Trabajo Práctico N°1

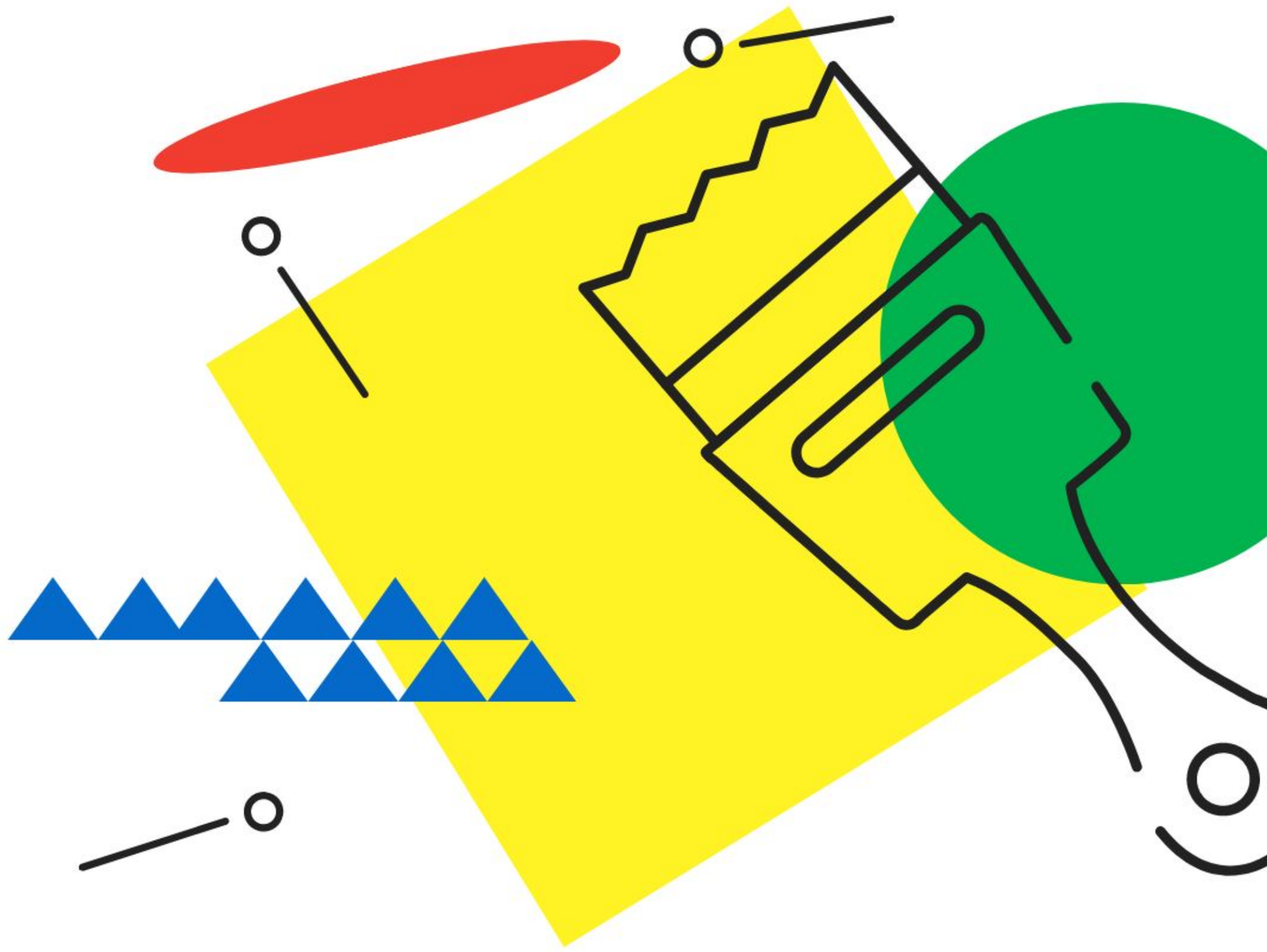
Métodos de Búsqueda + Algoritmos Genéticos

GRUPO 6

Luciana Diaz Kralj
Gonzalo Nicolas Rossin
João Nuno Diegues Vasconcelos
Mafalda Colaço Parente Morais Da Costa

Lado A: Métodos de Búsqueda.

TP N°1



01

Ejercicio 1:

8-Puzzle

Del tablero al azar al tablero solución

8-puzzle

8-puzzle consiste de un tablero de 3x3 con ocho fichas numeradas y un espacio en blanco, donde queremos llegar al tablero solución.

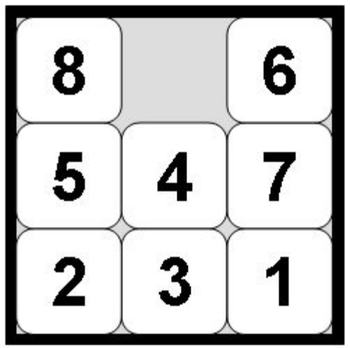
Nº de movimientos posibles en caso de espacio blanco		
2	3	2
3	4	3
2	3	2

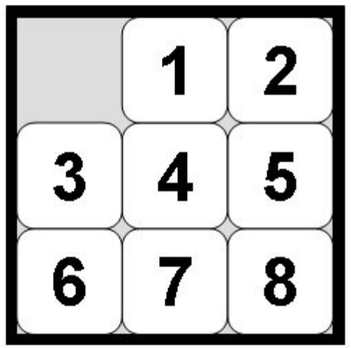
Acciones:

- Moviendo los números adyacentes al espacio vacío.
- Izquierda, derecha, arriba, abajo, dependiendo de la ubicación del mosaico.
- Branching factor de 2,66.

Estados: Hay 9! estados posibles, pero solo 9!/2 estados desde donde es posible llegar a la solución.

Condición de terminación: Los números en el tablero están ordenados.





8-puzzle

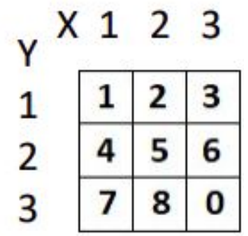
Estructura

Una matriz 3x3, con tiles con un valor que indica el orden [1 - 8] y un tuplo (Xs,Ys), que indica la posición del tile en la matriz.

Guardar la posición del espacio vacío (tile 0) como forma de mejorar la eficiencia.

Condiciones para poder moverse:

Nombre de movimiento	Condición previa	Costo
Arriba	Ys > 1	1
Abajo	Ys < 3	1
Izquierda	Xs > 1	1
Derecha	Xs < 3	1



Heurística 1

Fuera de Lugar

Cantidad de números que están fuera de su posición correcta.

Admisible porque no sobreestima su costo real.

5	7	3
8	2	
1	6	4

Solo el número 3 se encuentra en la posición correcta.

Entonces el valor de la heurística = 8 - 1 = 7

Heurística 2

Distancia de Manhattan

Distancia de los números que están fuera de su posición correcta: distancia horizontal + distancia vertical

Admisible porque nunca sobreestima el costo real.

1	7	3
8	2	4
	6	5

Los números 7 y 2 se encuentran en la posición incorrecta.

Heurística de 7: horizontal 1, vertical 2

Total de 7: horizontal + vertical = 3

Heurística total: 4

Heurística 3

Números incorrectos en línea y columna.

No es admisible porque sobreestima el costo real.

1	2	3
8	6	4
7		5

El número 6 se encuentra en la posición incorrecta y moverlo a la posición adecuada solo cuesta 1.

Heurística 2ª columna : 1
Heurística 2ª línea: 1

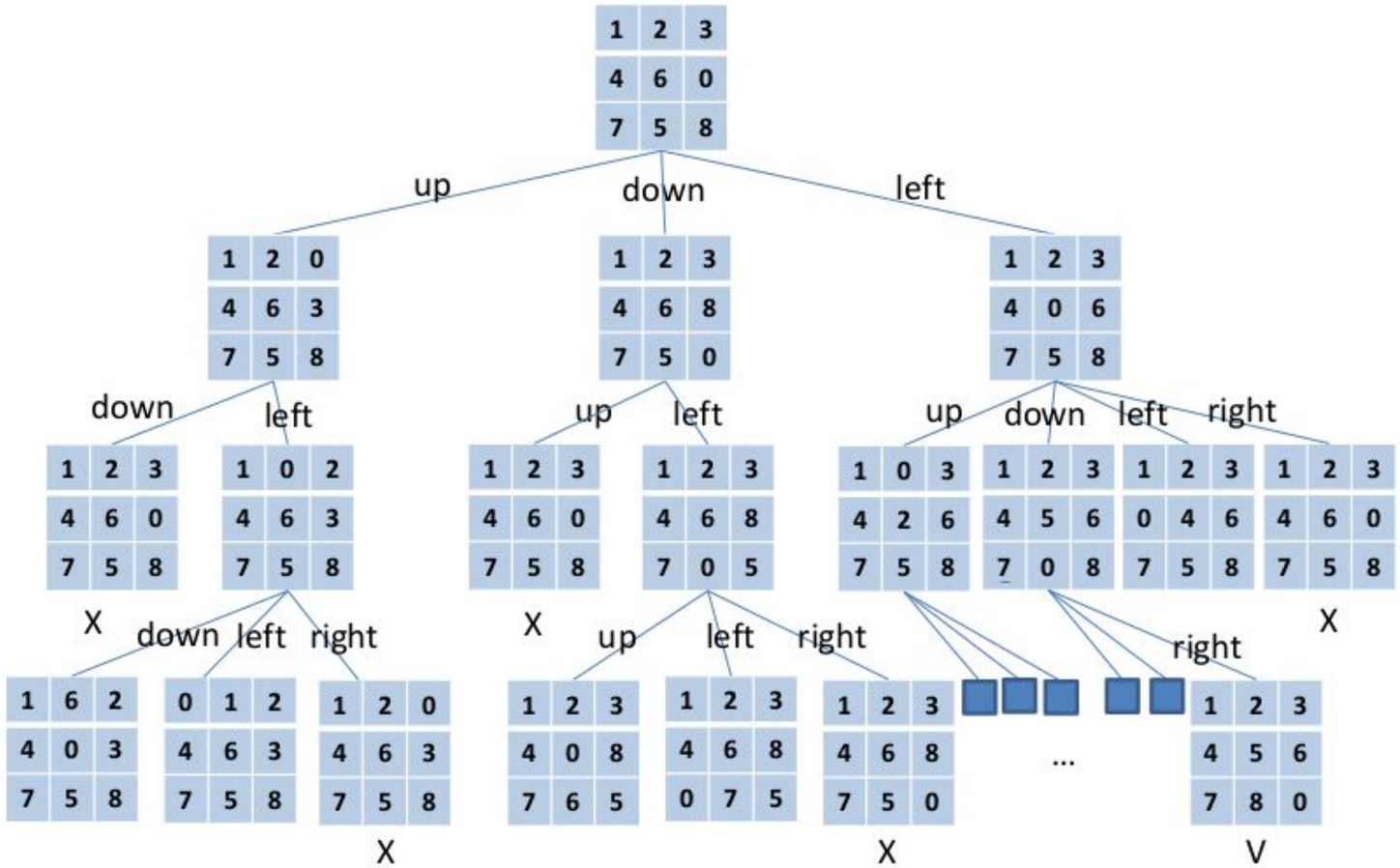
Heurística total: 2

Métodos de búsqueda

Métodos de búsqueda informados

Para este problema, dado que conocemos el estado objetivo y hemos desarrollado heurísticas, creemos que la mejor opción es utilizar métodos de búsqueda informados, de modo que deje atrás el uso de BFS y DFS, métodos de búsqueda desinformados.

BFS ejemplo


1 2 3
4 6 0
7 5 8
up
down
left
1 2 0
4 6 3
7 5 8
1 2 3
4 6 8
7 5 0
1 2 3
4 0 6
7 5 8
down
left
up
left
up
down
left
right
1 2 3
4 6 0
7 5 8
1 0 2
4 6 3
7 5 8
1 2 3
4 6 0
7 5 8
1 2 3
4 6 8
7 0 5
1 0 3
4 2 6
7 5 8
1 2 3
4 5 6
7 0 8
1 2 3
0 4 6
7 5 8
1 2 3
4 6 0
7 5 8
X
X
X
down
left
right
up
left
right
right
1 6 2
4 0 3
7 5 8
0 1 2
4 6 3
7 5 8
1 2 0
4 6 3
7 5 8
1 2 3
4 0 8
7 6 5
1 2 3
4 6 8
0 7 5
1 2 3
4 6 8
7 5 0
...
1 2 3
4 5 6
7 8 0
X
X
V

Métodos de busqueda

Greedy search

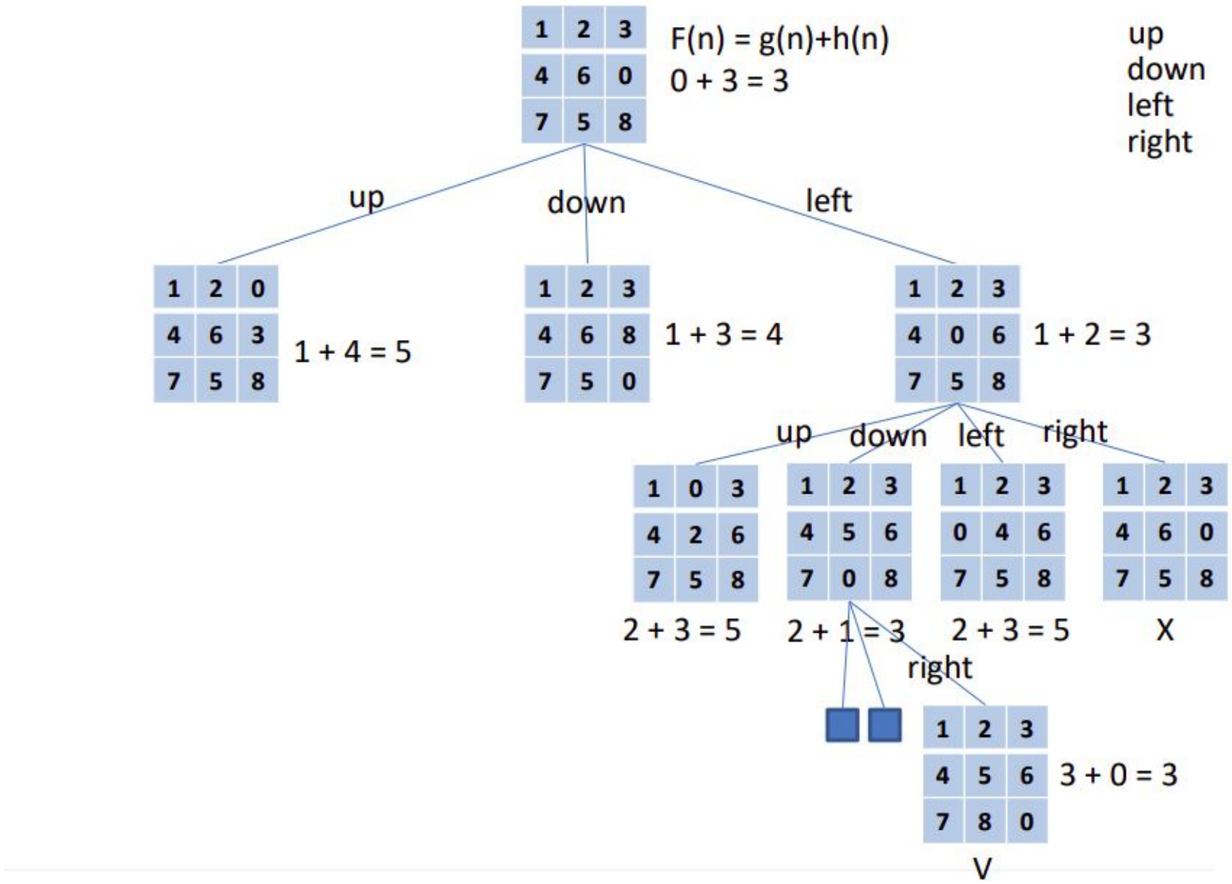
Con el Greedy Search simplemente elegimos el nodo que tiene el menor costo, eligiendo el Optimal Local Path.

A*

Algoritmo que se utiliza ampliamente en la búsqueda de rutas y en el recorrido de grafos, el proceso de trazar una ruta transversal entre múltiples puntos, denominados nodos. Reconocido por su rendimiento y precisión, goza de un uso generalizado.

A* sería lo mejor porque considera el camino anterior. Para eso, usa un priority queue con la suma del costo a llegar a ese nodo más el costo establecido por la heurística.

A* ejemplo



Usando la primera heurística

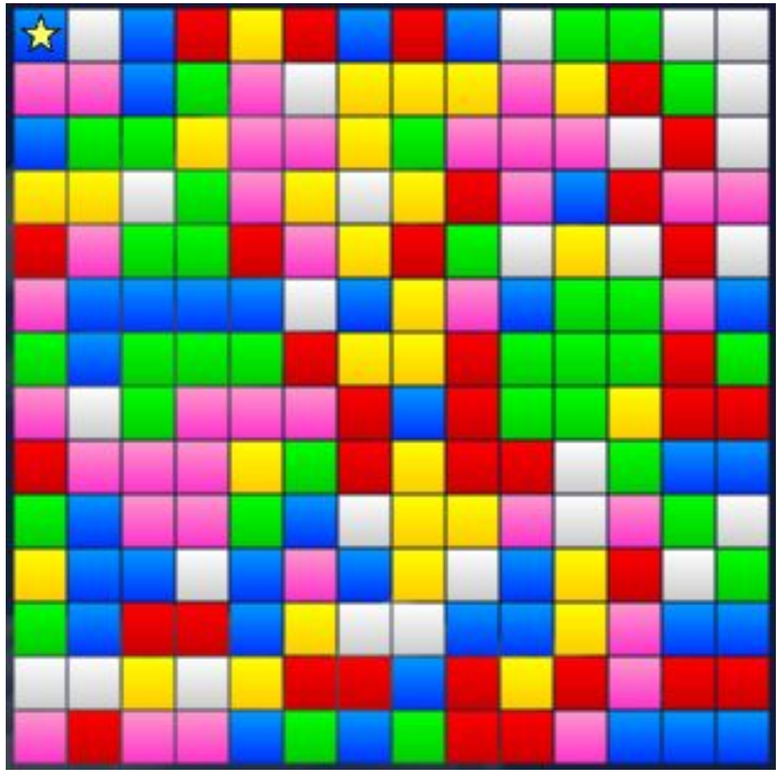
02

Ejercicio 2: Fill-Zone

La forma rápida de pintar toda la grilla

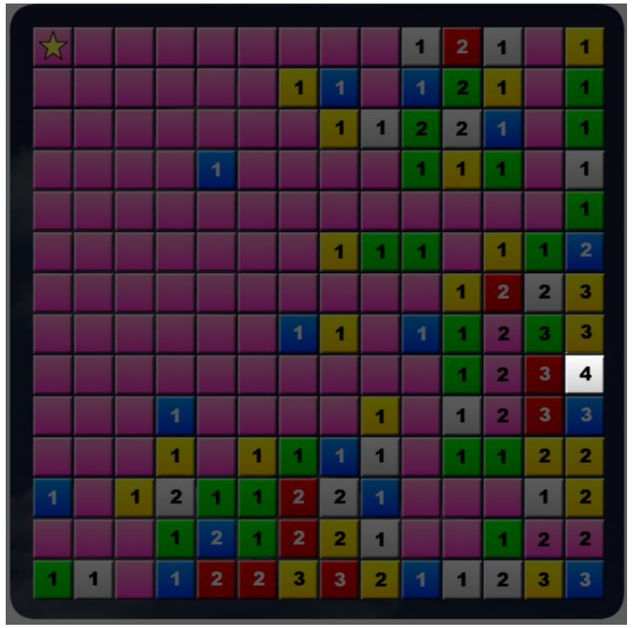
Fill-Zone

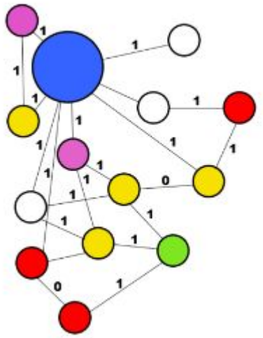
- El juego consiste en lograr que toda la grilla tenga el mismo color.
- Para esto, debemos cambiar el color de las celdas controladas por el jugador.
- Serán unidas todas las celdas adyacentes que sean del mismo color.



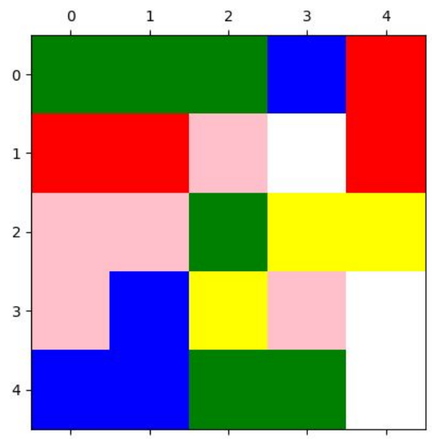
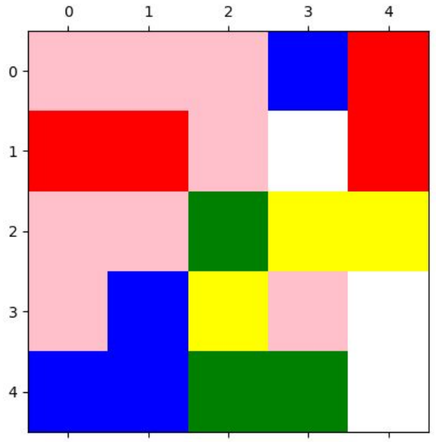
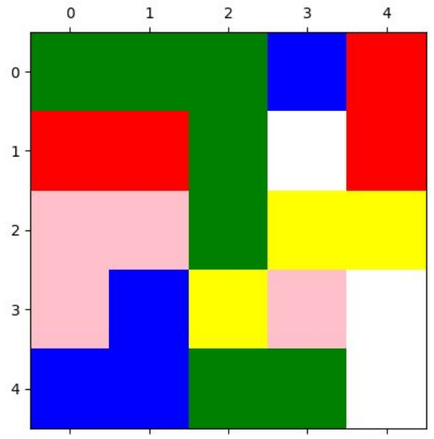
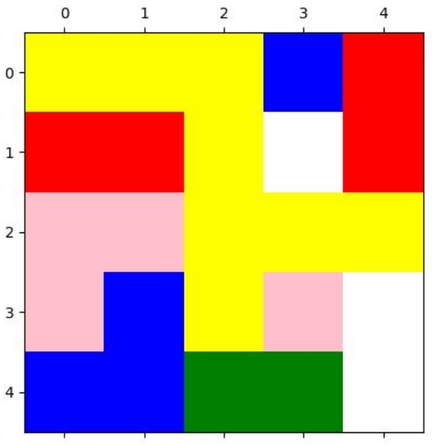
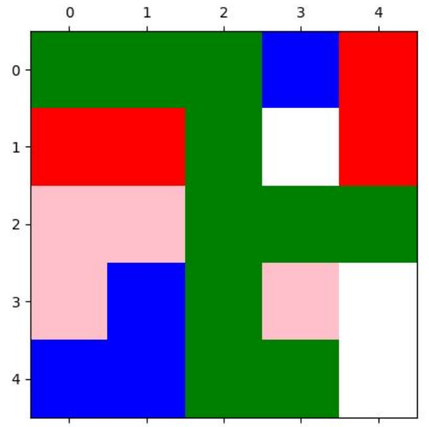
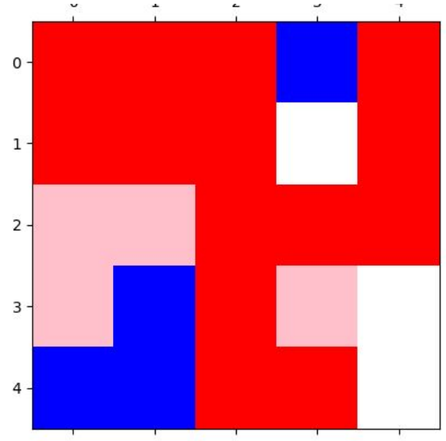
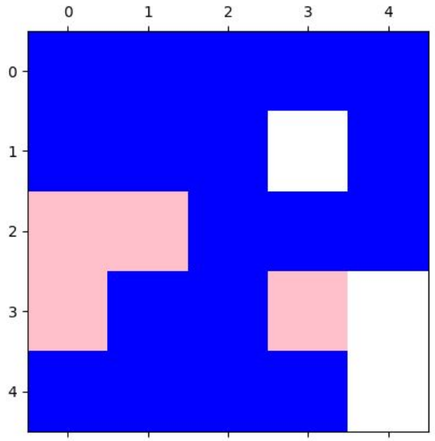
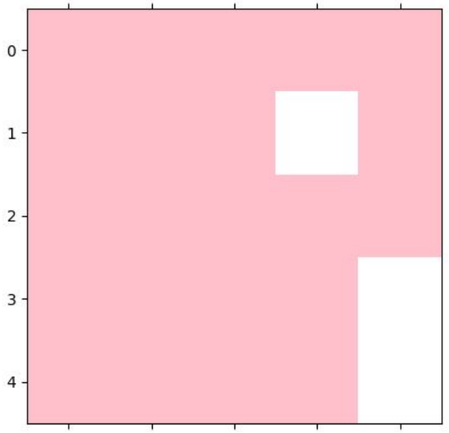
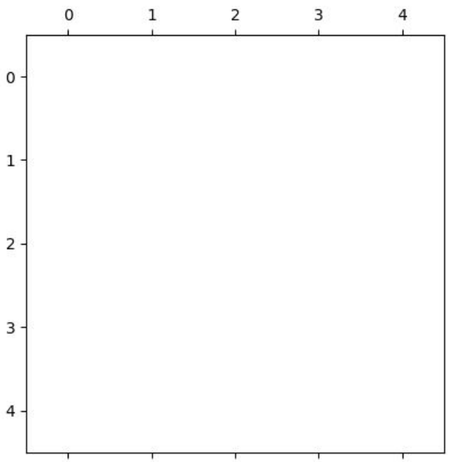
Heurística 1: Bronson Distance

- Estima la cantidad de pasos restantes para terminar el tablero en base a la cantidad de turnos necesarios para alcanzar la celda más alejada del tablero.
- Para conseguir este valor se realiza un grafo de las celdas no controladas y por medio de Dijkstra se obtiene el valor.
- Es admisible dado que para terminar el tablero se deberá llegar a este casillero y la cantidad de pasos para hacerlo no podrá ser inferior al valor calculado.





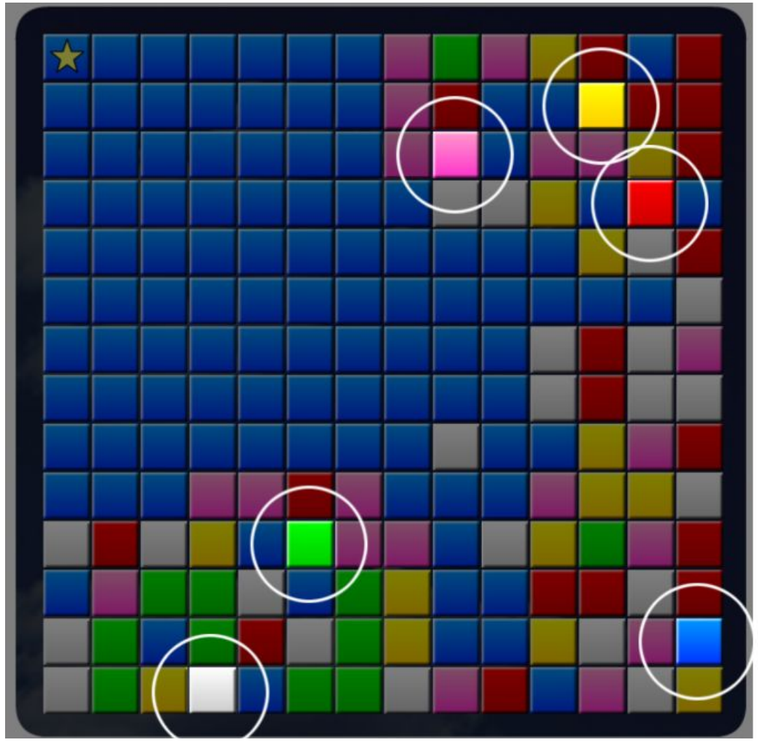
Heurística 1: Bronson Neighbors

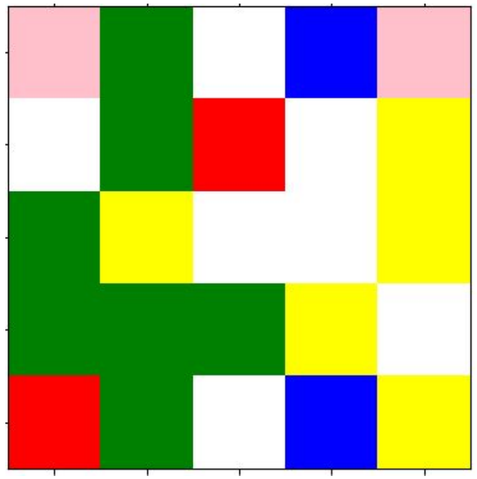
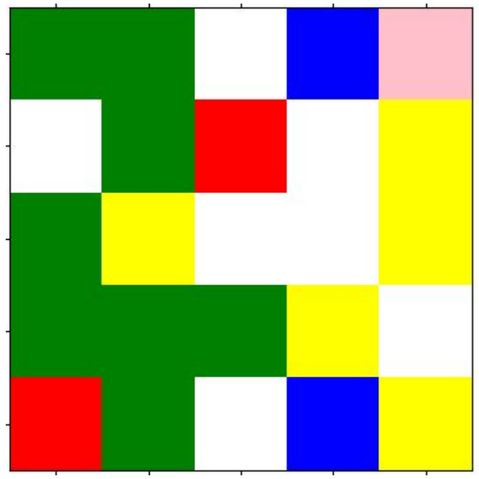
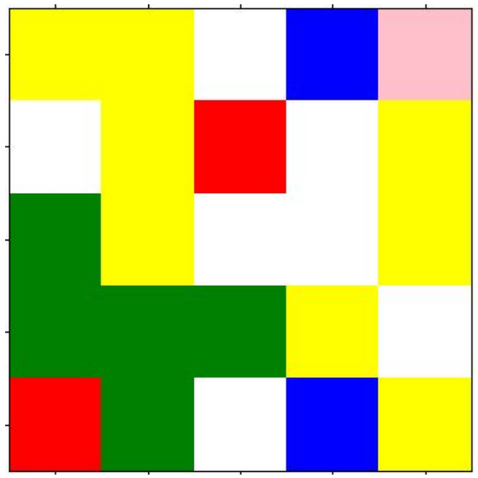
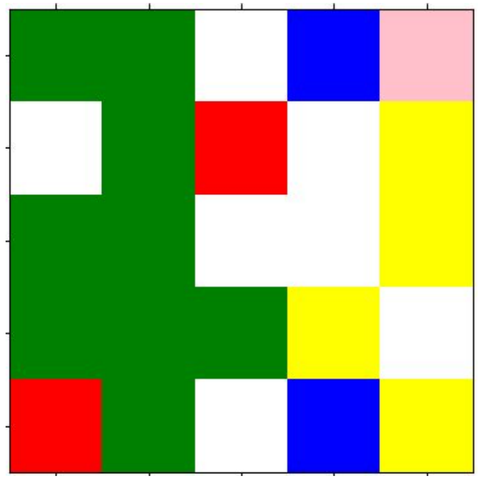
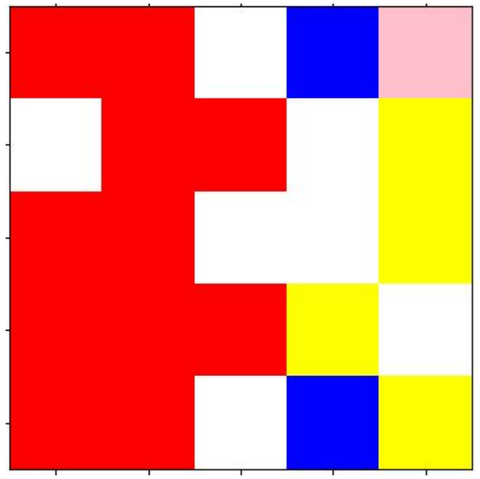
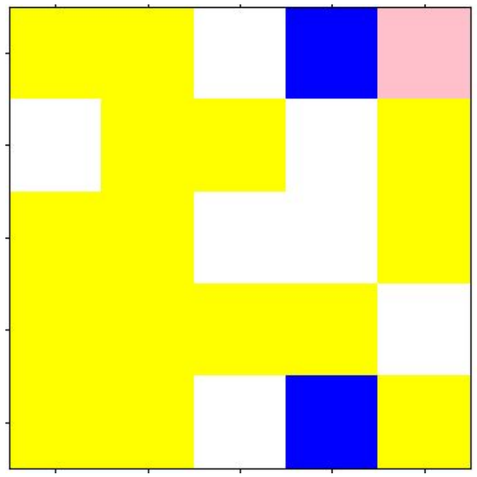
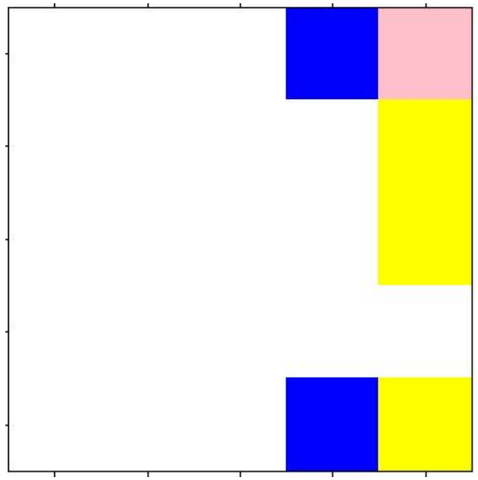
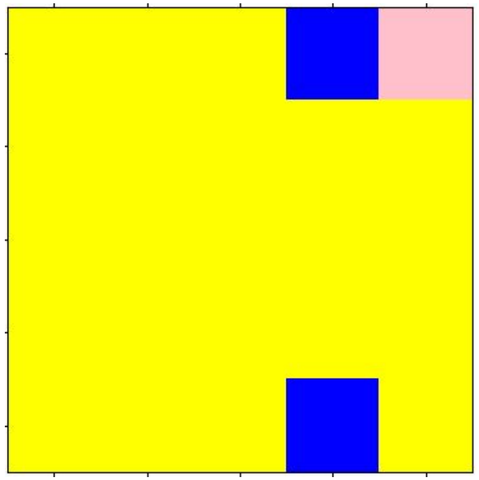
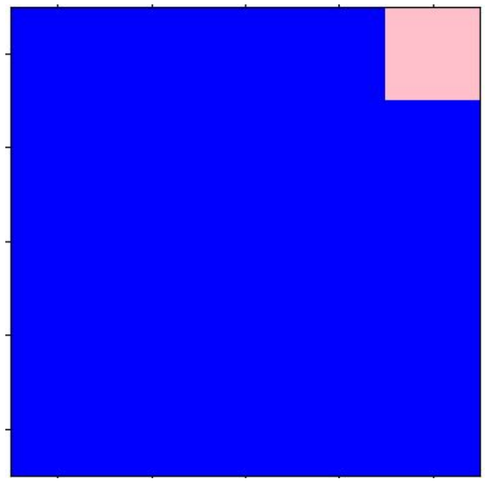
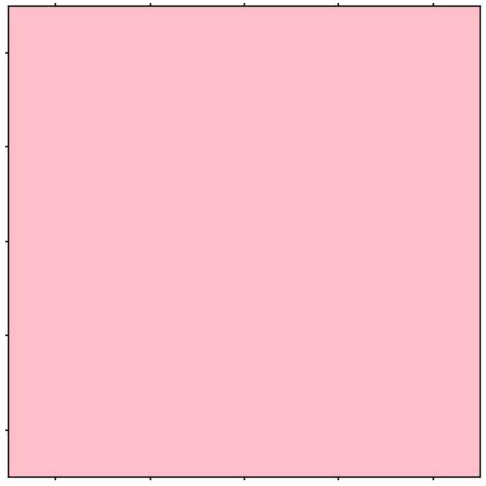
Heurística 2: Remaining colors

- Estima la cantidad de pasos restantes para terminar el tablero en base a la cantidad de colores diferentes no controlados restantes en el tablero.

- Es admisible ya que se debe cambiar al menos 1 vez a cada color no controlado para completar el juego.

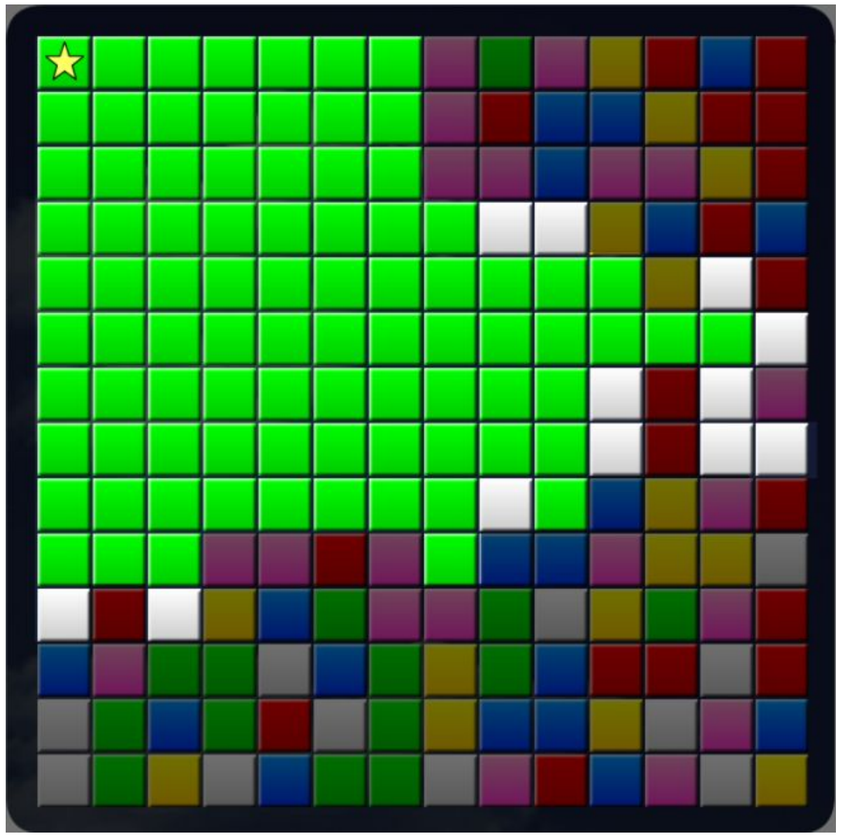


Heurística 2: Remaining colors

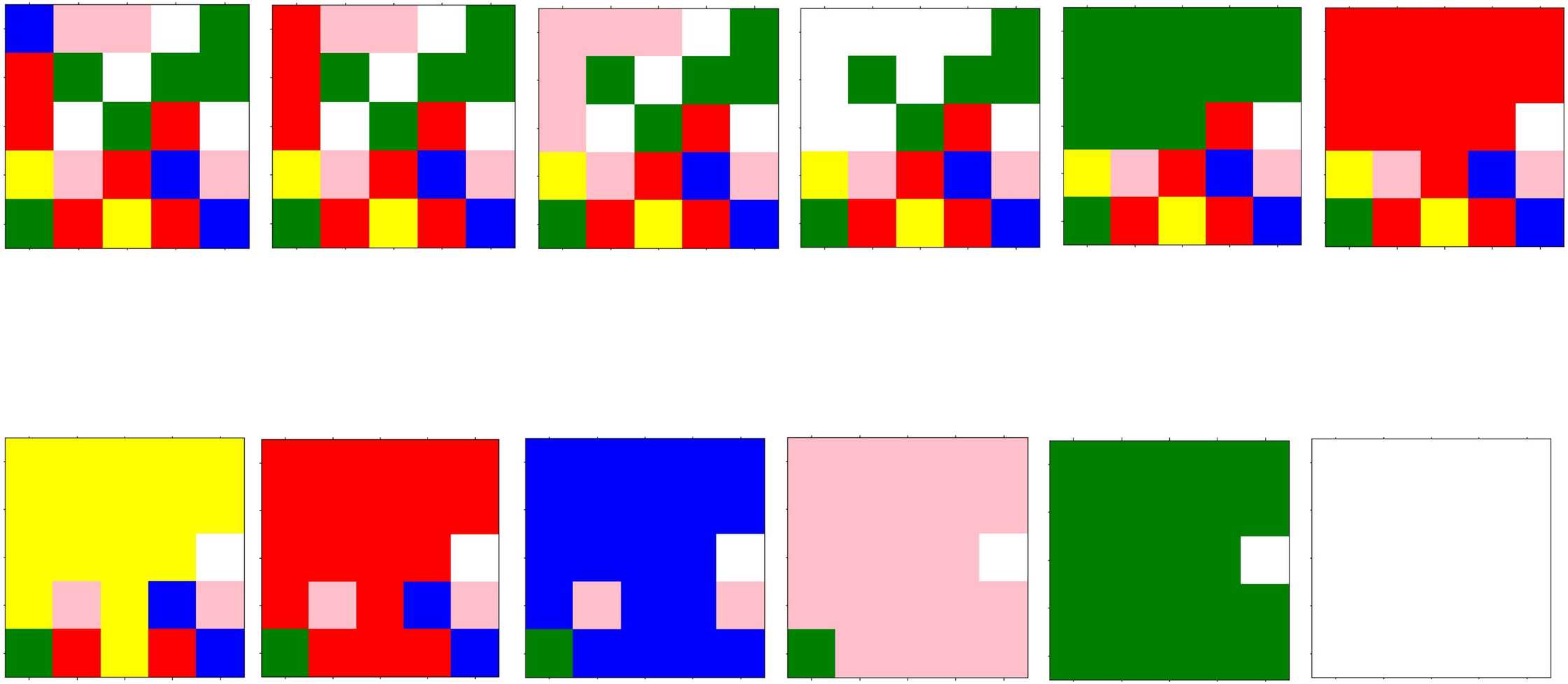











Heurística 3: Most Neighbors

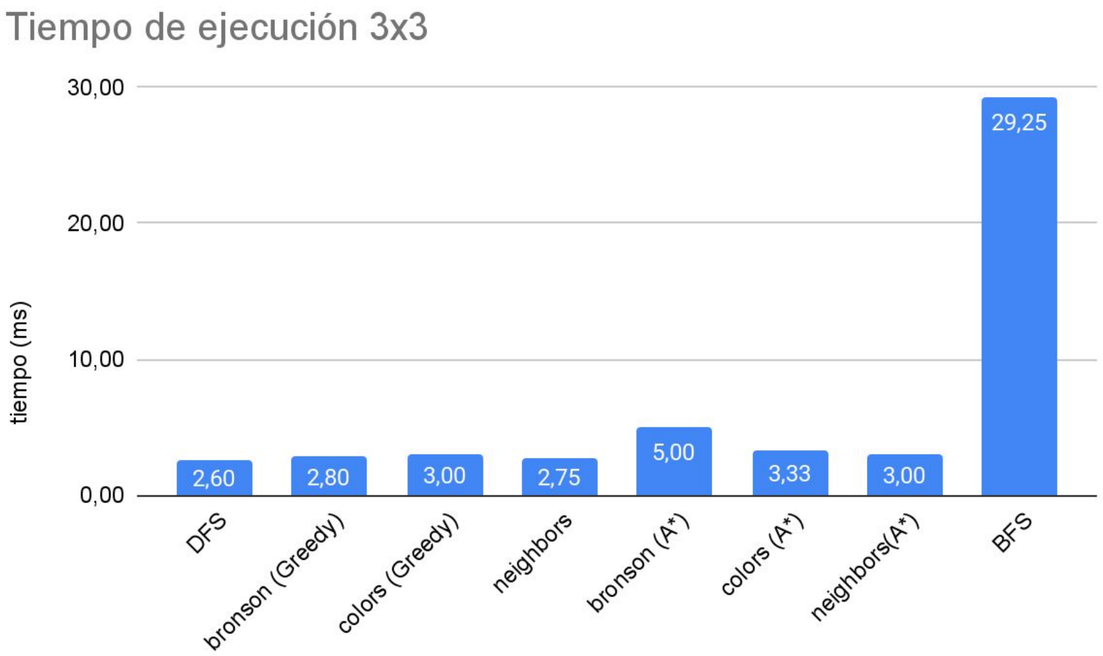
- Consiste en elegir el camino que me permita reducir en mayor medida la cantidad de celdas no controladas.
- No es admisible ya que no me permite estimar la cantidad de turnos restantes requeridos para completar el juego.

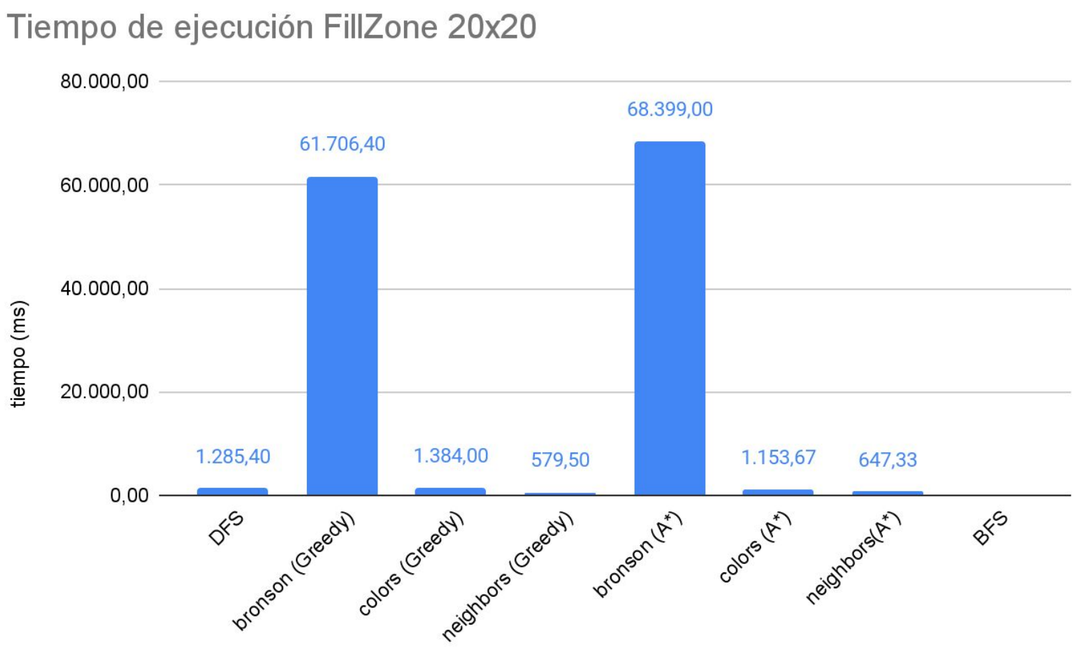


Heuristica 3: Most Neighbors



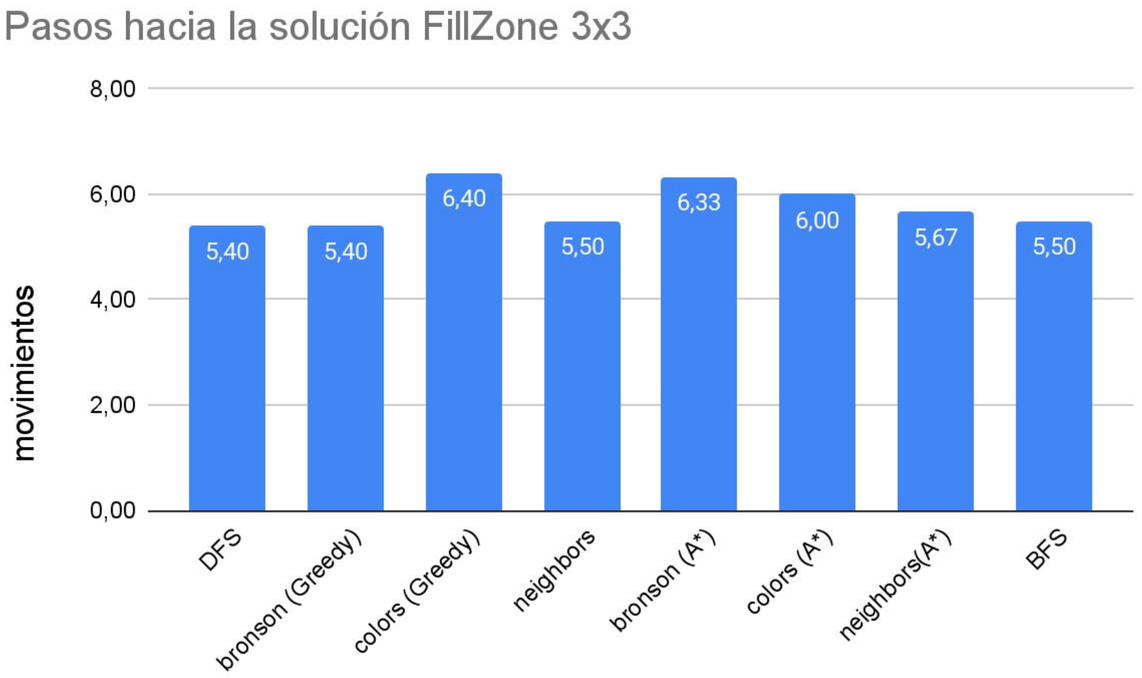
Resultados: tiempo de ejecución

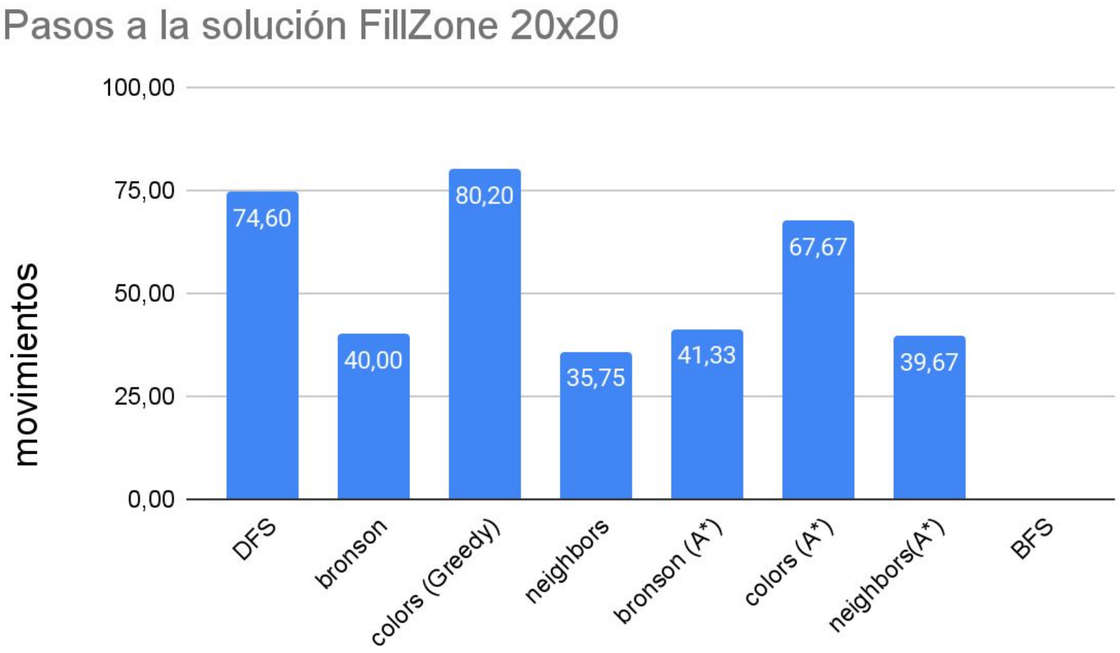




Podemos concluir que el algoritmo con el mayor tiempo de ejecución es el que utiliza la heurística de Bronson.

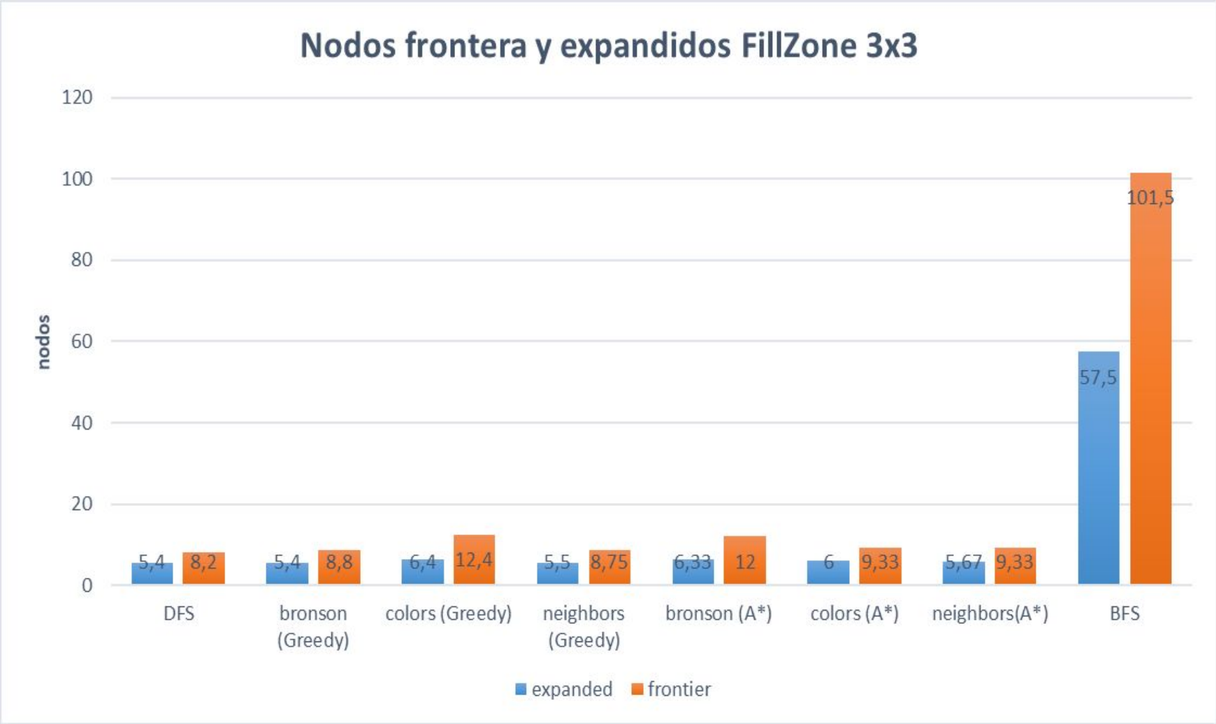
Resultados: pasos

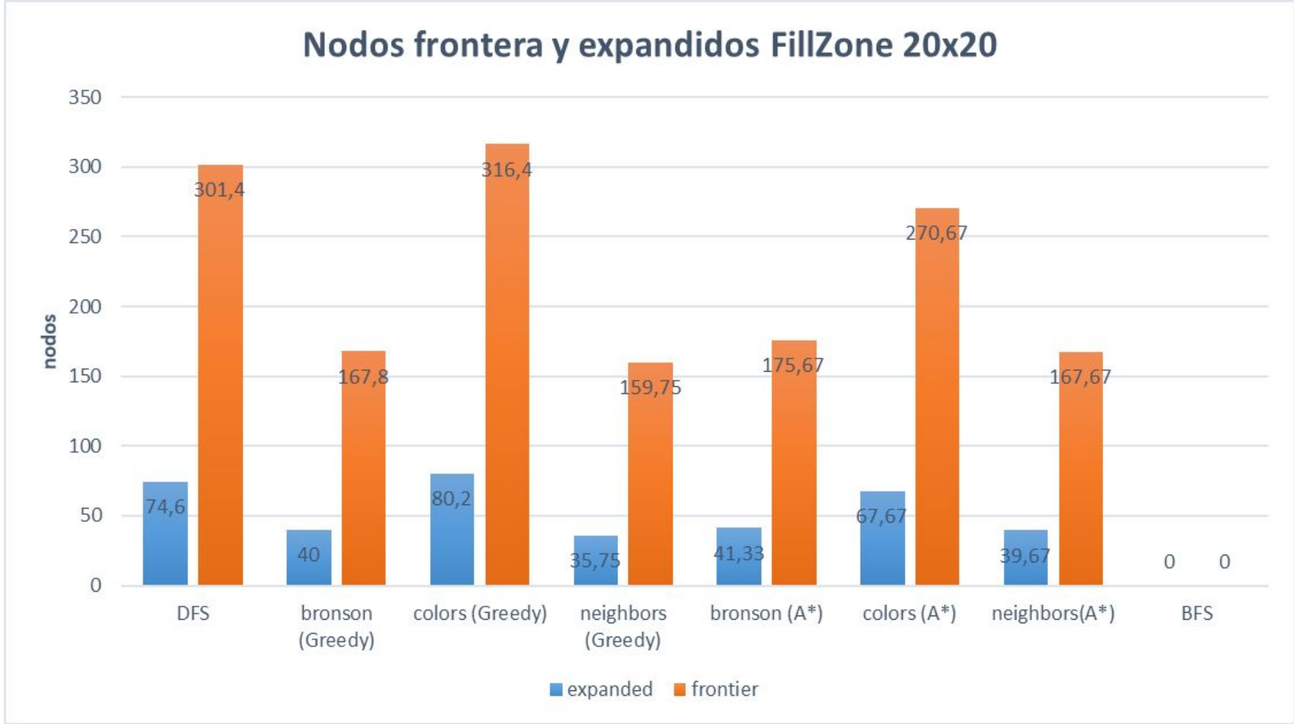




Podemos ver que si bien todos comienzan con una cantidad similar de pasos en tableros pequeños, al aumentar la dimensión los algoritmos que requieren mayor cantidad de pasos son DFS y los que utilizan la heurística Remaining Colors.

Resultados: nodos frontera y expandidos





Se puede observar que al aumentar las dimensiones del tablero los algoritmos DFS y los que utilizan la heurística de remaining colors utilizan mayor cantidad de nodos.

Resultados: Tabla de comparación

	3x3				14X14				20X20			
	t(ms)	steps	expanded	frontier	t (ms)	steps	expanded	frontier	t (ms)	steps	expanded	frontier
DFS	2,60	5,40	5,40	8,20	342,20	42,20	42,20	154,40	1.285,40	74,60	74,60	301,40
bronson (Greedy)	2,80	5,40	5,40	8,80	9.613,40	27,80	27,80	109,00	61.706,40	40,00	40,00	167,80
colors (Greedy)	3,00	6,40	6,40	12,40	219,80	27,20	27,20	112,20	1.384,00	80,20	80,20	316,40
neighbors (Greedy)	2,75	5,50	5,50	8,75	221,75	28,75	28,75	116,75	579,50	35,75	35,75	159,75
bronson (A*)	5,00	6,33	6,33	12,00	9.730,67	30,33	30,33	116,67	68.399,00	41,33	41,33	175,67
colors (A*)	3,33	6,00	6,00	9,33	306,67	38,00	38,00	149,33	1.153,67	67,67	67,67	270,67
neighbors(A*)	3,00	5,67	5,67	9,33	201,00	26,00	26,00	105,00	647,33	39,67	39,67	167,67
BFS	29,25	5,50	57,50	101,50	---	---	---	---	---	---	---	---

Nota: durante la ejecución del algoritmo BFS para tableros de 14x14 y superior se debió suspender la ejecución debido a que este no terminaba

Lado B: Algoritmos Genéticos.

TP N°1

01

Ejercicio 1:

ASCII art

Recreando imágenes

Problema ASCII art

Objetivo

A partir de una imagen cuadrada, representar de la mejor manera posible dicha imagen en un mapa de NxN caracteres ASCII.

Gen

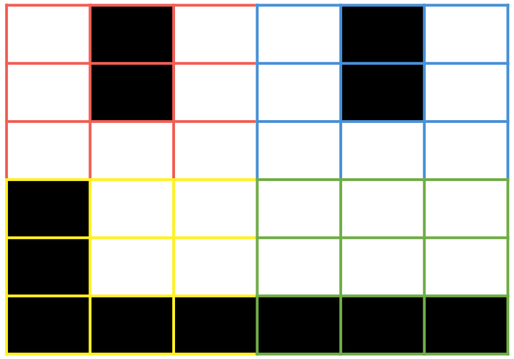
Se define un gen para cada sección cuadrada de MxM pixeles de la imagen.

Alelos

Valor decimal de un caracter ASCII. Hay 255 elementos en la tabla ASCII extendida, pero para representar una imagen los caracteres de control no sirven. Consideremos entonces 223 caracteres ASCII, con el espacio “ “ incluído.

Representación

Imagen cuadrada
de 6x6 pixeles



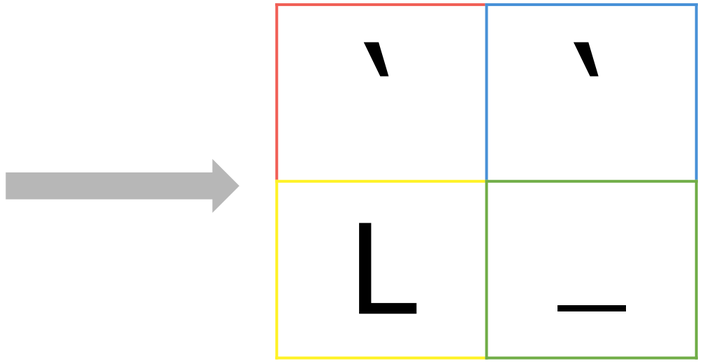
Genotipo

Gen 1	Gen 2	Gen 3	Gen 4
[32-255]	[32-255]	[32-255]	[32-255]

Ejemplo de cromosoma

Gen 1	Gen 2	Gen 3	Gen 4
39	39	76	95

\ \ L _



Cálculo de fitness

M_{imagen} de la imagen input

0	0	0	0	0	0
1	0	1	1	0	1
1	0	1	1	0	1
0	0	0	0	0	0
0	1	1	1	1	0
0	0	0	0	0	0

Ejemplo de cromosoma

Gen 1	Gen 2	Gen 3	Gen 4
84	67	84	79

$$M = \sqrt{\frac{N \cdot N}{\#genes}}$$

Elijo #genes tal que la división devuelva un entero

M_{ascii} Gen 1

0	0	0
1	0	1
1	0	1

M_{ascii} Gen 2

0	0	0
0	1	1
0	0	0

M_{ascii} Gen 3

0	0	0
1	0	1
1	0	1

M_{ascii} Gen 4

0	0	0
0	1	0
0	0	0

<u>Al iniciar el algoritmo</u>

Paso 1: Convierto la imagen de NxN a blanco y negro.

Paso 2: Genero una matriz M_{imagen} de NxN, asignando 0 a los píxeles negros y 1 a los blancos.

<u>Cálculo de fitness</u>

Paso 1: Obtengo la matriz M_{ascii} de MxM, mapa de píxeles del ASCII seleccionado, para cada gen.

Paso 2: Genero la matriz extendida con las matrices M_{ascii} de cada gen, de dimensión NxN

Paso 3: Defino fitness en base a la similaridad de las dos matrices mediante la Norma de Frobenius.

Implementación teórica

Selección por Torneos Determinísticos

Rápido de ejecutar, no usa variables intermedias, ni usa toda la población.

Crossover de dos puntos

Combino secciones de los padres que probablemente ya son buenas aproximaciones a la imagen deseada.

Mutación

Evito el problema de los máximos locales, intercambiando de lugar a los valores de los genes.

02

Ejercicio 2:
Color perfecto

Recreando colores en la paleta del artista

Generación de colores

Objetivo

A partir de una paleta de colores, buscar la mejor forma de mezclar proporciones de los diferentes colores para llegar a un color objetivo.

Estructura del cromosoma

- Un **gen** representa la proporción de un color de la paleta de entrada en el mix.
- Un **cromosoma** es una secuencia de genes con la condición restrictiva de que la suma de los valores de sus genes es uno (1).

Genotipo

Gen 1	Gen 2	Gen 3	Gen 4
[0-1]	[0-1]	[0-1]	[0-1]

Cromosomas

Color objetivo

RGB
[201 0 118]

Paleta de entrada

Rojo	Verde	Azul	Negro
[255 0 0]	[0 255 0]	[0 0 255]	[0 0 0]

Ejemplo de cromosoma resultante

Rojo	Verde	Azul	Negro
0.653	0.0029	0.3432	0.0009

Color resultante

RGB
[166 0 87]

Fitness

Paso 1: Conversión de proporciones a color en coordenadas.

- XYZ siendo RGB o L*a*b*
- α_i siendo la proporción del gen i en el mix.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \sum_{i=1}^{\#genes} \alpha_i \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix}$$

Paso 2: Cálculo de similitud con el color objetivo según distancia Euclídea.

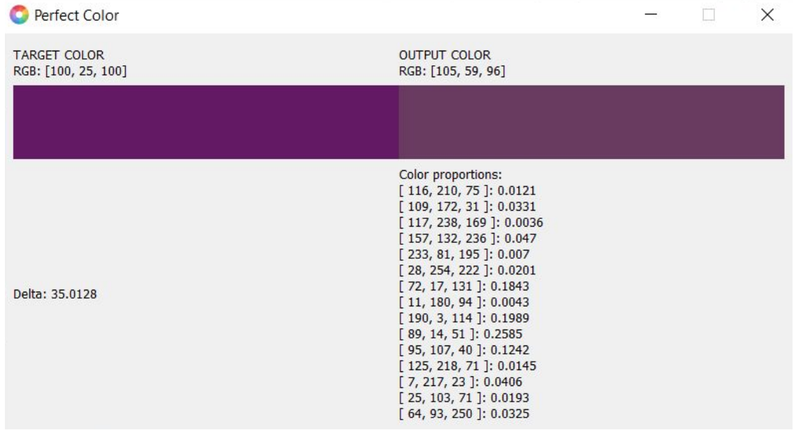
$$\Delta E = \sqrt{(\Delta X)^2 + (\Delta Y)^2 + (\Delta Z)^2}$$

Paso 3: Cálculo de fitness.

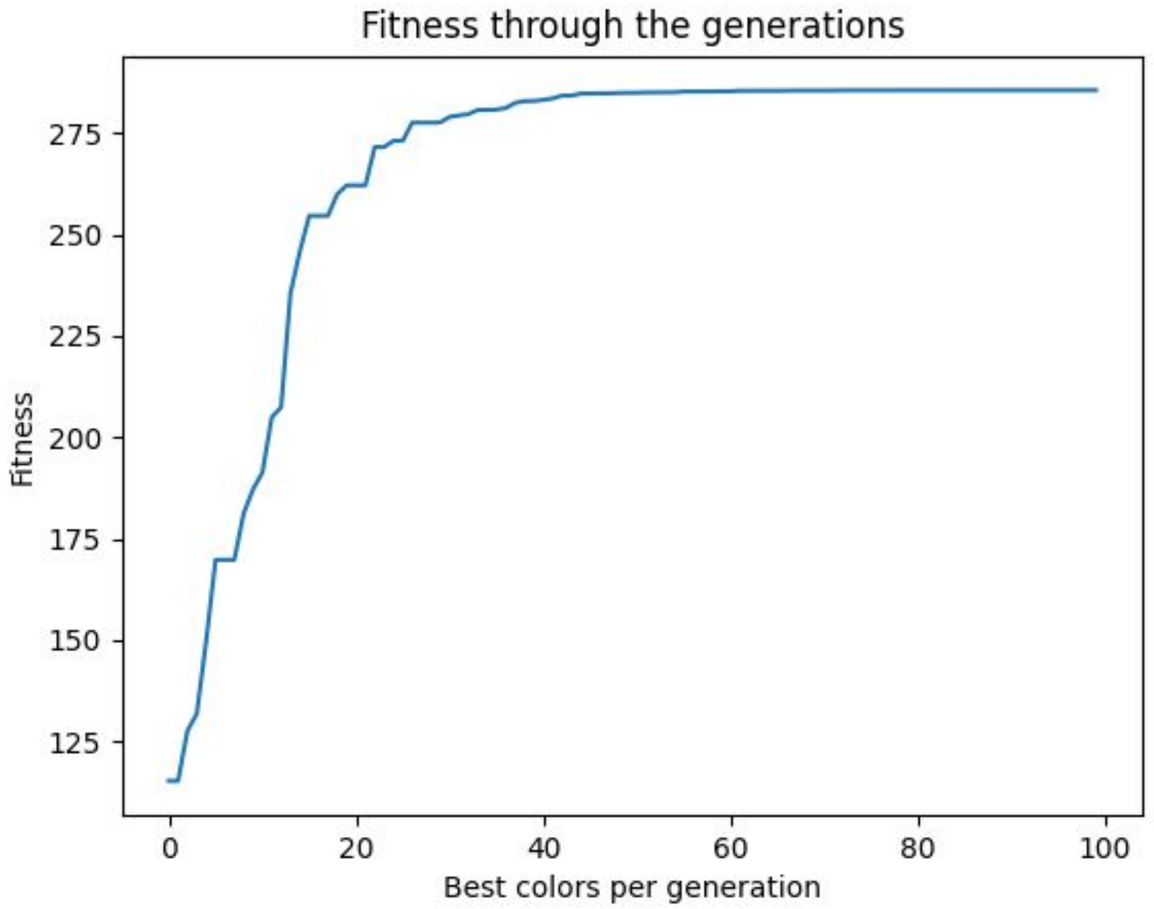
$$fitness = \frac{1}{\Delta E + \varepsilon} \cdot \omega$$

- ε para evitar dividir por cero.
- ω para obtener valores de fitness más grandes cuando se esperan deltas muy grandes.

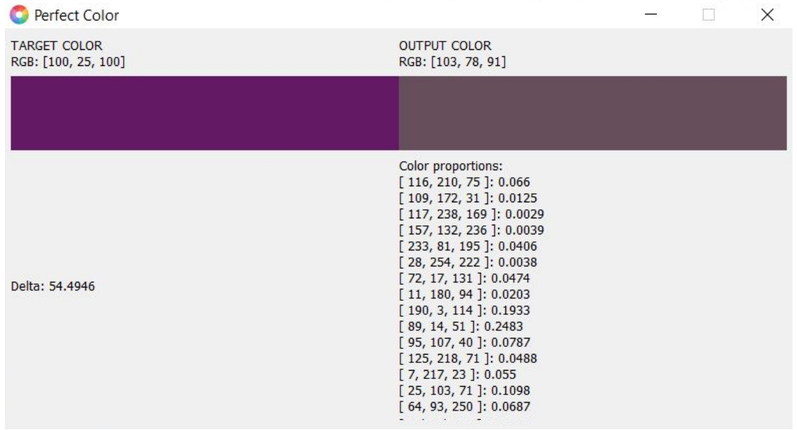
Resultados Elite



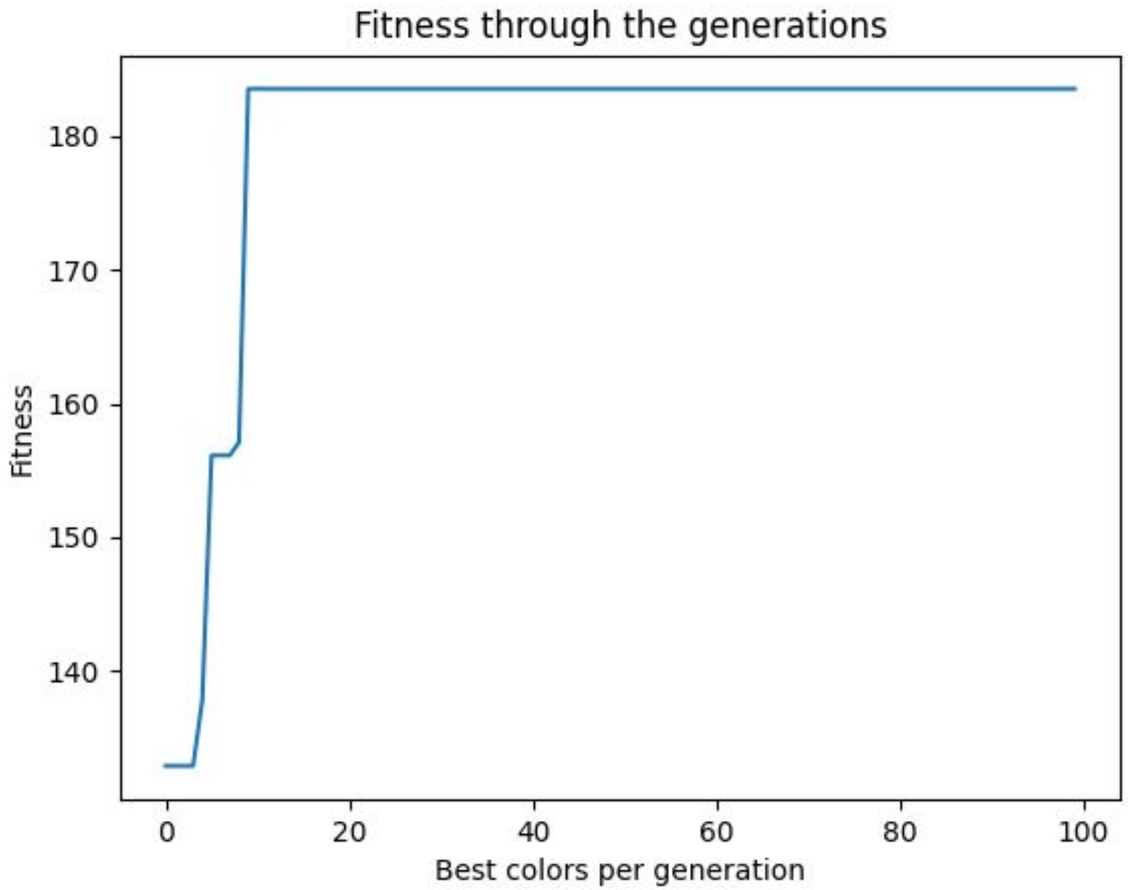
Implementación	Fill All select = Elite
Selección	Elite N = 100, K = 60
Crossover	Cruce de un punto prob = 0.9
Mutación	Completa prob = 0
Corte	200 generaciones



Resultados Elite

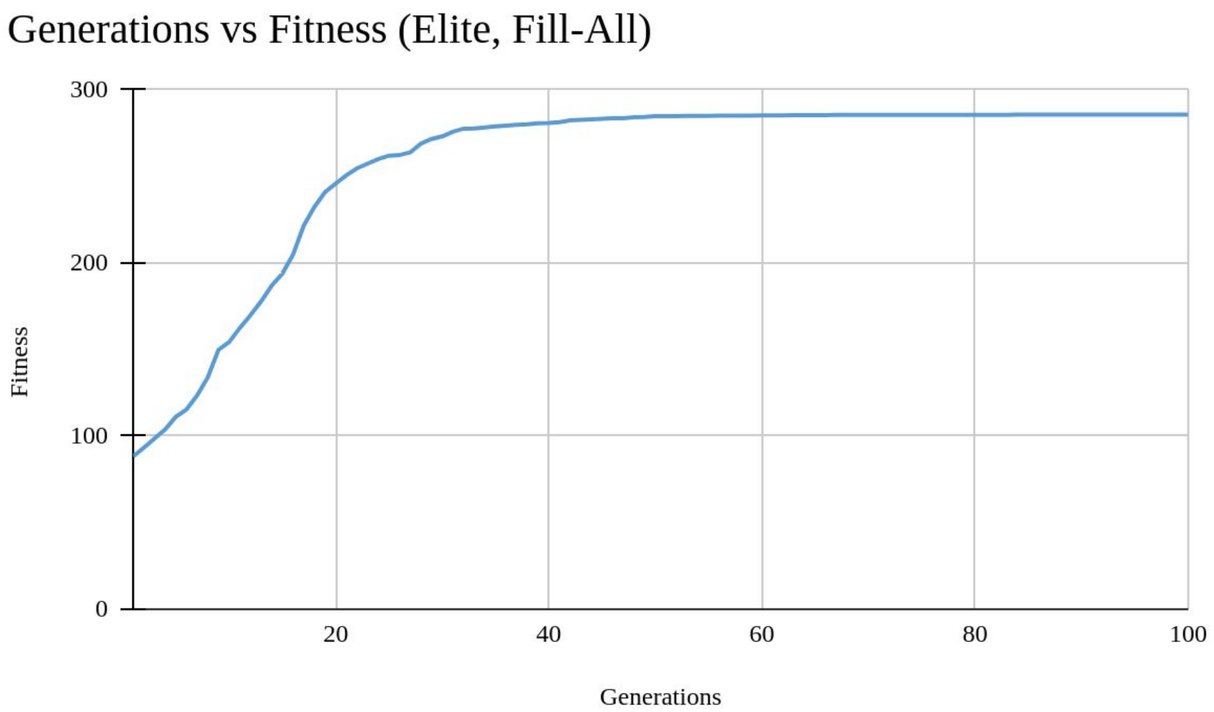


Implementación	Fill All select = Elite
Selección	Elite N = 100, K = 60
Crossover	Cruce de un punto prob = 0.9
Mutación	Completa prob = 0.4
Corte	200 generaciones

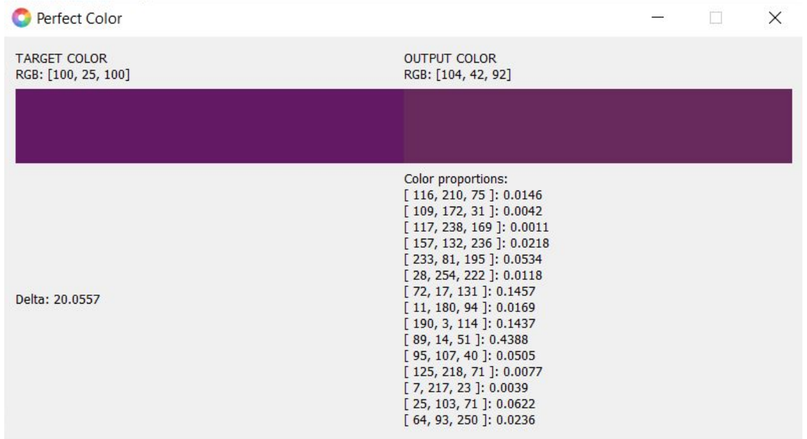


Resultados Elite

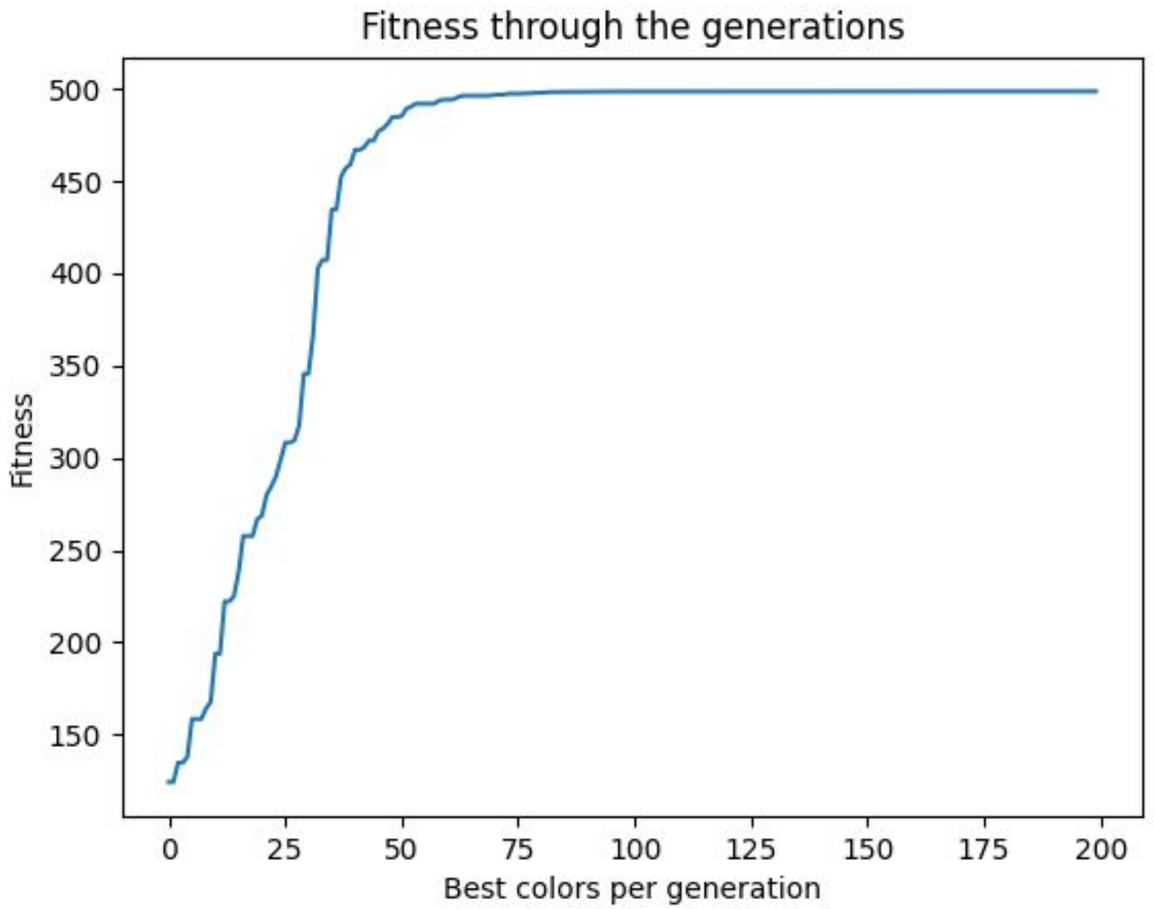
Implementación	Fill All select = Elite
Selección	Elite N = 100, K = 60
Crossover	Cruce de un punto prob = 0.9
Mutación	Completa prob = 0
Corte	200 generaciones



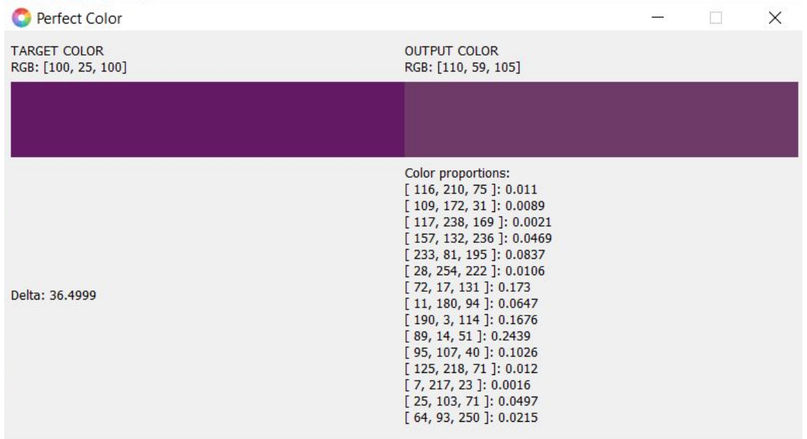
Resultados Ruleta



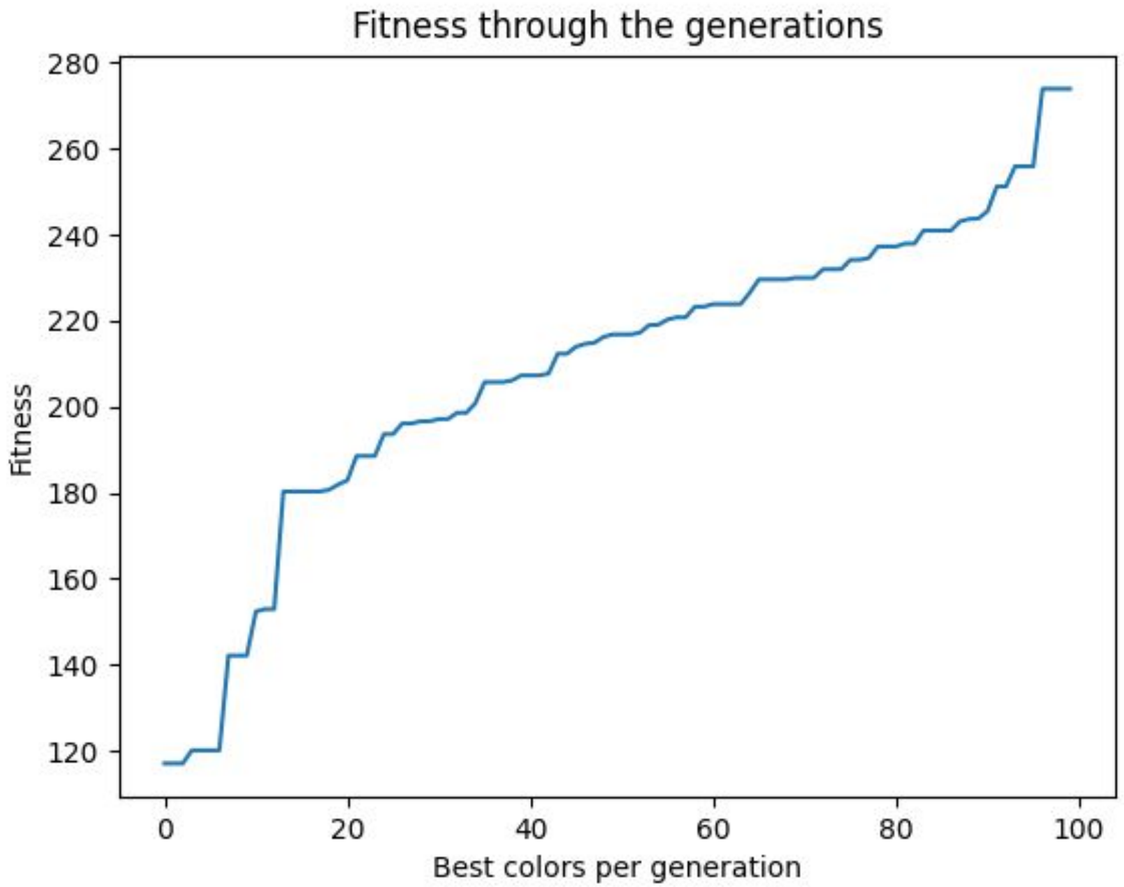
Implementación	Fill All select = Elite
Selección	Ruleta N = 100, K = 60
Crossover	Cruce de un punto prob = 0.9
Mutación	Completa prob = 0
Corte	200 generaciones



Resultados Ruleta

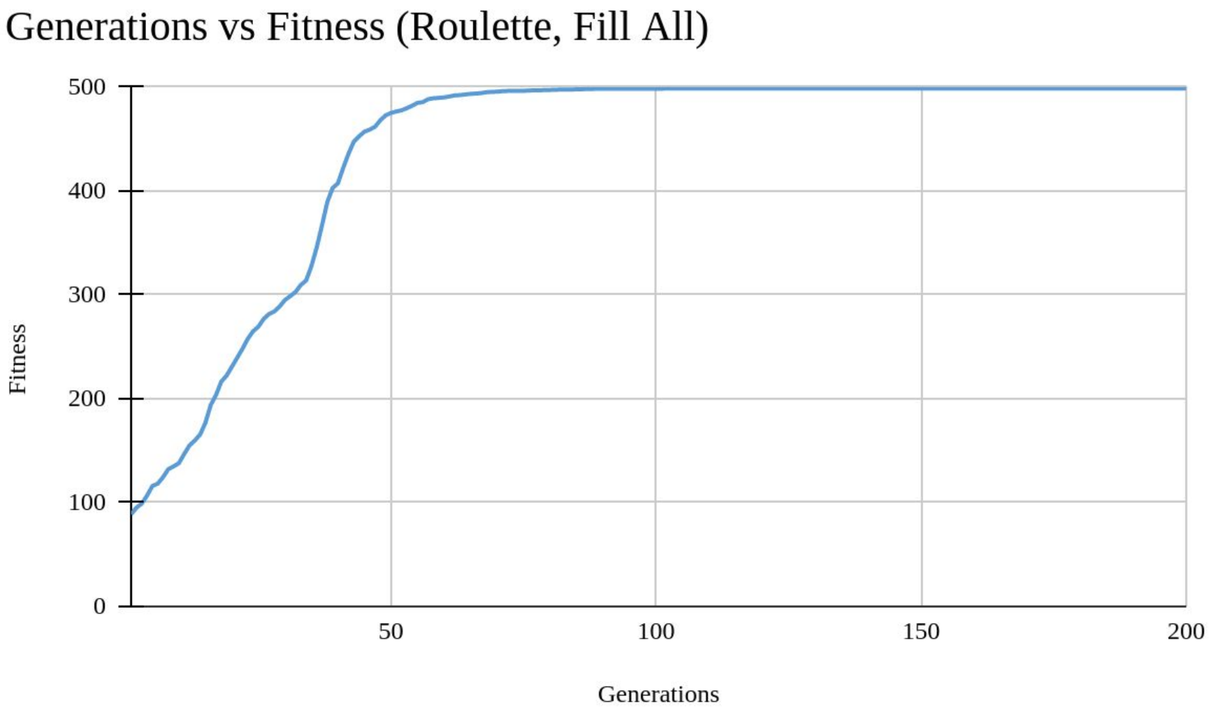


Implementación	Fill All select = Elite
Selección	Ruleta N = 100, K = 60
Crossover	Cruce de un punto prob = 0.9
Mutación	Completa prob = 0.4
Corte	200 generaciones

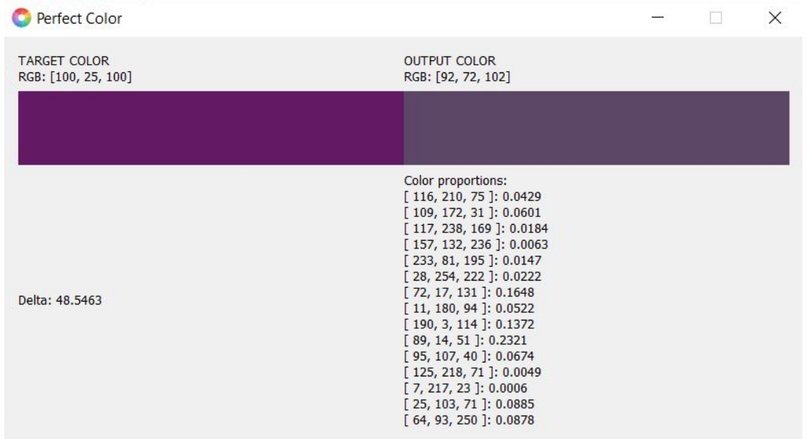


Resultados Ruleta

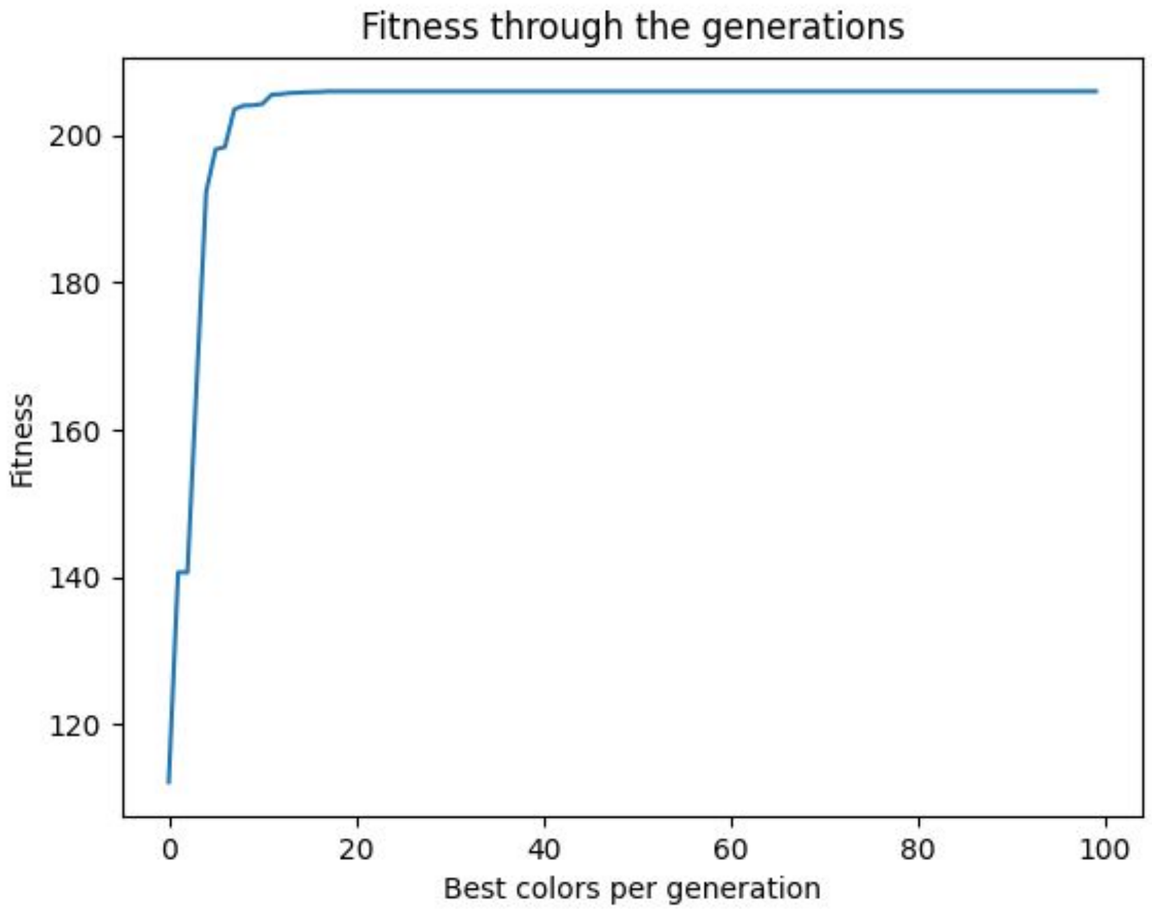
Implementación	Fill All select = Elite
Selección	Ruleta N = 100, K = 60
Crossover	Cruce de un punto prob = 0.9
Mutación	Completa prob = 0
Corte	200 generaciones



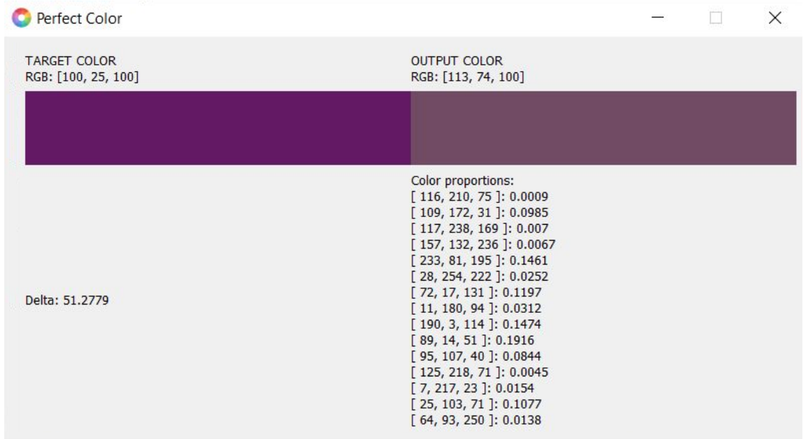
Resultados Torneos



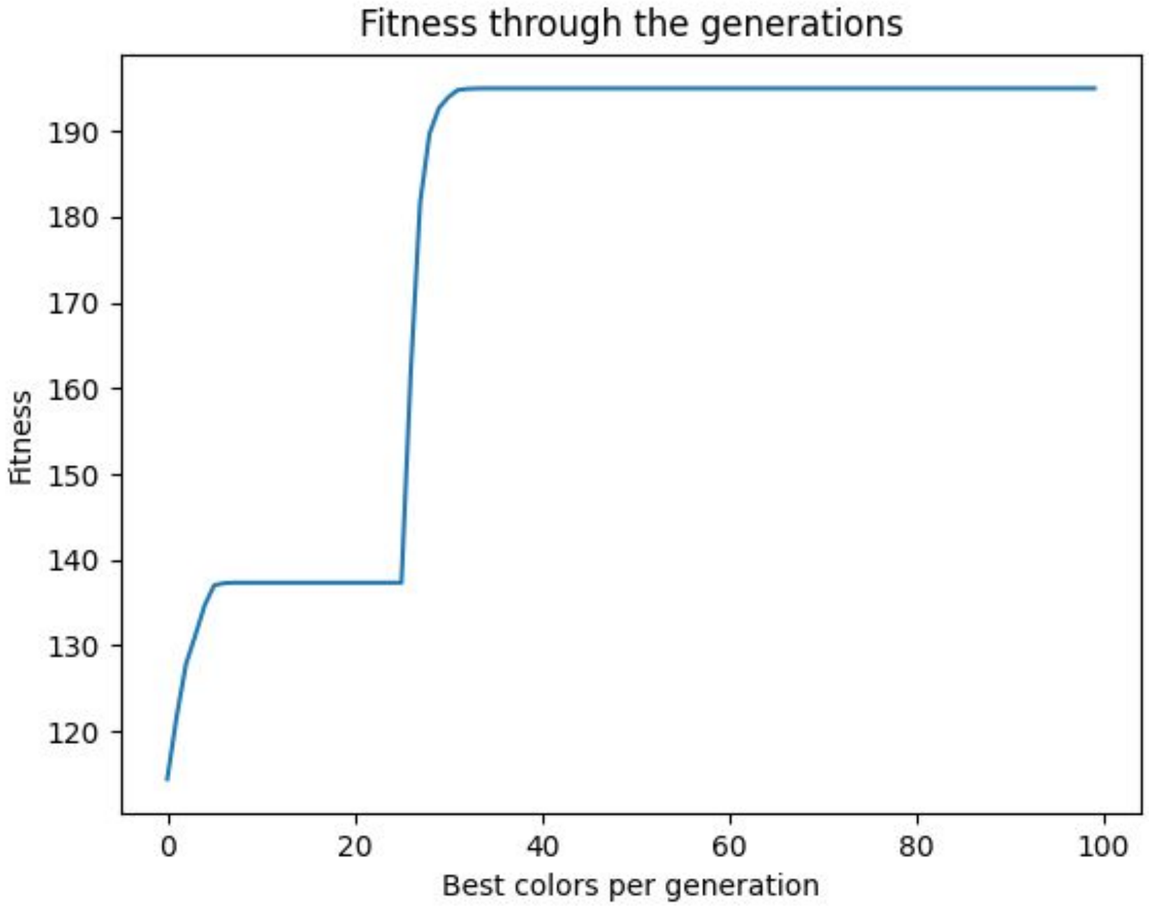
Implementación	Fill All select = Elite
Selección	Torneos determinísticos N = 100, K = 60, M = 15
Crossover	Cruce de un punto prob = 0.9
Mutación	Completa prob = 0.4
Corte	200 generaciones



Resultados Torneos

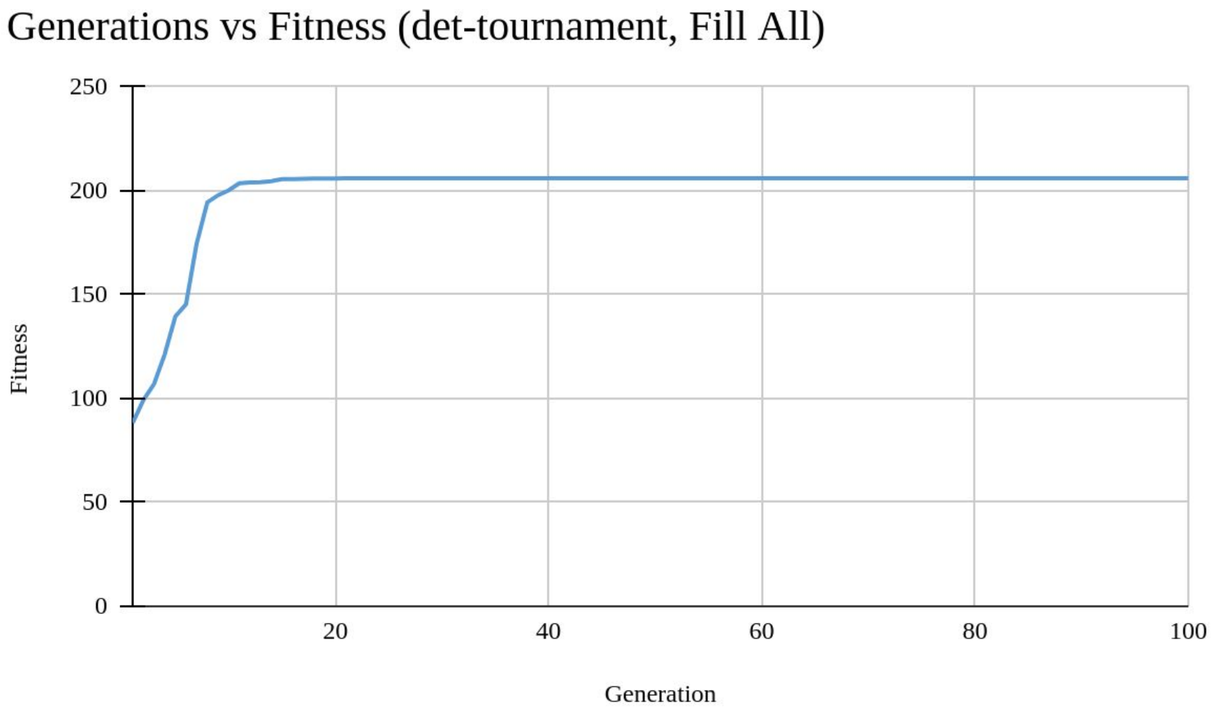


Implementación	Fill All select = Elite
Selección	Torneos determinísticos N = 100, K = 60, M = 100
Crossover	Cruce de un punto prob = 0.9
Mutación	Completa prob = 0.4
Corte	100 generaciones



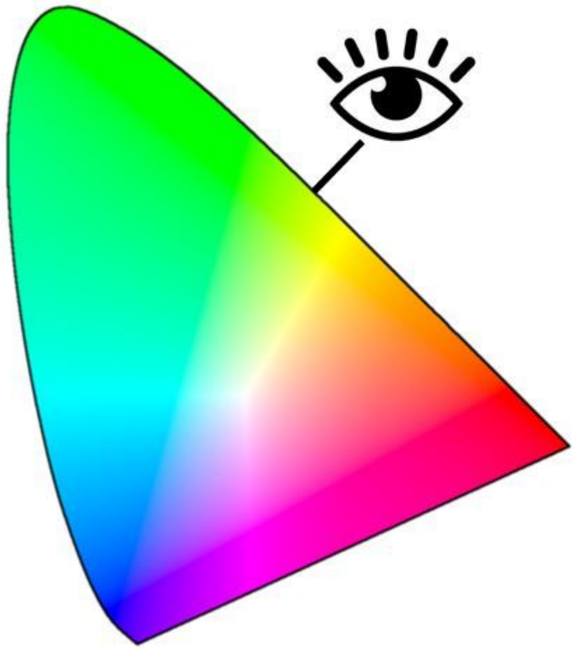
Resultados Torneos

Implementación	Fill All select = Elite
Selección	Torneos determinísticos N = 100, K = 60, M = 15
Crossover	Cruce de un punto prob = 0.9
Mutación	Completa prob = 0.4
Corte	100 generaciones

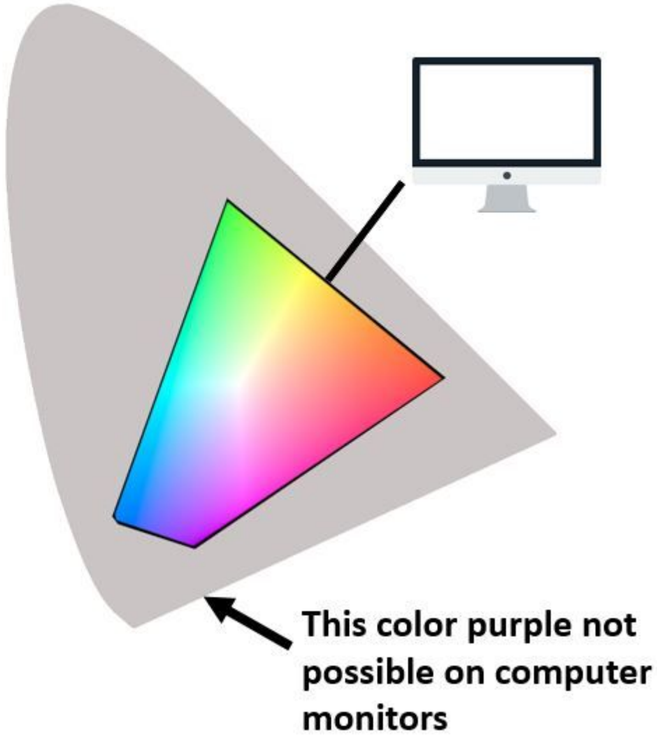


Cambiando el espacio

The CIE ‘LAB’ Gamut:



The RGB Gamut:



Resultados Delta Simulación

	RGB			LAB		
Runs	Elite	Ruleta	Torneo det	Elite	Ruleta	Torneo det
1	67,6254	79,5405	92,9332	31,5510	27,4964	31,5510
2	83,6542	67,0673	91,5424	35,9396	29,6555	35,9396
3	75,7032	77,8790	90,6189	31,0288	29,3947	31,0288
4	90,3859	77,9681	84,9088	31,6845	28,9535	31,6845
5	79,7741	79,3761	92,5812	30,1693	29,2661	30,1693
6	75,5850	73,8359	96,5811	33,8687	28,5981	33,8687
7	78,1159	72,9379	95,6577	31,2690	27,4989	31,2690
8	85,9480	74,9291	72,8837	33,2815	27,6775	33,2815
9	85,2173	71,3700	96,0979	31,2213	28,6024	31,2213
10	68,7095	74,4885	71,0462	32,1052	29,6931	32,1052
Promedio	79,0718	74,9393	88,4851	32,2119	28,6836	32,2119

A considerar