SIA: Trabajo prático 1
Métodos de búsqueda informados y no informados

Agustin Marseillan - 50134, Federico Ramundo - 51596, Conrado Mader Blanco - 51270

Entrega Final - Informe

***Resumen*—El presente informe busca analizar y comparar distintas estrategias de búsqueda de la solución de un problema en particular (DeepTrip) haciendo uso de un motor de inferencias, como así también evaluar las mejoras obtenidas mediante la aplicación de heurísticas.**

***Palabras clave*—DeepTrip, A*, Greedy, DFS, BFS, IDDFS, General Problem Solver, search strategy**

I. Introducción

EL juego DeepTrip consta de un tablero de diez filas por ocho columnas, en el que cada celda puede tener una ficha de color (rojo, naranja, azul, verde, violeta, amarillo) o estar vacío.

El objetivo es agrupar tres o mas fichas del mismo color, haciendo asi que exploten y ganando puntos. Cuando un grupo de fichas explota, en su lugar pasa a haber vacío. Se dice que un par de fichas están agrupadas si están una arriba de la otra, o una al lado de la otra.

El juego cuenta con gravedad invertida, lo que quiere decir que las fichas "flotan": si hay un espacio vacío arriba de una ficha entonces ésta se moverá hacia esa dirección hasta encontrar otra ficha o el fin del tablero. Esta propiedad permite que puedan ocurrir varias explosiones de fichas encadenadas.

Para poder agrupar las fichas, el jugador debe mover las filas, corriendo todas las fichas en ella a la vez, haciendo que cuando una ficha se salga del tablero por un lado, aparezca por el otro.

En el juego original, se cuenta con un tiempo determinado para hacer la mayor cantidad de puntos posibles, y cada vez que se explotan fichas, además de ganar puntos, se gana tiempo extra para seguir jugando. A su vez, cuando una fila se vacía completamente, se genera una nueva fila llena en la parte superior del tablero, haciendo que éste nunca tenga filas vacías. Para este trabajo, dado que el juego no cuenta con un estado de *goal*, se simplificó la lógica original eliminando la regeneración de filas, sacando el tiempo y los puntos, y estableciendo el *goal* como el tablero en el cual no quedan fichas (se explotaron todas). También se experimentó con tableros de tamaños diversos, y con la cantidad de dististos colores de las fichas.

II. Estados del problema

A. Estado inicial

El estado inicial es un tablero lleno en el cual no hay tres o mas fichas adyacentes del mismo color. Para que el tablero sea válido debe existir una solución, es decir, una combinación de movimientos que lleve a vaciar el tablero.

B. Estado final

El estado final es un tablero sin fichas, es decir en el cual todas sus celdas están vacias.

C. Estado intermedio

Un estado intermedio es todo aquel que no es ni inicial ni final, es decir, tiene algunas fichas y algunas celdas vacías.

III. Modelado del problema

El tablero de juego se modeló en la clase Board como una matriz de enteros, en el cual cada número representa un color distinto, siendo el cero el vacío. Para agilizar cuentas, además esta clase cuenta con información extra, como la cantidad de fichas que quedan de cada color o la cantidad de movimientos que se realizaron desde el tablero inicial hasta llegar al actual.

IV. Reglas

Dado un tablero de nxm el conjunto de reglas posibles son:

Para la fila n, correr m cantidad de fichas, siendo m un numero mayor a cero.

El número m debe ser mayor a cero dado que de ser igual, no se estaía realizando movimiento alguno, y menor a la cantidad de columnas del tablero, ya que hacer más sería repetir movimientos. Correr una fila i posiciones significa:

Por cada ficha o vacío en la posición (x, y) pasará a estar en la posición $(x, (y + i)\,\%m)$ siendo m la cantidad de columnas del tablero e i la cantidad de posiciones a correr.

Las reglas son aplicables sólo si la fila a la que afecta no esta vacía, dado que no se estarían moviendo fichas.

Tal y como estan definidas las reglas, por cada tablero puede llegar a haber $n*(m-1)$ ramificaciones, lo que provoca que el árbol de búsqueda sea muy ancho para tableros muy grandes. Por ello se decidió trabajar con tableros de hasta $8x8$.

Para ver un ejemplo de la aplicación de una regla ver el apartado A del anexo.

V. Costos

Dada la naturaleza del problema, la aplicación de todas las reglas tiene el mismo costo, siendo unitario. Esto se traduce a que el costo de ir de un estado al otro es la cantidad de movimientos realizados para llegar a él.

VI. Heurísticas

Por la propiedad de gravedad existente en el problema, es posible explotar en cadena varias fichas. Esto hace que sea extremadamente difícil el aproximar, dado un estado, la distancia a la solución, y provoca que la búsqueda de una buena heurística admisible también lo sea.

La detección temprana de estados irresolubles es muy importante para la optimización de la búsqueda. Para ello se cuenta con las siguiente condicion que un estado debe cumplir para saber que es solucionable, que es común a todas las heurísticas implementadas:

Un estado es resoluble si y sólo si la cantidad de cada color en su tablero es distinto de uno o dos.

Esto es fácilmente demostrable, dado que de no ser asi ya se sabe de antemano que es imposible de explotarlos (se necesitan al menos tres), provocando que el tablero no pueda ser vaciado.

Existe un control para prevenir el cálculo de un mismo tablero dos veces para evitar ciclos infinitos.

Para las heurísticas detlladas a continuación definimos las siguientes funciones:

- $Fichas(t)$ = cantidad de fichas restantes en el tablero.
- $Dim(t) = n*m$ = dimensión del tablero.
- $Colores(t)$ = la cantidad de colores distintos restantes del tablero.
- $Color(t,i)$ = la cantidad de fichas del color i que quedan en el tablero.
- $Islas(t)$ = la cantidad de grupos de dos fichas del mismo color.
- $Movs(t)$ = la cantidad de movimientos realizados desde el estado inicial hasta el actual.

A. Heuristica I - Cantidad de fichas

La heurística corresponde a la fórmula:

$$h_1(t) = Fichas(t)$$

Ésta heuística es claramente no admisible dado que asume que se necesitan tantos movimientos como fichas restantes, lo cual es falso dado que siempre se explotan en más de una, y como ya se mencionó antes, es posible vaciar un tablero con sólo un movimiento, siendo en teste caso $h^*(t) = 1 < h_1(t) = Fichas(t) = Dim(t)$.

B. Heuristica II - Cantidad de explotaciones

Ésta heurística se calcula mediante la fórmula:

$$h_2(t) = \sum_{i=1}^{k} \lfloor Color(t,i)/3 \rfloor$$

Siendo k la cantidad de diferentes colores en el tablero.

Ésta heurística asume que se realiza una explotación de tres fichas por movimiento, lo cual nuevamente la hace no admisible, dado que es posible ganar el tablero en un solo movimiento con explotaciones en cadena, haciendo que $h^*(t) = 1 < h_2(t)$. Esto es así porque no es posible que $h_2(t) = 1$ cuando el tablero esta lleno, porque significaría que éste esta compuesto por un solo grupo de tres fichas del mismo color, lo que no pasa por como está definido el estado inicial del juego.

C. Heuristica III - Colores restantes

El valor heurístico de un estado viene dado por el resultado de:

$$h_3(t) = Fichas(t) * 0,6 + Colores(t) * Dim(t) * 0,4$$

En éste caso el valor heurístico corresponde a un promedio ponderado entre la cantidad de fichas restantes y los colores faltantes, multiplicando este último valor además por las dimensiones del tablero para darle aún más peso en el promedio. Esto es así porque se considera mucho mejor un tablero al cual ya se le eliminaron todas las fichas de un mismo color a uno al que todavía le quedan algunas, dotando a esta heurística de la estrategia de eliminar colores lo más rápido posible.

Ésta heurística tampoco es admisible dado que, como se dijo anteriormente, puede ser que el tablero se gane en un movimiento, pero $h_3(t)$ en un tablero completo siempre es mayor a uno dado que $Fichas(t)*0,6 = Dim(t)*0,6 = n*m*0,6 > 1$ salvo que $n = 1m = 1$, lo que no tiene sentido, y como $Colores(t) * Dim(t) * 0,4 > 0$ llegamos a que $h^*(t) = 1 < h_3(t)$.

D. Heuristica IV

Corresponde a la siguiente fórmula:

$$h_4 = \frac{Fichas(t)}{2(Islas(t)+1)}$$

En esta heuística le da mayor valor a los estados que tienen más fichas agrupadas de a dos, asumiendo que resul-

tará más fácil encontrar una tercera para hacerlas explotar.

No es una heurística admisible, puede verse en un contraejemplo en el anexo, apartado B.

E. Heuristica V

La heurística corresponde a la fórmula:

$$h_5 = \begin{cases} Dim(t) \text{si } t \text{ es un tablero inicial} \\ (Movs(t) * (Dim(t) - Fichas(t) + 1))^{-1} \text{sinó} \end{cases}$$

Ésta heurística favorece a los estados que eliminaron mayor cantidad de fichas en menos movimientos. Diferencia más los estados en una etapa temprana, donde hay más libertad de movimientos, pero no es tan eficaz en estados más cercanos al final. La función heurística es una función partida, dado que su valor sería infinito en el tablero inicial si no lo fuera, porque $Movs(t) = 0$ si t es inicial. A fines prácticos no es un problema porque nunca se calcula el valor heurístico de el nodo raíz, pero teóricamente es incorrecto que ese valor sea infinito.

Se trata de una heurística admisible dado que su valor siempre es menor o igual a uno. Por más de que lo sea, no resulta práctica dado que evalúa a muchos estados con el mismo valor, haciendo que el árbol de búsqueda sea muy ancho, tardando en encontrar la solución.

F. Heurísticas descartadas

Se pensó en una heurística admisible en el cual el valor correspondía a la formula:

$$h(t) = \frac{Fichas(t)}{Dim(t)}$$

Decimos que esta heurística es admisible dado que es imposible que sobreestime a $h^*(t)$. Esto es así porque, a lo sumo, cuando el tablero lleno, se puede resuelve con un sólo movimiento, es decir, costo 1. Esto lleva a que el valor heurístico también sea 1, ya que $Fichas(t)$ correspondiería a su valor máximo (porque el tablero esta lleno) que equivale a la dimensión del tablero, es decir $Dim(t) = n * m$. Ene ste caso $h(t) = h^*(t)$. En cualquier otro caso, cuando el tablero no está lleno, $h(t)$ vale menos que 1, porque $Fichas(t) < Dim(t)$, lo que nos lleva a que $h(t) < h^*(t)$. De estas dos situaciones se saca que $h(t) \leq h^*(t)$.

VII. Resultados

Para ver la tabla de los promedios de los diferentes tiempos tardados en los diferentes métodos de búsqueda ver laa secciones C y D del anexo.

A. Comparación de algoritmos

A.1 Algoritmos no informados

Era de esperarse que el tiempo tardado en BFS crezca mucho cuando el tablero crece de tamaño, dado que como se explicó antes, la cantidad de nodos abiertos por cada nodo está en el orden de $n * (m - 1)$, sin embargo encuentra la solución con la cantidad mínima de pasos, a diferencia de DFS que la encuentra rápido, pero en un nivel más profundo en el árbol. IDDFS, toma lo mejor de cada algortimo, siendo un poquito más lento que DFS, pero encontrando la solución con la cantidad mínima de pasos.

En cuanto a la cantidad de nodos expandidos, no sorprende el hecho de que DFS es el que menos nodos expande, seguido de BFS, y por último IDDFS, dado que éste debe iterar creado y destruyendo varios nodos iguales varias veces.

No se puede hablar de la conveniencia de un método sobre otro ya que esto depende del tablero inicial y no se aplica ninguna estrategia de juego para la explosión de nodos en el árbol, por lo tanto encontrar una solución de forma rápida va a depender de que la solución aparezca en los nodos que primero explota cada método. Por la naturaleza del problema, cada tablero suele tener más de una solución y los mismos suelen resolverse al aplicar una sucesión de varias reglas. Es por esto que podemos ver que el algoritmo de búsqueda DFS (e IDDFS) es el que ha llegado a encontrar una solución de forma más rápida, ya que aplica varias reglas a un mismo tablero, antes de probar con otro, y BFS va aplicando pocas reglas a todos los tableros.

A.2 Algoritmos informados

Se puede observar que *Greedy* es más eficaz en tableros pequeños que A^*, mientras que éste último es un poco más lento pero permite encontrar soluciones en tableros mucho más grandes. Esto se da de esta forma dado que la diferencia entre *Greedy* y A^* es su función $f(n)$, (siendo $f(n)_greedy = h(n)$ y $f(n)_{A*} = g(n) + h(n)$), lo que hace que *Greedy* ignore el costo del estado, es decir la cantidad de pasos que le tardo llegar hasta ahi y sólo tenga en cuenta la heurística. Esto puede provocar, que si las heurísticas evalúan los estados con valores similares, si no tenemos en cuenta el costo, el árbol de búsqueda resulta muy ancho por lo que es estadísticamente más lento al encontrar la solución que con uno más angosto, como se da en A^*. Esto se ve claramente si se echa un vistazo a la tabla de cantidad de nodos expandidos.

B. Comparación de heurísticas

Llama la tención como los tiempos en las distintas heurísticas en el algorimto *Greedy* son muy cercanos, mientras que en A^*, las heurísticas h_4 y h_5 son mucho más lentas que las demás. La diferencia entre *Greedy* y A^* es su función $f(n)$ como se explicó anteriormente, lo que implica que la diferencia está en $g(n)$, la función de costo, siendo más cercana a las heurísticas 1, 2, y 3 que a las 4 y 5.

Podemos concluir que las heurísticas 2 y 3 son las más eficaces a la hora de encontrar soluciones, ya sea en tableros pequeños como grandes, dada la velocidad con la que llegan a la solución. Esto puede explicarse dado que son lasnheurísticas que se aproximan más a la realidad: En el caso de h_2, es fácil darse cuenta empíricamente que la mayoria de las fichas explotadas son por grupos de a tres fichas; en el caso de h_3, tiene mucho sentido tratar de eliminar color por color para lograr el objetivo con mayor precisión.

Las heurísticas 4 y 5 son buenas para tableros pequeños pero resultan inservibles en los que son más grandes, sobretodo en A^*.

VIII. Conclusión

Es notable la diferencia que existe a la hora de buscar una solución a un problema tan sencillo utilizando búsquedas informadas y no informadas. Particularmente, en este caso en el que el árbol de derivación es excepcionalmente ancho, el uso de búsquedas informadas ayuda mucho a la hora de expandir los nodos, y darles un peso a la hora de elegir el próximo a expandir hace que el trabajo resulte órdenes de magnitud más rápido.

Resulta indispensable el calculo de estados ya visitados o estados sin solución sobretodo en algoritmos como DFS que van en profundidad, ya que de no ser así podrían encontrarse en un ciclo infinito corriendo la misma fila una y otra vez sin llegar a ningún lado.

En cuanto a las heurísticas, llama la atención cómo las más intuitivas fueron las más eficaces, dándo una idea de que un algoritmo que se aproxima a la intuición humana es el que lleva a encontrar la solución más rapidamente.

IX. Anexo

A. Ejemplo de aplicación de una regla

Teniendo el siguiente tablero de 3x3:

$$\begin{matrix} 5 & 2 & 2 \\ 4 & 4 & 1 \\ 3 & 1 & 4 \end{matrix}$$

Si aplicamos la regla *correr 1 posición la fila 3* contaremos con el siguiente tablero:

$$\begin{matrix} 5 & 2 & 2 \\ 4 & 4 & 1 \\ 1 & 4 & 3 \end{matrix}$$

Como ahora se tienen tres fichas del mismo color (mismo número en este caso), este grupo explota y las fichas debajo de estas flotan, generando el siguiente tablero:

$$\begin{matrix} 5 & 2 & 2 \\ 1 & 0 & 1 \\ 0 & 0 & 3 \end{matrix}$$

B. Contraejemplo de h_4

Si se cuenta con el tablero siguiente:

$$\begin{matrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 1 & 2 & 3 \end{matrix}$$

Vemos que $h_4(t) = \frac{Fichas(t)}{2(Islas(t)+1)} = \frac{9}{2(0+1)} = 4{,}5$. Pero si aplicamos la regla *correr 1 posición la fila 2* tenemos el tablero:

$$\begin{matrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{matrix}$$

Es facil ver que explotan todas las fichas, llegando al *goal*, y ganando en un movimiento, por lo que $h^*(t) = 1 < 4{,}5 = h_4(t) \Rightarrow h_4(t)$ no es admisible.

C. Tablas de resultados de tiempos

Las siguientes tablas muestran el tiempo promedio en milisegundos al resolver 20 tableros de cada tamaño con cada algoritmo.

C.1 No informados

	4x4[ms]	5x5[ms]	6x6[ms]	7x7[ms]	8x8[ms]
BFS	51	78	∞	∞	∞
DFS	31	68	390	1002	∞
IDDFS	12	47	∞	∞	∞

C.2 Greedy

	4x4[ms]	5x5[ms]	6x6[ms]	7x7[ms]	8x8[ms]
h_1	7	92	154	545	377
h_2	1	101	203	415	371
h_3	1	72	366	534	361
h_4	11	15	155	304	263
h_5	2	5	479	1029	3678

C.3 A^*

	4x4[ms]	5x5[ms]	6x6[ms]	7x7[ms]	8x8[ms]
h_1	6	253	201	355	113
h_2	3	4	123	299	431
h_3	1	8	131	128	36
h_4	14	82	∞	∞	∞
h_5	15	68	∞	∞	∞

D. Tablas de resultados de cantidad de nodos

Las siguientes tablas muestran la cantidad de nodos generado promedio al resolver 20 tableros de cada tamaño con cada algoritmo.

D.1 No informados

	4x4	5x5	6x6	7x7	8x8
BFS	1245	5146	∞	∞	∞
DFS	268	733	1388	2531	∞
IDDFS	2135	7216	∞	∞	∞

D.2 Greedy

	4x4	5x5	6x6	7x7	8x8
h_1	213	788	1256	2178	1598
h_2	214	1270	3014	2987	2541
h_3	213	788	1452	2176	1598
h_4	285	296	1966	2863	1471
h_5	210	214	522	1313	4987

D.3 A^*

	4x4	5x5	6x6	7x7	8x8
h_1	206	446	895	1399	895
h_2	319	380	547	2721	1248
h_3	208	446	1772	1420	845
h_4	304	513	∞	∞	∞
h_5	254	544	∞	∞	∞