Sistemas de Inteligencia Artificial

Franco Arolfo Jorge Mozzino Francisco Pagliaricci

25 de marzo de 2013

# Métodos de búsqueda no informados e informados

# 1. Objetivo

Se propone con este trabajo implementar un Sistema de Producción que será utilizado para resolver el problema 'Fill Zone'. A partir de un motor de inferencia programado en Java, se realizaron las respectivas modificaciones para que pudiera trabajar con estrategias de búsqueda no informadas (depth first, breadth first y profundización iterativa) e informadas (Greedy Search y A*).

# 2. Descripción del juego: 'Fill Zone'

Se le brinda al jugador una grilla de NxN celdas, donde cada celda esta pintada con un color. Así mismo, la cantidad de colores a utilizar esta predefinida, en nuestro caso usaremos 6. Al inicio del juego, dicha grilla cuenta con sus celdas dispersas en forma aleatoria (ver figura 1).

El objetivo del juego es que el tablero finalice con todas sus celdas del mismo color. Para ello, el jugador debe ir cubriendo el tablero de la siguiente manera: por cada turno, debe elegir uno de entre todos los colores disponibles y se pintara a todas las celdas del mismo color que la celda del margen superior izquierdo que sean adyacentes a ella. Es requerimiento además que el jugador complete el tablero con un numero máximo de turnos.

Por ejemplo, supongamos que se comienza con el tablero de la figura 1. Eligiendo el color verde en el próximo turno, se obtendría el tablero presentado en la figura 2.

# 3. Desarrollo del proyecto

## 3.1. Preliminares

Decidimos reutilizar el motor de búsqueda desarrollado en Java propuesto por la cátedra, modificando lo necesario para nuestro problema.

Las modificaciones realizadas consistieron en generar subclases que implementen los métodos getNode() y setNode(). Dichos métodos varían según si es un algoritmo no informado o informado, y entre los informados, en donde qué lugar de la cola se sitúan los nodos que se visitan.

## 3.2. Modelado del problema

Presentaremos a continuación nuestro modelo del problema 'Fill Zone':

*Estado:* Modelaremos un estado como un 'board', que consta de una matriz de enteros, y otro entero 'movesLeft', que indica la cantidad de movimientos que el jugador esta permitido a realizar. En el 'board' decidimos modelar cada color como un entero.

*Estado inicial:* Consta de un 'board' genreado de manera aleatoria, con el 'movesLeft' inicializado con la cantidad de movimientos que se pueden hacer.

*Reglas:* Presentamos las reglas que definimos para el juego:

1. El jugador elige color Verde.
2. El jugador elige color Azul.
3. El jugador elige color Magenta.
4. El jugador elige color Blanco.
5. El jugador elige color Amarillo.
6. El jugador elige color Rojo.

En todos los casos, el tablero cambia según lo ya explicado.

*Función de costos:* Cada regla consume 1 movimiento. De esa forma, teniendo en cuenta que la función heurística busca estimar la cantidad de movimientos faltantes para resolver el tablero, es lógico que la función de costo devuelva 1 para todas las reglas.

*Test objetivo:* Un estado es final si todas sus celdas son del mismo color y aún quedan movientos por ejecutar, o es el último permitido ('movesLeft' ¿= 0).

## 3.3. Búsqueda no informada: DFS, BFS y ID

Los algoritmos ya definidos (BFS, DFS y ID) se dice que son de 'búsqueda NO informada' por que no cuentan con información alguna sobre el problema, más que su propia definición. Éstos algoritmos ya son conocidos y no ahondaremos sobre ellos en este informe.

Vamos a adelantar, como luego se verá en las tablas, que el DFS es el único algoritmo aplicable a este juego, ya que, debido a la ramificación del árbol de búsqueda y los profundos niveles que éste alcanza en general, desarrollar todo el árbol (como lo hacen de alguna manera tanto BFS como profundización iterativa) requiere mucho tiempo.

## 3.4. Búsqueda informada: Greedy search y A*

Los algoritmos de búsqueda informados, a diferencia de los no informados, cuentan con información del problema en forma de heurísticas. Las heurísticas son una estimación de cuan lejos se está de alcanzar la solución.

La diferencia entre Greedy search y A* radica en que A* contempla el costo (computado como la suma de los costos de las reglas aplicadas en el camino) y lo suma al valor heurístico del estado, mientras que Greedy search sólo tiene en cuenta el valor heurístico.

Desarrollamos 4 heurísticas: 3 de manera incremental y la otra basada en una estrategia de juego.

### 3.4.1. Procedimiento incremental

Comenzamos proponiendo una heurística $H1$ simple que consiste en decir que los turnos estimados que faltan equivalen a la cantidad de celdas que faltan cubrir.

$$H1 = CeldasTotales - CeldasCubiertas$$

Se puede ver, fácilmente, que este es el tope máximo de turnos que faltan para cubrir todo el tablero. Claramente es una heurística no admisible. Basta considerar el caso en que faltan 2 celdas del mismo color en el tablero. La cantidad de turnos es 1, pero esta heurística devolvera 2 (ver figura 3).

Basándonos en este caso en el que falla, surgió la idea de bajar esta cota.

Definimos *isla* como el conjunto maximal de celdas de un mismo color para el cual existe un camino de ese color para todo par de celdas en el conjunto.

De la misma manera, definimos *isla principal* como aquella isla que contiene la celda (0, 0).

Entonces, definimos la heurística $H2$ como la cantidad de islas que hay en el tablero, sin contar la isla principal.

$$H2 = CantidadDeIslas - 1$$

De forma análoga que con la primera heurística, basta con tomar un tablero en que haya 2 islas del mismo color, separadas por la isla principal para demostrar que esta heurística no es admisible. Se puede ver que con el tablero mencionado, toma 1 movimiento cubrir el tablero, pero la heurística devuelve 2 (ver figura 4).

Finalmente, nuestro último paso es una heuristica en la que decidimos ver el tablero como un mapa, y obtener el grafo equivalente (ver figuras 6 y 5. Luego, definimos que 2 nodos son *k-vecinos*, si están a distancia k como mínimo entre ellos.

La heurística entonces, consiste en sacar el máximo $k$ entre nodos de cada color y sumar las k.

Las islas *singulares* son aquellas islas para las cuales no hay otra isla con el mismo color. Para la suma anterior, se las considera con k = 1, y si la isla principal es sigular no se toma en cuenta en la suma.

Esta heurística, al igual que las otras es no admisible. El tablero y la demostración, sin embargo, son muy complicados y largos.

### 3.4.2. Adaptación de estrategia

Una popular estrategia para resolver este juego, consiste en ir moviéndose hacia las esquinas, siendo más importante buscar la esquina inferior derecha. Debido a esto, buscamos deducir alguna fórmula que nos ayudara a estimar la distancia a la solución.

La fórmula resultante fue

$$H4 = (diagSuperior + diagInferior + 2 * diagPrincipal) * 5 + CeldasNoCubiertas$$

Donde $diagSuperior$ es la distancia a la esquina superior derecha, $diagInferior$ esla distancia a la esquina inferior izquierda y $diagPrincipal$ es la distancia a la esquina inferior derecha

El hecho de sumar las celdas no cubiertas demuestra que es no admisible (referirse a la demostración de H1).

Notar, también, que no se están estimando turnos sino distancias.

## 3.5. Conclusiones y trabajo futuro

Como se puede observar en la tabla 1 la heurística que a priori creíamos como la peor, está bastante pareja en relación al resto teniendo en cuenta tiempos de ejecución y nodos explotados. Al principio nos llamó mucho la atención, por lo que decidimos seguir paso a paso la ejecución del programa con las heurísticas y notamos que al principio ambas seguían un camino común pero en un punto tomaban caminos distintos. Lo que ocurría es que la heurística simple (por fortuna, si se quiere), tomaba el camino más óptimo, mientras que las otras no. Entonces la heurística básica resolvía el problema rápidamente, mientras que las otras se iban por ramas poco óptimas. Como, además, las heurísticas bajaban más de un turno entre nivel y nivel, había que realizar muchos niveles hacia abajo para concluir que no se había tomado un camino adecuado. Esto demuestra que estábamos equivocados al creer que heurísticas mejores son mejores en todos los casos y que muchas veces puede depender de la suerte (i.e. qué nodo se está explotando primero).

También concluímos, como era de esperarse, que los algoritmos BFS y Profundización iterativa no son eficientes para este problema, ya que buscan la mejor solución y van desarrollando todo el árbol que en este problema adquiere dimensiones enormes. Por este motivo, algoritmos basandos en profundizar, como DFS o los informados, son más adecuados ya que encuentran una solución más rápidamente y, para este problema en particular, no se está buscando optimizar los movimientos realizados sino llegar a un tablero solución.

Se presentaron diversas heurísticas de forma .evolutiva", es decir, se empezó con una heurística básica y se fue refinando y mejorando. Por cuestiones de tiempo, resultó imposible seguir investigando hasta poder lograr ganar el juego (en el sentido de encontrar una heurística que prediga con exactitud la cantidad de movimientos faltantes. Proponemos a continuación algunas ideas que el lector puede utilizar si le interesa adentrarse un poco más en el tema.

1. Nótese que los vértices de corte en el grafo de islas indican islas para las cuales sí o sí hay que gastar un movimiento (i.e. no se puede esperar para cubrir varias islas de ese color). Sería interesante estudiar cómo se puede aprovechar esto para refinar la heurística.

2. Así como en H3 se va tomando los máximos de los valores de $k$, se podría refinar esto buscando algún tipo de transitividad y promediando distintos valores de $k$ para el mismo color.

# 4. Anexo figuras y tablas

Figura 1: Estado inicial.

Figura 2: Estado inicial.

Figura 3: Prueba de que h1 no es admisible

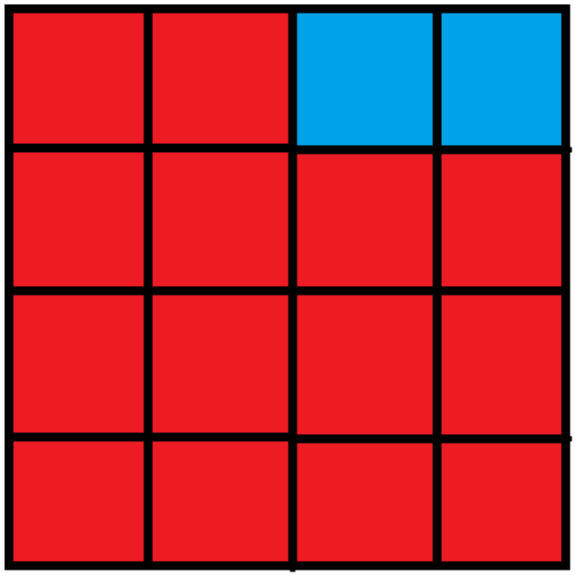
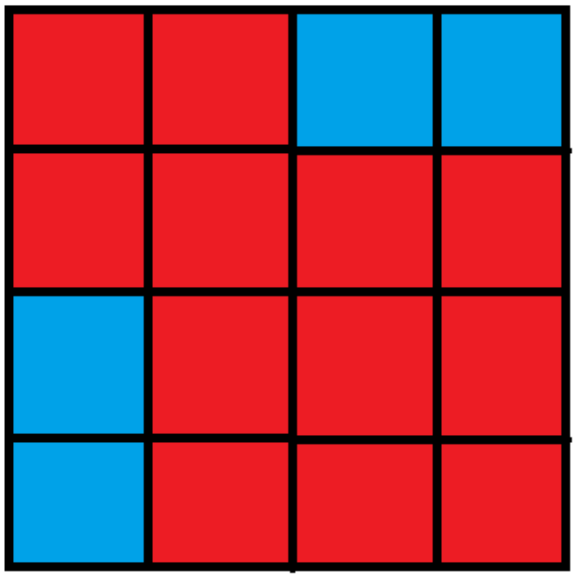


Figura 4: Prueba de que h2 no es admisible



| H1 | H2 | H4 | H3 |
|---|---|---|---|
| T:432 N:164 | T:220 N:147 | T:211 N:125 | T:1247 N:171 |
| T:233 N:145 | T:263 N:156 | T:353 N:680 | T:1993 N:218 |
| T:114 N:100 | T:132 N:135 | T:142 N:135 | T:677 N:163 |
| T:228 N:145 | T:290 N:420 | T:293 N:145 | T:1455 N:168 |
| T:482 N:1615 | T:254 N:147 | T:264 N:165 | T:1282 N:167 |

Cuadro 1: Tabla de que mide tiempos y nodos explotados en un tablero de 14x14 con 30 movimientos máximos. T indica el tiempo en milisegundos y N la cantidad de nodos explotados. Se muestran los resultados para 5 tableros

| | H1 | H2 | H4 | H3 |
|---|---|---|---|---|
| WINS | 3 | 1 | 1 | 0 |
| TIME AVG | 297.8 | 231.8 | 252.6 | 1330.8 |
| NODE AVG | 433.8 | 201.0 | 250.0 | 177.4 |

Cuadro 2: Tabla que muestra los promedios y cantidad de victorias (en tiempo) de la tabla 1

| H1 | H2 | H4 | H3 |
|---|---|---|---|
| DOMINATED | ISLANDS | CORNERS | GRAPH |
| T:343 N:164 | T:88484 N:35156 | T:198 N:125 | T:1242 N:171 |
| T:233 N:141 | T:313 N:629 | T:4567 N:9536 | T:1997 N:218 |
| T:104 N:100 | T:138 N:135 | T:143 N:135 | T:672 N:163 |
| T:230 N:137 | T:43352 N:23634 | T:398 N:1089 | T:1472 N:168 |
| T:15335 N:14867 | T:937 N:3471 | T:260 N:151 | T:1277 N:167 |

Cuadro 3: Tabla de que mide tiempos y nodos explotados en un tablero de 14x14 con 28 movimientos máximos. T indica el tiempo en milisegundos y N la cantidad de nodos explotados. Se muestran los resultados para 5 tableros.

| | H1 | H2 | H4 | H3 |
|---|---|---|---|---|
| WINS | 3 | 0 | 2 | 0 |
| TIME AVG | 3249.0 | 26644.8 | 1113.2 | 1332.0 |
| NODE AVG | 3081.8 | 12605.0 | 2207.2 | 177.4 |

Cuadro 4: Tabla que muestra los promedios y cantidad de victorias (en tiempo) de la tabla 3. Notar como la única heurística que mantiene estabilidad con respecto a la tabla 2 es H3.

Figura 5: Un tablero genérico

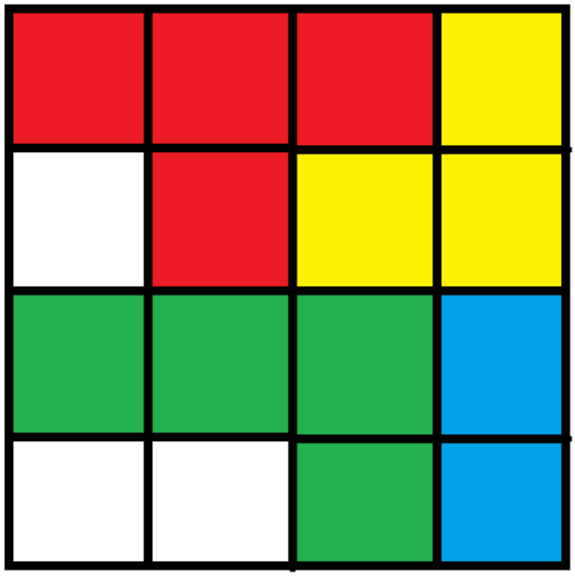
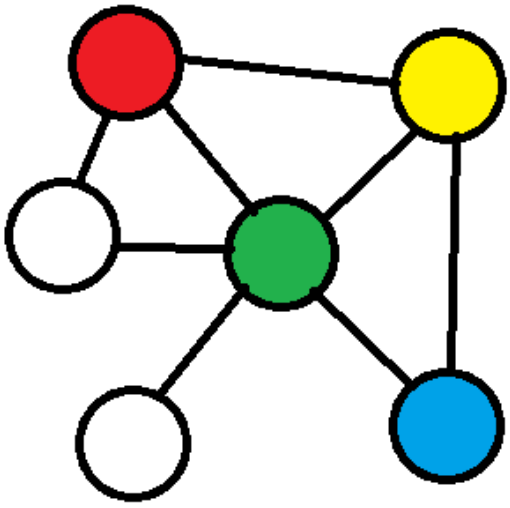


Figura 6: Grafo que representa el tablero de la figura 5



| H1 | H2 | H4 | H3 |
|---|---|---|---|
| T:323 N:1181 | T:124 N:1531 | T:15 N:181 | T:153 N:859 |
| T:34 N:800 | T:10 N:287 | T:138 N:2036 | T:41 N:257 |
| T:42 N:926 | T:54 N:1112 | T:5 N:118 | T:59 N:572 |
| T:14 N:293 | T:211 N:2570 | T:6137 N:17211 | T:67 N:660 |
| BFS | DFS | ID | |
| T:40043 N:28244 | T:27 N:669 | T:418 N:45606 | |
| T:568 N:4421 | T:50 N:1286 | T:220 N:7929 | |
| T:63090 N:34398 | T:920 N:7007 | T:39169 N:44479 | |
| T:12805 N:18718 | T:6240 N:17575 | T:14921 N:38456 | |

Cuadro 5: Tabla de que mide tiempos y nodos explotados en un tablero de 6x6 con 10 movimientos máximos. T indica el tiempo en milisegundos y N la cantidad de nodos explotados. Se muestran los resultados para 4 tableros.

| | H1 | H2 | H4 | H3 | BFS | DFS | ID |
|---|---|---|---|---|---|---|---|
| WINS | 1 | 1 | 2 | 0 | 0 | 0 | 0 |
| TIME AVG | 103.25 | 99.75 | 1573.75 | 80.0 | 29126.5 | 1809.25 | 13682.0 |
| NODE AVG | 800.0 | 1375.0 | 4886.5 | 587.0 | 21445.25 | 1809.25 | 13682.0 |

Cuadro 6: Tabla que muestra los promedios y cantidad de victorias (en tiempo) de la tabla 5. Notar como las búsquedas informadas son claramente más rápidas que las no informadas.