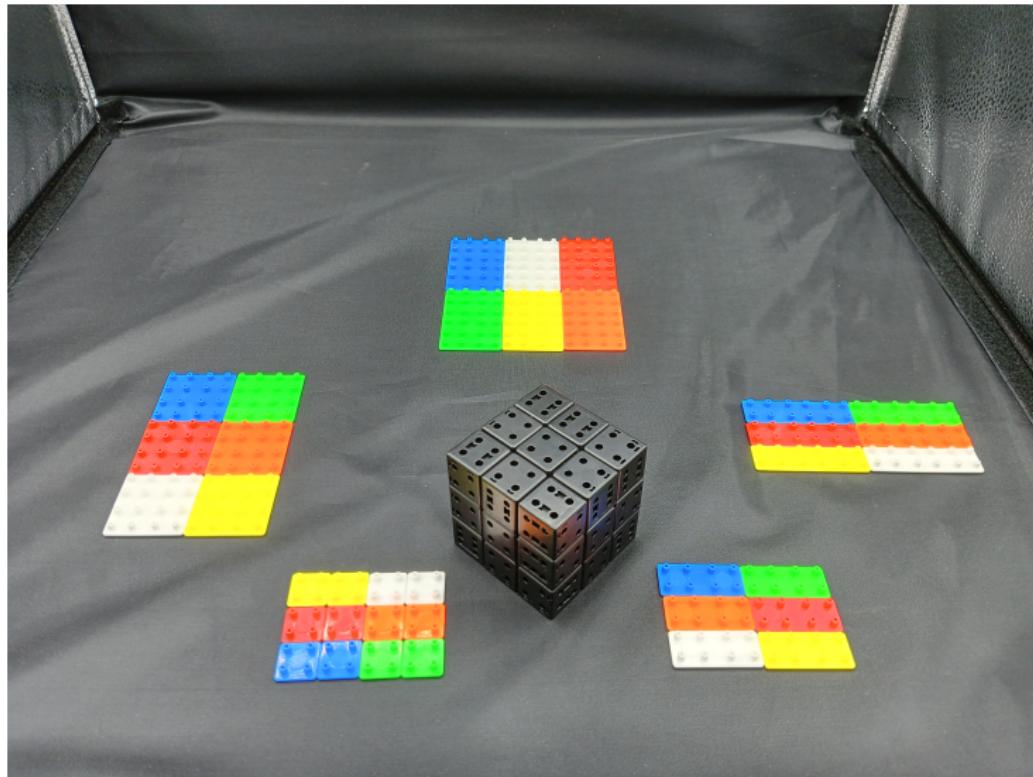
Bandaged Cubes

Cubos Alcatraz y Bicube



Bandaged Cubes

Bloqueos configurables



Dificultad

Pérdida de su estructura algebraica

Algoritmo de Thistlethwaite (1981)

$$G0 = \langle R, L, F, B, U, D \rangle$$

$$G1 = \langle R, L, F, B, U2, D2 \rangle$$

$$G2 = \langle R, L, F2, B2, U2, D2 \rangle$$

$$G3 = \langle R2, L2, F2, B2, U2, D2 \rangle$$

$$G4 = I(\text{resuelto})$$

Algoritmo de Kociemba (1992)

$$G0 = \langle R, L, F, B, U, D \rangle$$

$$G1 = \langle R2, L2, F2, B2, U, D \rangle$$

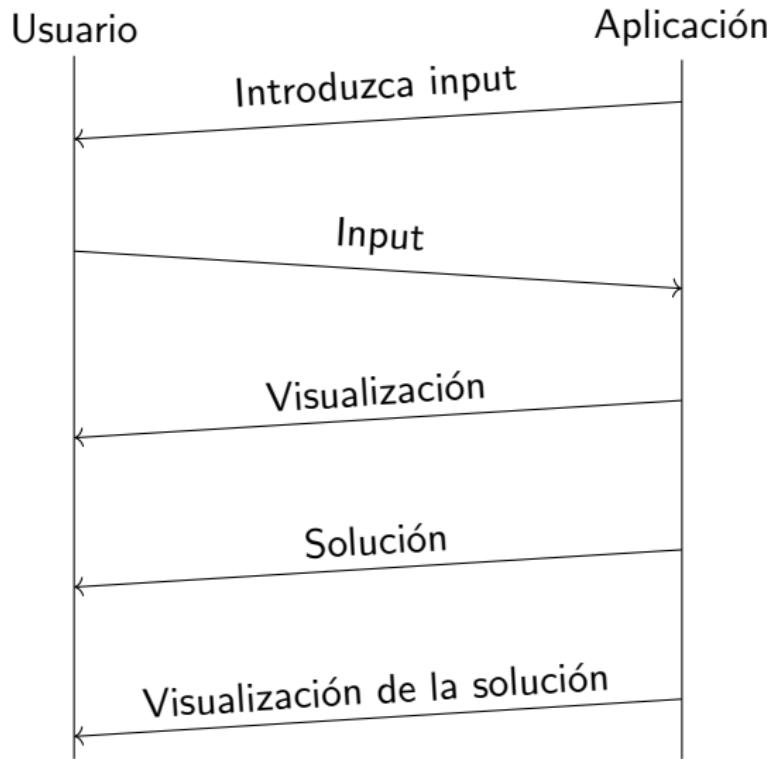
$$G2 = I(\text{resuelto})$$

- Los métodos basados en teoría de grupos no son válidos. La cota de 20 movimientos tampoco.
- Un cubo estándar tiene $4,3 \cdot 10^{19}$ estados posibles, un petabyte son 10^{15} bytes.

Requisitos

- Implementar una librería de funciones para codificar cubos con bloqueos
- Encontrar una solución **correcta y óptima** a un cubo en tiempo razonable
- Introducir un estado de un cubo con o sin bloqueos
- Procesar el input, detectar posibles errores y completar los bloques
- Generar una visualización del cubo original
- Generar otra visualización aplicando los giros para resolverlo

Caso de uso



Input

Orden de las piezas

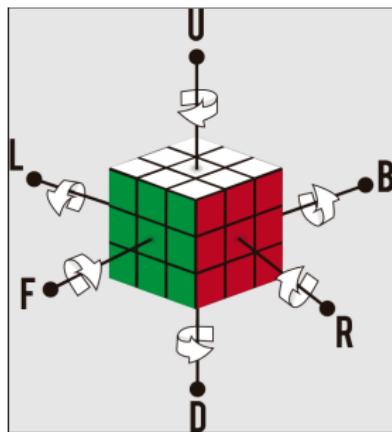
| | | |
|----|----|----|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |
| 37 | 38 | 39 |
| 19 | 20 | 21 |
| 10 | 11 | 12 |
| 46 | 47 | 48 |
| 40 | 41 | 42 |
| 22 | 23 | 24 |
| 13 | 14 | 15 |
| 49 | 50 | 51 |
| 43 | 44 | 45 |
| 25 | 26 | 27 |
| 16 | 17 | 18 |
| 52 | 53 | 54 |
| 28 | 29 | 30 |
| 31 | 32 | 33 |
| 34 | 35 | 36 |

Output (consola)

Solution found: D' F' D R' D2 R D F D2 F' D'
11 moves

Output (consola)

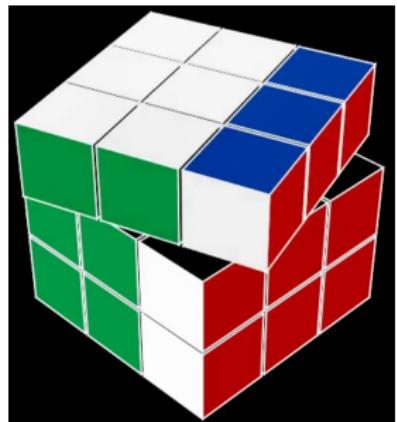
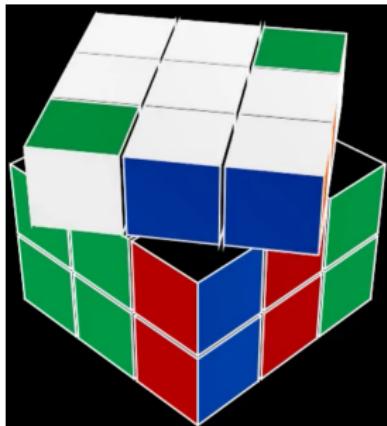
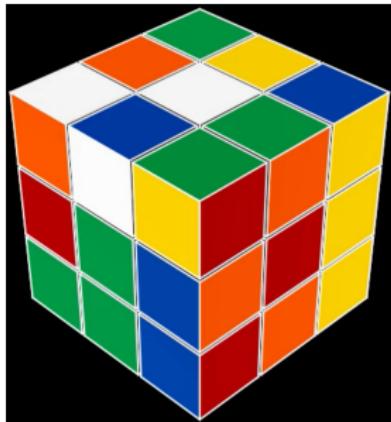
*Solution found: D' F' D R' D2 R D F D2 F' D'
11 moves*



<https://twistypuzzles.fandom.com/wiki/Notation>

Output y visualización (vídeo)

*Solution found: D' F' D R' D2 R D F D2 F' D'
11 moves*



Representación de un cubo

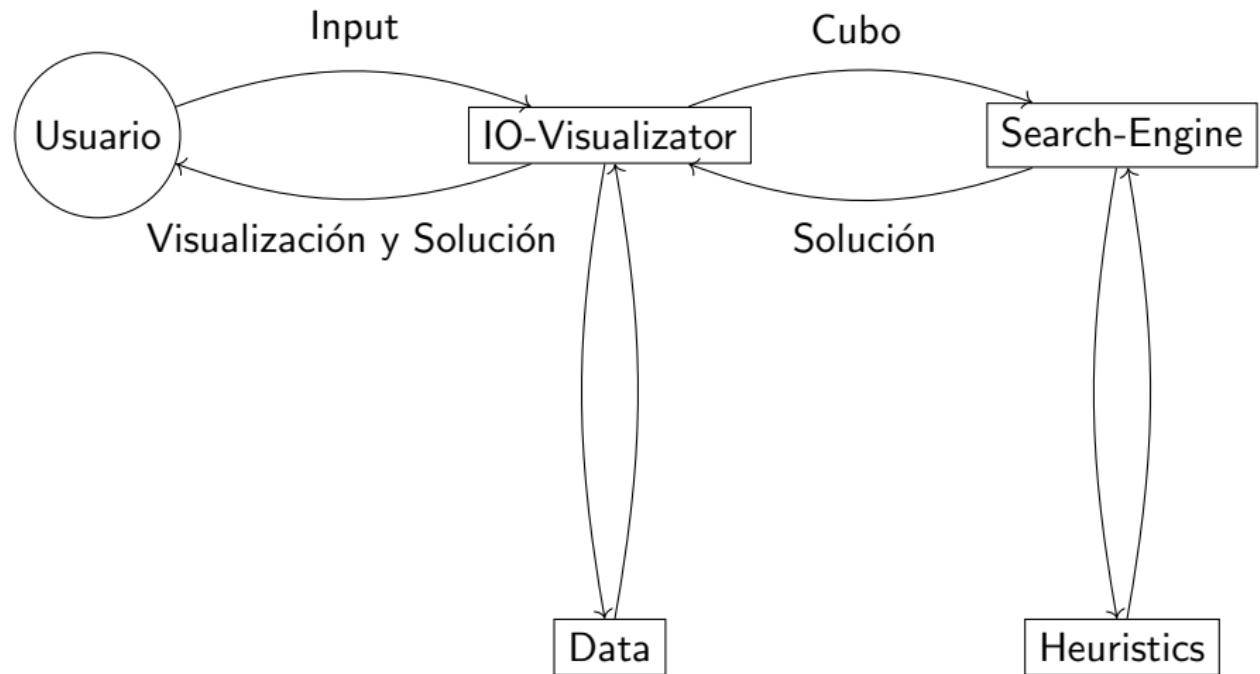
| | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 3 | 26 | 6 | | | | | | | | | |
| 24 | 48 | 28 | | | | | | | | | |
| 0 | 30 | 9 | | | | | | | | | |
| 4 | 25 | 2 | 1 | 31 | 11 | 10 | 29 | 8 | 7 | 27 | 5 |
| 39 | 52 | 33 | 32 | 49 | 34 | 35 | 50 | 37 | 36 | 51 | 38 |
| 20 | 47 | 22 | 23 | 41 | 13 | 14 | 43 | 16 | 17 | 45 | 19 |
| | | | 21 | 40 | 12 | | | | | | |
| | | | 46 | 53 | 42 | | | | | | |
| | | | 18 | 44 | 15 | | | | | | |

Codificación de un estado ($x \in S_{54}$)

Ejemplo: Cubo: $[3, 4, 5, 0, 1, 2, \dots, 52, 53]$

Bloques: $\{\{0, 1, 2, 30, 31\}, \{53, 40, 41\}\}$

Diseño software



¿Qué algoritmo podemos usar?

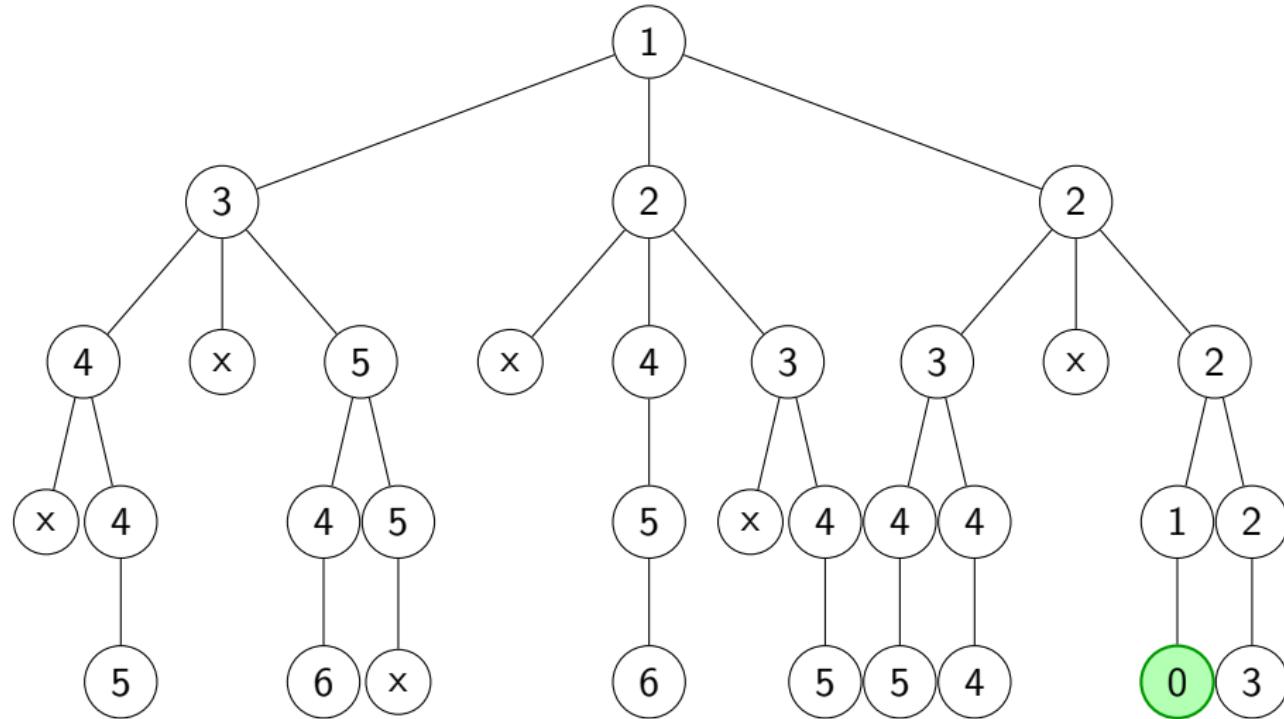
Adaptación del algoritmo de Korf (1997)

- Basado en IDA* (iterative deepening A*)
- Profundidad iterativa
- Heurísticas almacenadas en bases de datos de patrones
- $\mathcal{O}(b^d)$ en tiempo, $\mathcal{O}(d)$ en memoria
- (b es el factor de ramificación, d es la profundidad de la solución)

Algoritmo de búsqueda (Korf)

Ejemplo de IDA*

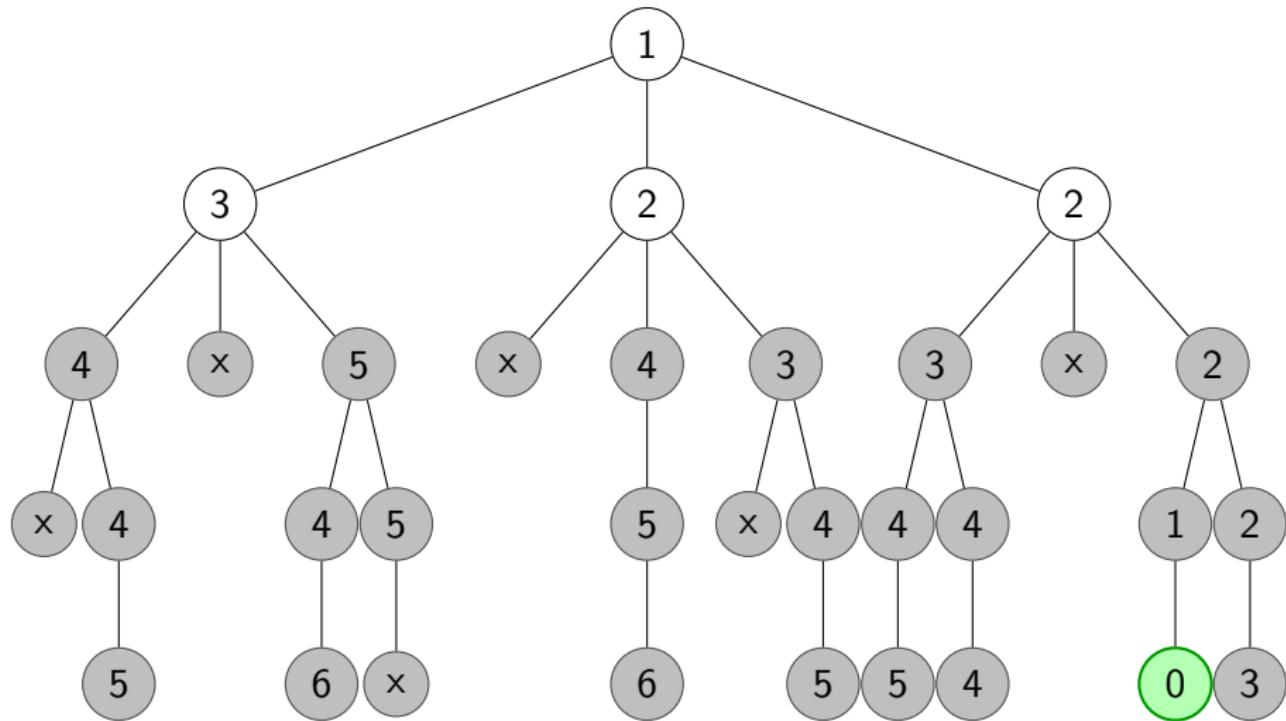
Inicio



Algoritmo de búsqueda (Korf)

Ejemplo de IDA*

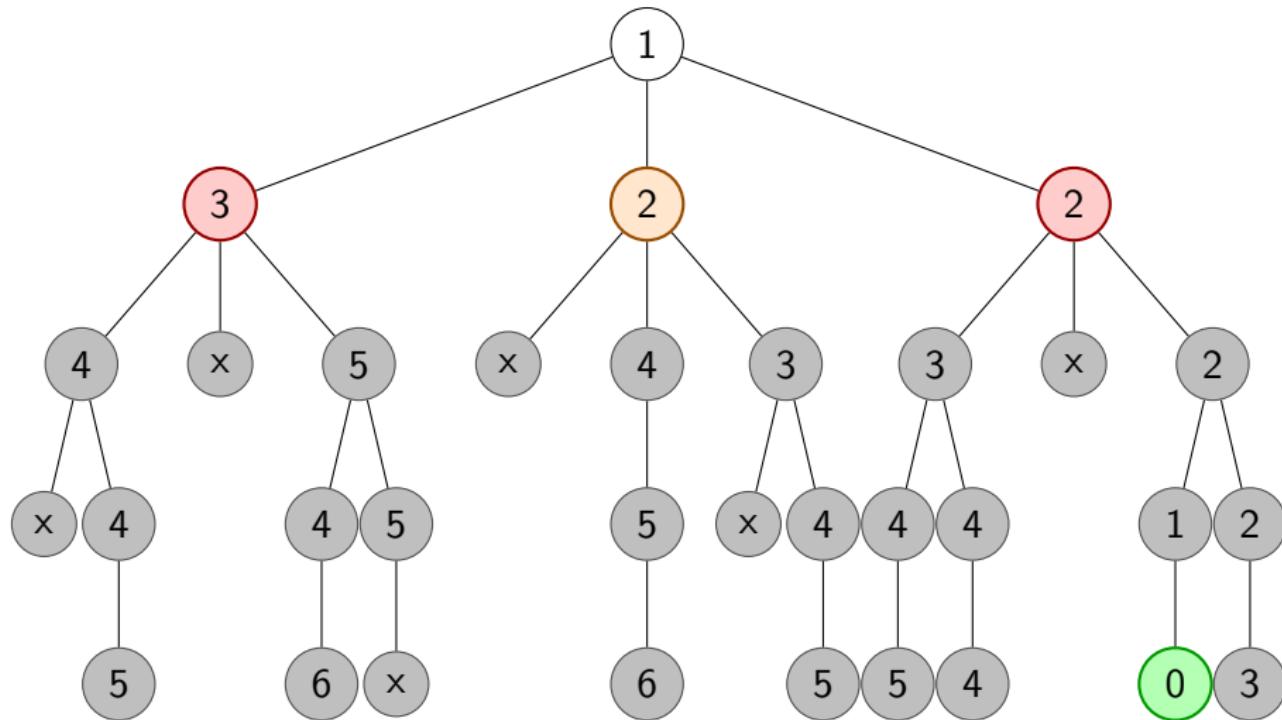
Primera iteración ($d = 1$)



Algoritmo de búsqueda (Korf)

Ejemplo de IDA*

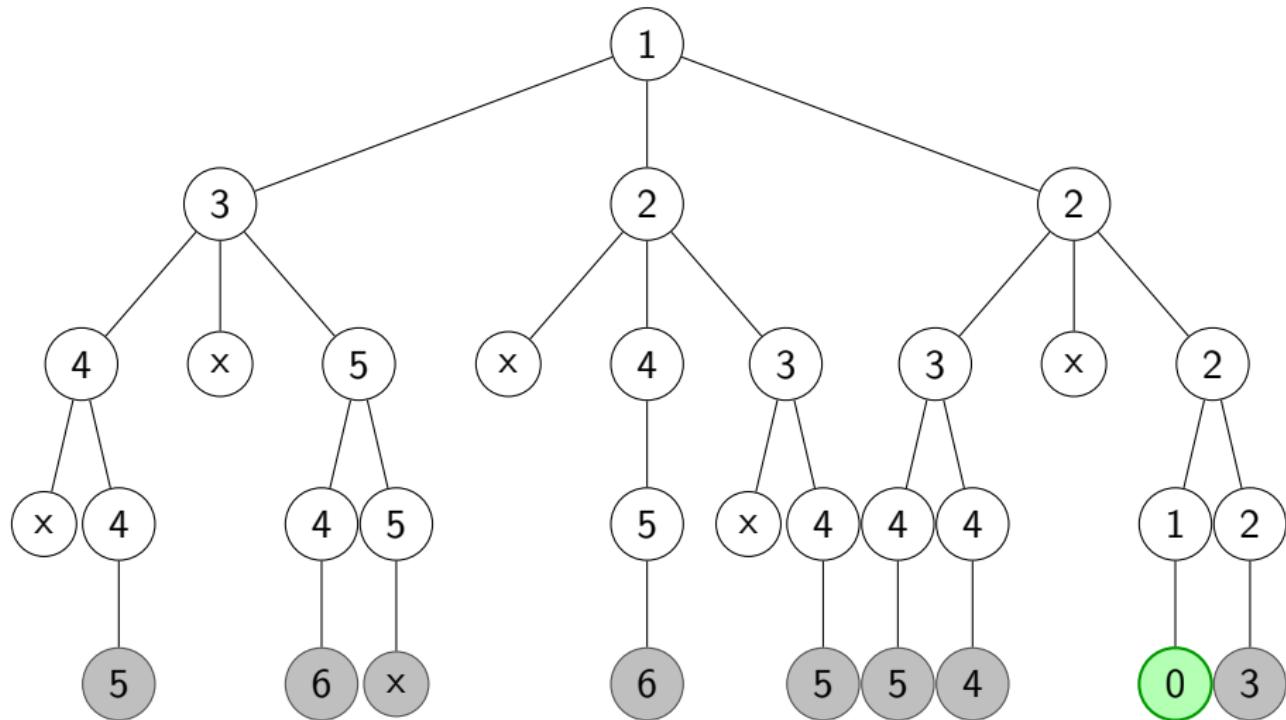
Primera iteración ($d = 1$)



Algoritmo de búsqueda (Korf)

Ejemplo de IDA*

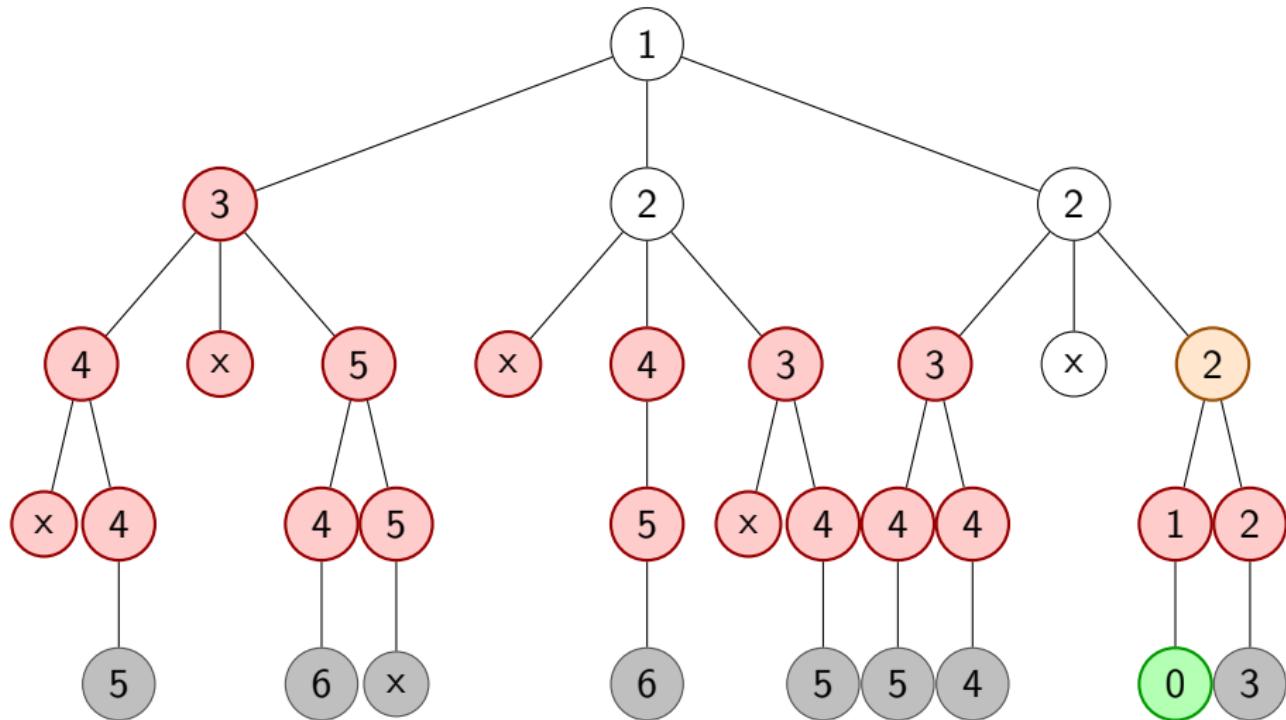
Segunda iteración ($d = 3$)



Algoritmo de búsqueda (Korf)

Ejemplo de IDA*

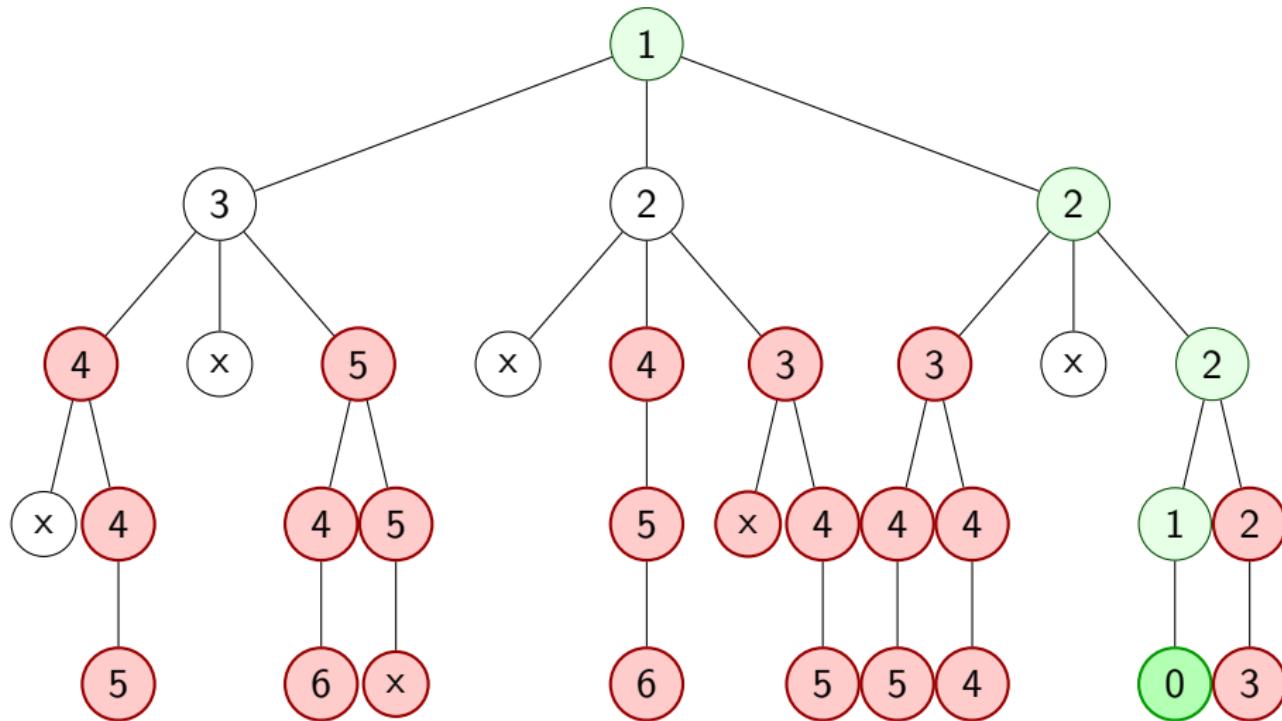
Segunda iteración ($d = 3$)



Algoritmo de búsqueda (Korf)

Ejemplo de IDA*

Tercera iteración ($d = 4$)



Algoritmo de búsqueda (Korf)

Generalidades de IDA*

Para un estado arbitrario n del árbol de búsqueda, se definen:

$$f(n) = g(n) + h(n)$$

- $f(n)$ movimientos mínimos totales pasando por n
- $g(n)$ movimientos realizados desde el inicio (real)
- $h(n)$ heurística de movimientos restantes (estimados, admisible)

Algoritmo de búsqueda (Korf)

Generalidades de IDA*

Para un estado arbitrario n del árbol de búsqueda, se definen:

$$f(n) = g(n) + h(n)$$

- $f(n)$ movimientos mínimos totales pasando por n
- $g(n)$ movimientos realizados desde el inicio (real)
- $h(n)$ heurística de movimientos restantes (estimados, admisible)
- Se fija una profundidad máxima d_i en cada iteración
- Se busca hasta d_i mediante bfs, solo explorando los nodos $f(n) \leq d_i$
- Empezar por $d_0 = h(n_0)$
- Si no se encuentra solución, actualizar d_{i+1} por el menor $f(n) > d_i$ y seguir buscando

Reducción del árbol de búsqueda

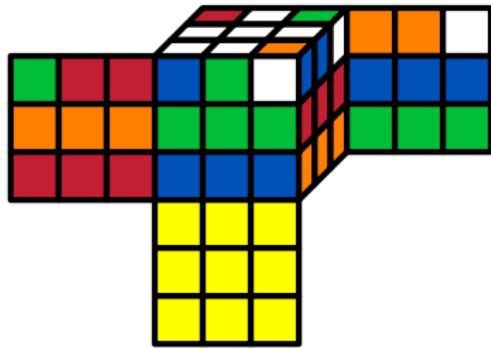
Secuencias canónicas

| Descripción | Ejemplo | Factor de ramificación |
|------------------------|------------------------------------|------------------------|
| 2 capas contiguas | $\langle R, U \rangle$ | 3 |
| 3 capas con 2 opuestas | $\langle R, L, U \rangle$ | 4.85 |
| 3 capas contiguas | $\langle R, U, F \rangle$ | 6 |
| 4 capas con 2 opuestas | $\langle R, L, U, F \rangle$ | 8.19 |
| 6 capas | $\langle R, L, U, D, F, B \rangle$ | 13.34 |

Factor de ramificación promedio de algunos subgrupos

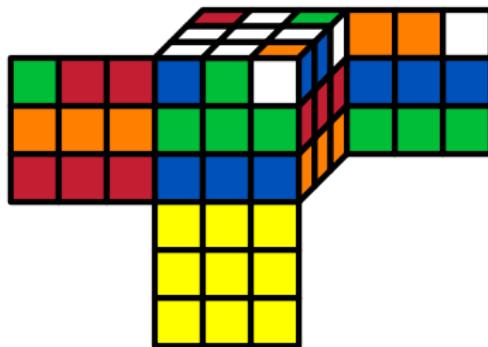
Heurísticas de Korf

Bases de datos de patrones



Heurísticas de Korf

Bases de datos de patrones



| Pieza | Movimientos necesarios |
|--------------|------------------------|
| Esquinas | 8 |
| Aristas 0-5 | 7 |
| Aristas 6-11 | 1 |
| Heurística | $\max\{8, 7, 1\} = 8$ |

Para saber el número de movimientos necesarios, partimos desde el estado resuelto, y exploramos el árbol mediante dfs hasta una profundidad determinada.

Tamaños de las bases de datos

| Pieza | Posibilidades |
|--------------|---|
| Esquinas | $8! \cdot 3^7 = 88,179,840$ |
| Aristas 0-5 | $\frac{12!}{6!} \cdot 2^6 = 42,577,920$ |
| Aristas 6-11 | $\frac{12!}{6!} \cdot 2^6 = 42,577,920$ |
| Total | 173.335.680 |

Si tenemos una función de hashing perfecta, podemos almacenar las heurísticas en vectores de enteros de 1 byte. Total **173 MB**.
Dividimos por tipos de piezas para reducir la cantidad de posibilidades.

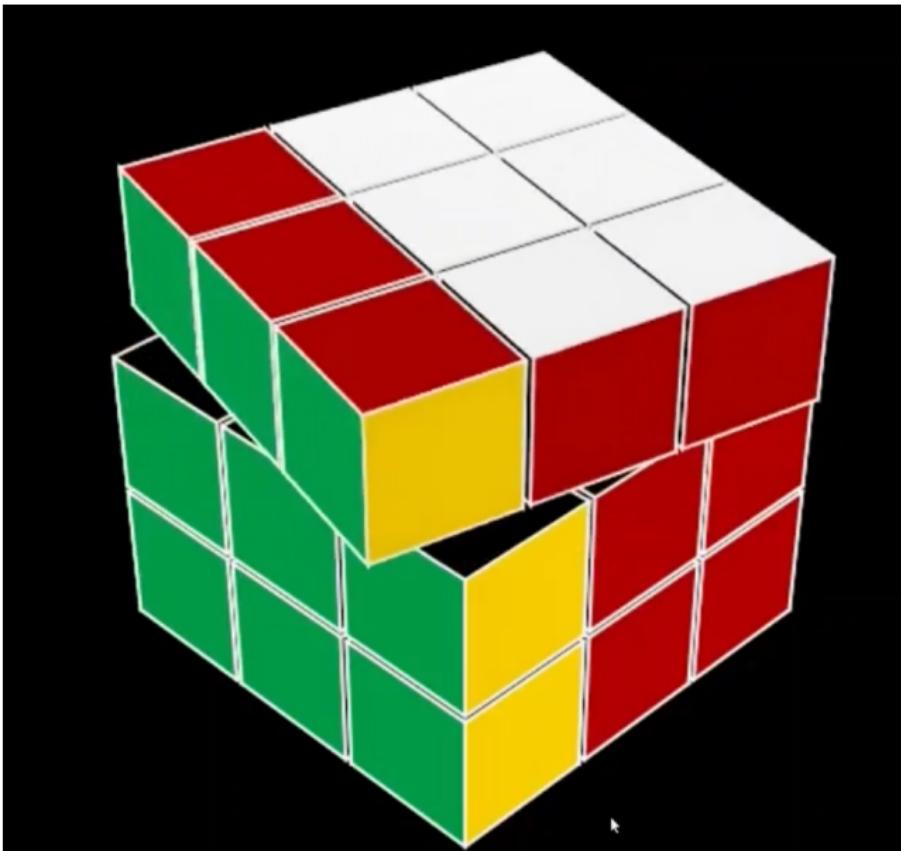
Indexación de heurísticas

Funciones de hashing perfectas

Un cubo es un elemento de $S_8 \times \mathbb{Z}_3^8 \times S_{12} \times \mathbb{Z}_2^{12}$

| | |
|-------------|--|
| Elemento | Asignación numérica (sistema de numeración) |
| Orientación | Base 3 ó 2 |
| Permutación | Orden lexicográfico (numeración factorial y npr) |

Vídeo de funcionamiento



Resultados

Optimidad de una mezcla



TheMaoiSha con el cubo deepest,
resoluble en 252 movimientos

- El solucionador encuentra una solución de tamaño 252
- Ha encontrado soluciones de 22 movimientos para los cubos Alcatraz y Bicube

<https://www.youtube.com/watch?v=gqBemWZscrk&t=499s>

Resultados

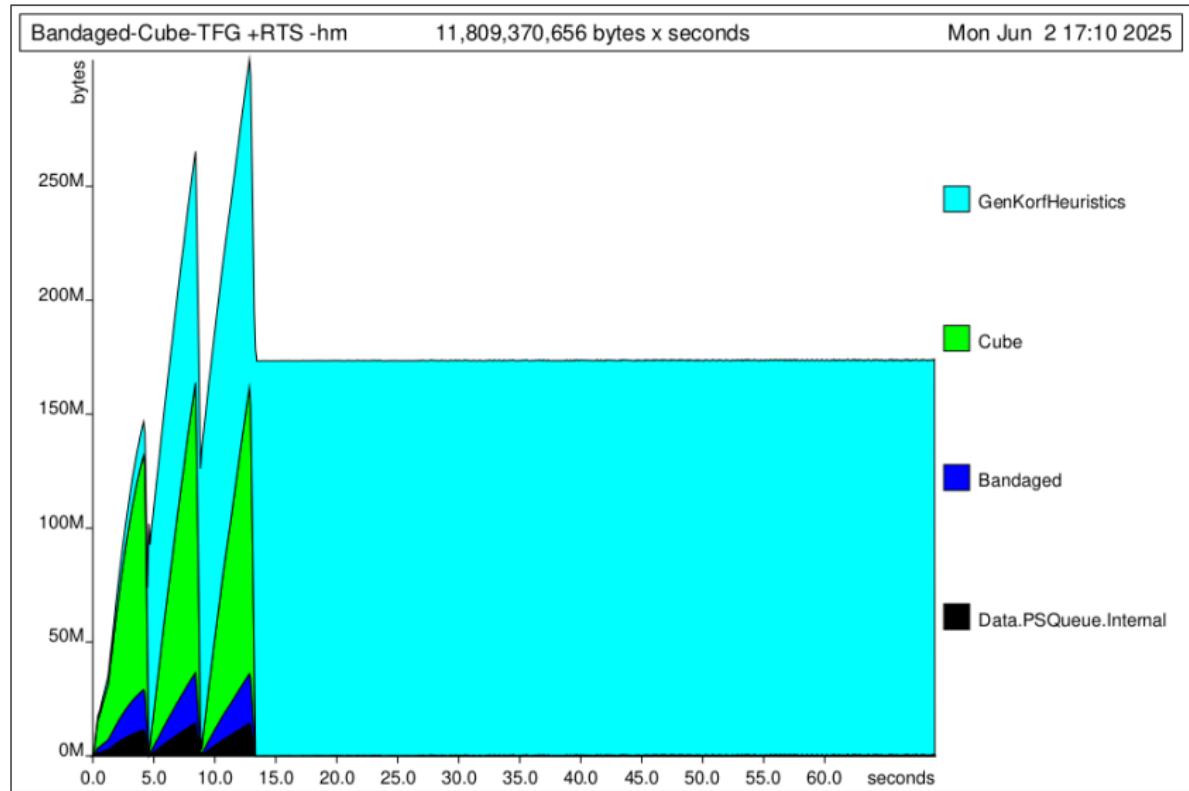
Tiempo de ejecución

| Cubo | Peor mezcla conocida | Segundos |
|----------------|----------------------|----------|
| Alcatraz | 22 | 14.25 s |
| Bicube | 22 | 8.5 s |
| TheMaoiSha-252 | 252 | 20 s |

La generación de heurísticas de profundidad 5 tarda entre 6.5 y 7 segundos

Resultados

Uso de memoria



Conclusiones

- Primer solucionador de este tipo
- Funcionamiento correcto en tiempo razonable
- Herramienta válida para encontrar algunos resultados
- Primera versión, funcionamiento aceptable con gran margen de mejora

Trabajo futuro

- Mejoras en la interfaz (corrección de errores, sencillez de uso, accesibilidad)
- Mejoras de eficiencia (heurísticas precomputadas)
- Probar otros métodos (subóptimos, concurrentes, distribuidos)
- Extender el solucionador a otros cubos
- Utilizar el solucionador para encontrar resultados

¡Viva el código libre!

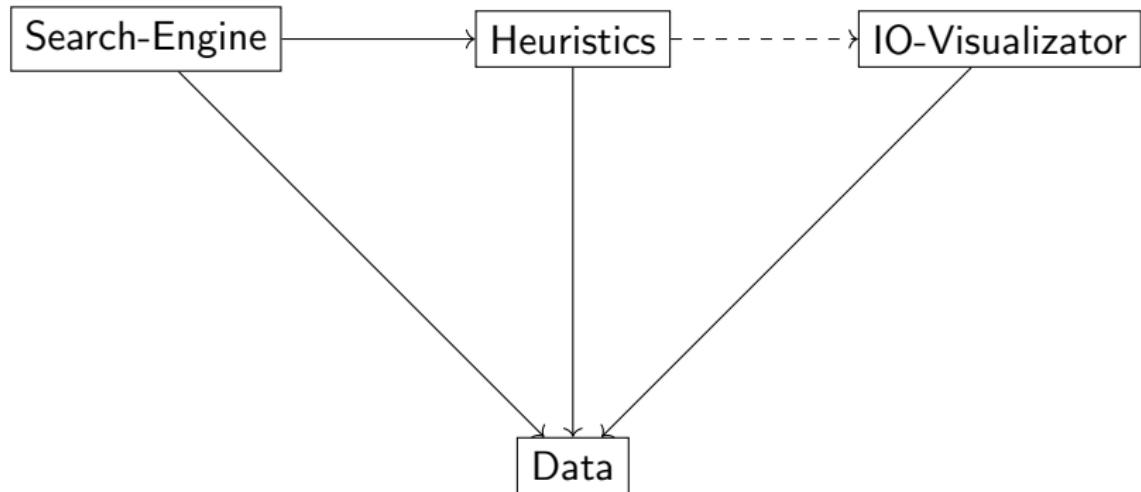


https://github.com/Alexsq1/Real_Bandaged_Cube_Explorer

Backup

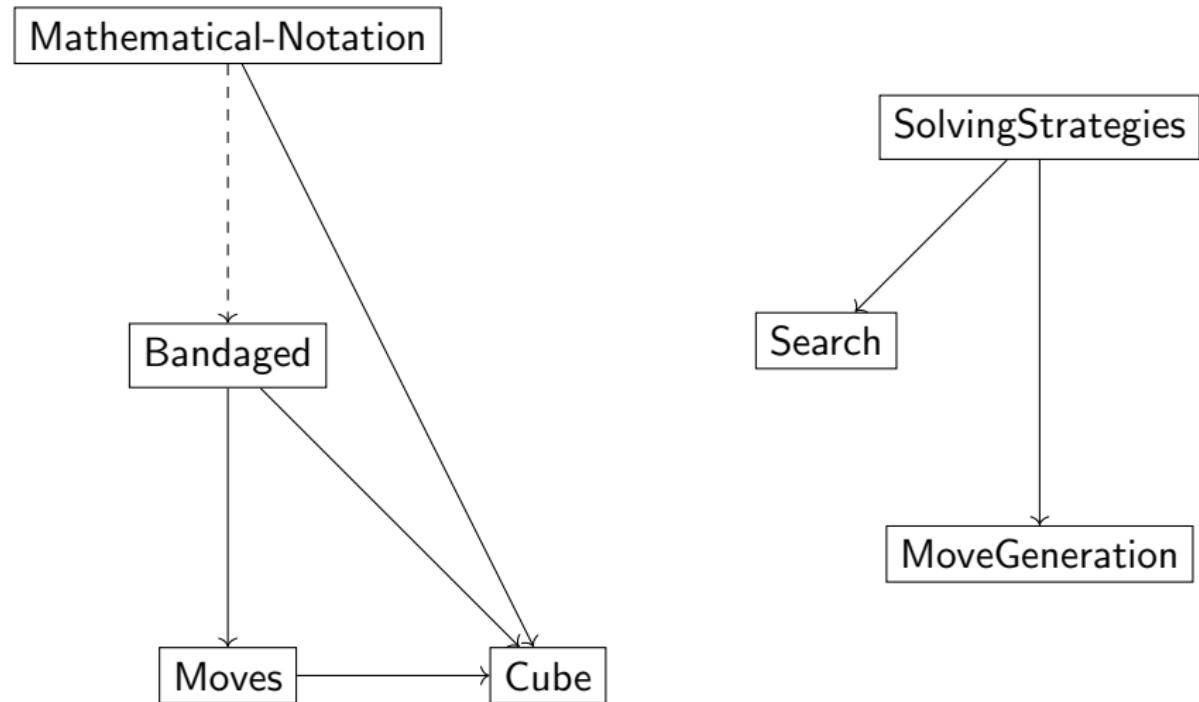
Diapositivas de reserva, no comentar en la defensa, tener preparadas por si se preguntase algo

División de paquetes y dependencias



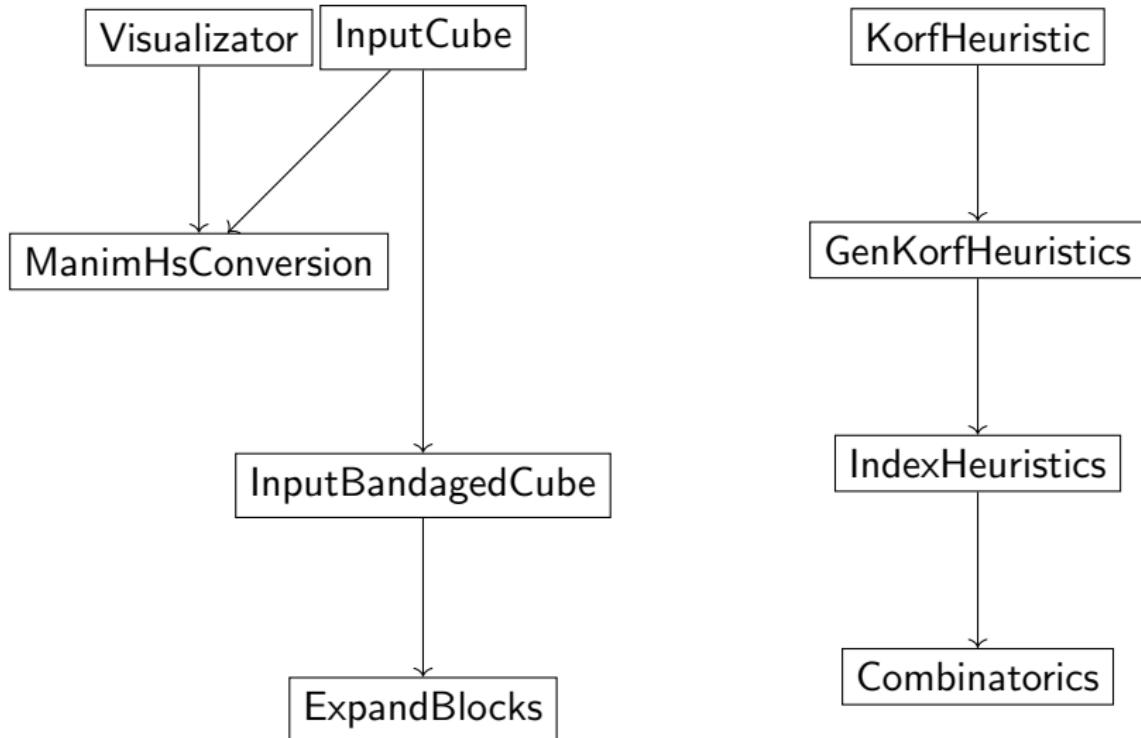
División de módulos

Data y Search



División de módulos

IO-Visualizer y Heuristics



Líneas de código de cada paquete

| Paquete | líneas de código aproximadas |
|---------------|------------------------------|
| Data | 450 |
| IO-Visualizer | 550 |
| Search | 250 |
| Heuristic | 300 |
| Total | 1550 |

Indexación de heurísticas

Ejemplo de orientación

Orientación de 1/2 aristas:

$$(1 \ 0 \ 1 \ 1 \ 0 \ 0)$$

$$o_e = 101100_2 = 44$$

$$o_e \in [0, 2^6 - 1]$$

Orientación de esquinas:

$$(0 \ 1 \ 2 \ 0 \ 0 \ 1 \ 1 \ 1)$$

$$o_c = 0120011x_3 = 409$$

$$o_c \in [0, 3^7 - 1]$$

Indexación de heurísticas

Ejemplos de permutación de esquinas

Transformar cada elemento por los números menores que no han aparecido. Posteriormente, multiplicar por las cifras factoriales según el sistema.

$$(4 \ 2 \ 3 \ 5 \ 1 \ 7 \ 6 \ 0)$$

$$4 : 2 : 2 : 2 : 1 : 2 : 1 : 0$$

$$p_c = 4 \cdot 7! + 2 \cdot 6! + 2 \cdot 5! + 2 \cdot 4! + 1 \cdot 3! + 2 \cdot 2! + 1 \cdot 1! + 0 \cdot 0! = 21,899$$

Combinando los elementos de permutación y orientación:

$$k_c = 3^7 \cdot p_c + o_c$$

Indexación de heurísticas

Ejemplo de permutación de aristas

$$(11 \ 0 \ 5 \ 10 \ 8 \ 9)$$

$$11 : 0 : 4 : 8 : 6 : 6$$

$$p_{e_i} = \frac{1}{(12-6)!} (11 \cdot 11! + 0 \cdot 10! + 4 \cdot 9! + 8 \cdot 8! + 6 \cdot 7! + 6 \cdot 6!) = 612,352$$

Combinando los elementos de permutación y orientación:

$$k_e = 2^6 \cdot p_e + o_e$$