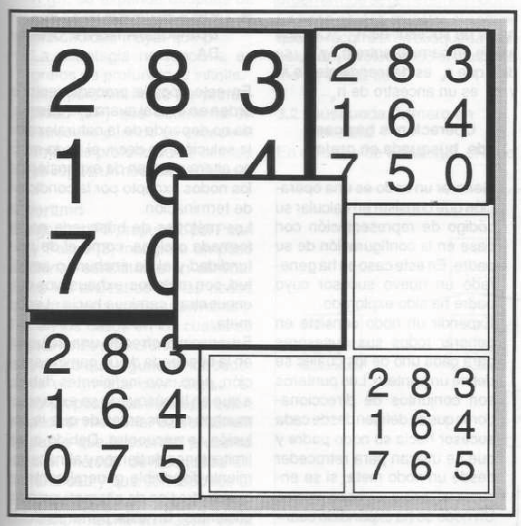
Comparación de métodos de búsqueda



❑ **Marta Elena Millán, M.Sc.**
Licenciada en Matemática - Física
Profesora Asociada
Departamento Ciencias de la Computación
Universidad del Valle

RESUMEN

En este artículo se comparan, en términos de tiempo de ejecución, algunas de las técnicas de búsqueda utilizadas por los solucionadores de problemas. Se detallan los algoritmos comúnmente utilizados para búsqueda informada y no informada. Cada uno de los algoritmos que se presentan, Profundidad, Amplitud, Mejor primero y Estratégico Primero se han implementado en Pascal y se han ejecutado con la misma configuración inicial y final.

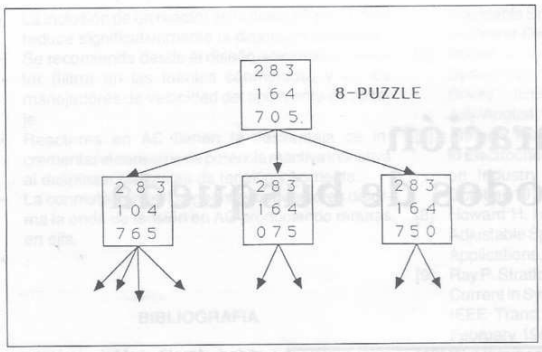
1. INTRODUCCION

Muchos de los métodos de solución de problemas en inteligencia artificial utilizan la noción de ensayo y error y en ellos, los problemas se resuelven buscando una solución en el espacio de posibles soluciones.

Dos enfoques se emplean en la representación de problemas: el de espacio de Estados (EE) y el Enfoque de Reducción (ER).

En el primer enfoque, se utilizan dos conceptos básicos: el concepto de estado y el de operador. Un estado es una descripción o un código que representa un objeto en el espacio y un operador es como una función que transforma un estado en otro. Para el caso del "8 puzzle" un estado es una configuración particular de fichas y un operador corresponde a uno de los cuatro posibles movimientos del espacio blanco: hacia arriba, hacia abajo, hacia la izquierda y hacia la derecha.

En el segundo enfoque se incorpora la noción de subproblema. En este caso, el problema original se representa como un conjunto de subproblemas cuya solución impli-



ca la solución del problema original. Los subproblemas, a su vez, se descomponen en subproblemas hasta obtener problemas primitivos cuyas soluciones sean triviales.

Para encontrar una solución, cualquiera de los enfoques requiere algún tipo de búsqueda.

2. NOTACION Y CONCEPTOS BASICOS

Un grafo es un conjunto de vértices o nodos que representan descripciones de subproblemas o estados. Los grafos que se consideran tienen un nodo inicial asociado con la descripción del problema inicial. Los nodos se conectan por medio de arcos que representan los operadores de que dispone el solucionador de problemas. Los sucesores de un nodo se calculan cuando se aplican operadores a la descripción del estado asociado con el nodo. Un nodo n' es sucesor de otro nodo n si existe un arco de n a n' de tal manera que n es el padre de n'.

Un árbol es un grafo para el que cada nodo, excepto la raíz, tiene solamente un padre. Cuando un nodo no tiene sucesores se dice que es una hoja o nodo terminal.

Un camino de longitud k desde n_1 hasta el nodo n_k, es una secuencia de nodos $n_1, n_2, ..., n_k$ donde cada n_i es un sucesor de n_{i-1}. Cuando existe un camino entre n_1 y n_k se dice que n_k es descendiente de n_1 y n_k es un ancestro de n_k

Operaciones básicas de búsqueda en grafos

- Generar un nodo es una operación que consiste en calcular su código de representación con base en la configuración de su padre. En este caso se ha generado un nuevo sucesor cuyo padre ha sido explorado.
- Expandir un nodo consiste en generar todos sus sucesores para cada uno de los cuales se define un puntero. Los punteros son conjuntos de direccionadores que se definen desde cada sucesor hacia su nodo padre y que se utilizan para retroceder desde un nodo meta, si se encuentra, hacia el nodo inicial. Un nodo se ha expandido cuando todos sus sucesores se generan. El orden en el cual se generan los nodos determina la estrategia o procedimiento de búsqueda. La búsqueda puede ser no-informada o ciega cuando el orden en el que se expenden los nodos depende solamente de la información recogida, hasta el momento, en la búsqueda sin que se tenga en cuenta la parte del grafo que aún no se ha explorado ni la meta, o informada o guiada cuando se utiliza información adicional sobre el dominio del problema y sobre la naturaleza de la meta para guiar la búsqueda en direcciones promisorias.

Los nodos que se han expandido al igual que sus sucesores y que están disponibles en el procedimiento de búsqueda, se les llama cerrados y a los que se han generado y esperan expansión abiertos. Estos dos conjuntos de nodos se registran en dos listas CERRADA y ABIERTA.

3. PROCEDIMIENTOS DE BUSQUEDA NO-INFORMADA

En este tipo de procedimiento el orden en el cual avanza la búsqueda no depende de la naturaleza de la solución, es decir, el nodo meta no altera el orden de expansión de los nodos excepto por la condición de terminación.

Los métodos de búsqueda no-informada o ciega, como el de profundidad y el de anchura o amplitud, son métodos exhaustivos que encuentran caminos hacia el nodo meta.

En principio ofrecen una solución en la búsqueda de un camino solución, pero, son ineficientes debido a que en la búsqueda se expanden muchos nodos antes de que la solución se encuentre. Debido a las limitaciones de tiempo y almacenamiento disponible, generalmente se busca otro tipo de alternativas más eficientes. En esta parte se detallan los algoritmos que pertenecen a este tipo de búsqueda: Primero en profundidad (PP) y Primero en Amplitud o Anchura (PA).

3.1 Búsqueda primero en profundidad

En este tipo de búsqueda tiene prioridad el nivel de profundidad de los nodos en el grafo de búsqueda. Cuando un nodo se selecciona para explorarlo, todos sus sucesores se

consideran primero, antes de que otro nodo sea explorado.
Después de la expansión, uno de los hijos generados se escoge para expansión y el proceso de exploración avanza hasta que se alcanza la meta o el límite de profundidad definido inicialmente.

Características

- Es una técnica LIFO (Last In First Out) en la cual, el nodo de mayor profundidad, en ABIERTA, es el último que se ha generado.
- Cualquier nodo de profundidad, n (p), se expande después de que se hayan expandido los de profundidad mayor que p.
- La estrategia no funciona en grafos de profundidad infinita.
- Incorpora un límite de profundidad (LP) que determina un retroceso cuando se excede el límite de profundidad o cuando se llega a un nodo final-fracaso.

Algoritmo

El algoritmo PP propuesto por Pearl [4] y Pazos [3] incorpora un paso adicional, consistente en eliminar todos los ancestros de los nodos que no tienen sucesores en ABIERTA, en los casos en los cuales se llega al límite de profundidad definido o en los que alguno de los sucesores es un final con fracaso. El único propósito de esta operación es ahorrar memoria. La secuencia de pasos que Nilsson [2] presenta para el método de búsqueda en profundidad y que corresponde al árbol de búsqueda y a la implementación hecha en Pascal es la siguiente:

1. Poner el nodo inicial en una lista llamada ABIERTA.
2. Si ABIERTA está vacía salir con fracaso; en otro caso continuar.
3. Eliminar el primer nodo de ABIERTA y ponerlo en CERRADA; llamar a este nodo n.
4. Si el nivel de profundidad del nodo n es igual al límite de profundidad, ir al paso 2. En otro caso, continuar.
5. Expandir el nodo n, generando todos los sucesores. Añadirlos al comienzo de la lista ABIERTA, en cualquier orden, asignándoles punteros hacia n.
6. Si alguno de los sucesores es un nodo meta, salir con la solución obtenida recorriendo el camino a través de los punteros; en otro caso ir a 2.

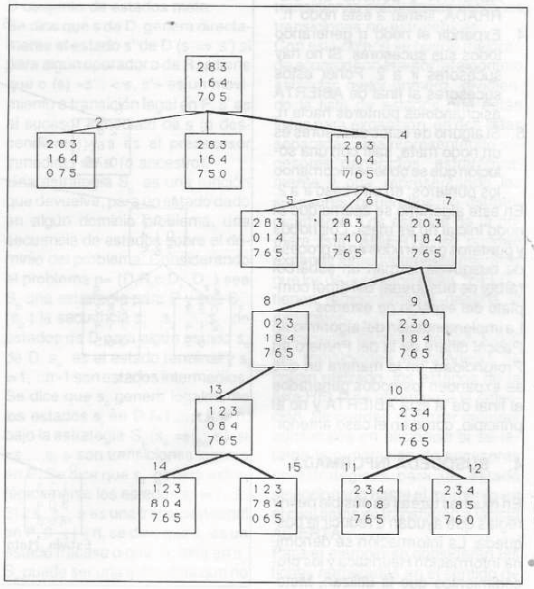
El árbol de la figura anterior describe la aplicación del algoritmo en profundidad al "8-puzzle", de acuerdo con la implementación hecha en Pascal. El número asignado a cada estado corresponde al orden en que se generan los nodos. La línea gruesa muestra el camino que se recorre en la búsqueda de la solución, involucrando el mecanismo de vuelta atrás.

3.2 Búsqueda Primero en Anchura

En este tipo de estrategia, el árbol se recorre por niveles y tienen prioridad los nodos del grafo de búsqueda con nivel de profundidad menor. Del grafo se exploran, progresivamente, secciones en capas de igual profundidad.

Características.

- Es una técnica FIFO (First In First Out), en la cual se explora primero el nodo que mayor tiempo lleva en ABIERTA, es decir el que se haya generado primero.
- Cualquier nodo p, de profundidad n(p), se expande después de que se hayan expandido los de profundidad mayor que p.
- Si existe solución, la estrategia de búsqueda en anchura en grafos finitos garantiza terminar con la solución.
- La solución obtenida es la de más bajo nivel posible.
- En lugar de un solo camino se



almacena una copia completa del grafo de búsqueda que se explora. Por esa razón, en la práctica, los solucionadores de problemas humanos la utilizan poco.

- Exploran casi el doble de nodos que en los métodos de búsqueda primero en profundidad.
- En este método, los nodos se expanden en el mismo orden en que se generan.
- El método garantiza encontrar el camino de solución más corto hacia la meta. Si la meta no existe el método sale con fallo, para grafos finitos.

Algoritmo

1. Poner el nodo inicial en ABIERTA.
2. Si ABIERTA está vacía salir con fracaso, de otra manera continuar.
3. Eliminar el primer nodo de ABIERTA y ponerlo en CERRADA; llamar a este nodo n.
4. Expandir el nodo n generando todos sus sucesores. Si no hay sucesores ir a 2. Poner estos sucesores al final de ABIERTA asignándoles punteros hacia n.
5. Si alguno de estos sucesores es un nodo meta, salir con una solución que se obtiene recorriendo los punteros; en otro caso ir a 2.

En este algoritmo se asume que el nodo inicial no es meta. Los nodos y punteros generados en el proceso de búsqueda forman un subárbol (árbol de búsqueda) del árbol completo del espacio de estados.

La implementación del algoritmo en Pascal difiere de la del Primero en Profundidad, en la manera en que se expanden los nodos generados al final de la lista ABIERTA y no al principio, como en el caso anterior.

4. BUSQUEDA INFORMADA

En muchas tareas es posible definir reglas que ayudan a reducir la búsqueda. La información se denomina Información Heurística y los procedimientos que la utilizan, Métodos de Búsqueda Heurística o Informada.

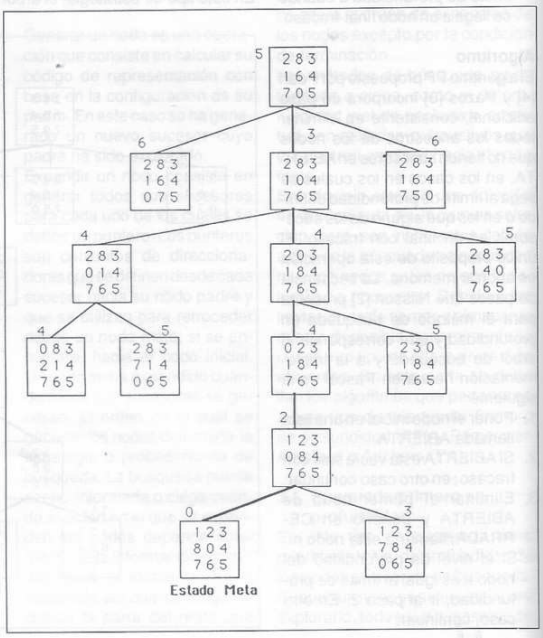
Las técnicas no-informadas se pueden modificar incluyendo más información, por ejemplo, poniendo los sucesores en ABIERTA, en algún orden determinado por Información Heurística, para así, expandir el nodo más prometedor. También se pueden ordenar, en cada paso, los nodos de ABIERTA. Para aplicar cualquier tipo de ordenamiento se requiere medir, de alguna forma, lo "prometedor" de un nodo. Esa medida es una Función de Evaluación.

4.1 Mejor primero (MP)

Lo prometedor de un nodo se determina numéricamente por medio de una función de evaluación f (n) que puede depender de la descripción de n, de la descripción de la meta, de la información disponible en la búsqueda hasta el momento y de cualquier conocimiento adicional que se tenga del dominio del problema. Las diferentes estrategias MP difieren en el tipo de función de evaluación que utilizan. El algoritmo que se describe trabaja sobre un grafo espacio de estados, el nodo seleccionado para expansión es aquel cuyo valor de f (.) es menor.

Algoritmo

1. Poner el nodo inicial s en ABIERTA, lista de nodos no expandidos. Calcular f(n), el valor de la función de evaluación para el nodo.
2. Si ABIERTA está vacía, salir con fracaso; en otro caso, continuar.
3. Eliminar de ABIERTA el nodo n para el cual el valor de f sea mínimo, y ponerlo en CERRADA



mínimo, y ponerlo en CERRADA (utilizada para expandir nodos). Llamar a este nodo n.
4. Expandir el nodo n, generando todos sus sucesores asignándoles punteros hacia n. Asignarle a cada sucesor n' un puntero hacia n.
5. Si alguno de los sucesores es nodo meta, terminar con la solución obtenida retrocediendo a través de los punteros; en otro caso continuar.
6. Para cada sucesor n' calcular f (n') y ponerlos en ABIERTA.
7. Ir al paso 2.

Para el caso particular del "8-puzzle", el árbol de búsqueda se presenta a continuación. El valor de la función de evaluación utilizada calcula el total de fichas en posición incorrecta y aparece al lado de cada configuración del "8-puzzle".
La implementación correspondiente en Pascal introduce dos nuevas funciones. La primera corresponde a la función de evaluación, que para el caso del "8-puzzle", consiste en determinar el número total de fichas fuera de su lugar, con respecto a la meta. La segunda determina el nodo más prometedor para expandir.

4.2. Otros enfoques

Las técnicas de búsqueda, en los Solucionadores de Problemas, se apoyan en funciones heurísticas que miden numéricamente lo prometedor de un estado. Adicionalmente, se incorporan formas de planificación para reducir la búsqueda. El propósito de estos esquemas es construir un plan global para restringir el conjunto de caminos posibles en el espacio de búsqueda y por consiguiente la búsqueda de una solución en este espacio de soluciones restringido.
El plan completo se construye a partir de un estado inicial y hasta un estado meta utilizando, en el diseño del plan, información relacionada con los operadores.
Georgeff [1], examina algunos algoritmos básicos de búsqueda que utilizan este tipo de estrategias como heurísticas para guiar y restringir la búsqueda. La base de estos algoritmos es definir estrategias prometedoras hasta que éstas, o no son aplicables por más tiempo, o generan transiciones ilegales. Al igual que en los métodos de búsqueda, cuando esto ocurre se retrocede y se intenta con alguna otra estrategia. Este tipo de algoritmos se denominan algoritmos de búsqueda estratégica. En esta parte se presentan uno de estos esquemas de búsqueda propuesto por Georgeff[1].

Definiciones

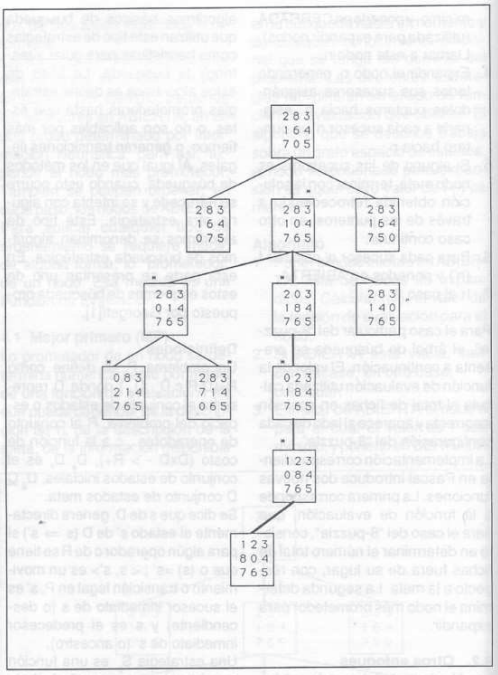
Un problema P se define como $P=(D,R,c,D_s,D_g)$ donde D representa al conjunto de estados o espacio del problema, R al conjunto de operadores , c a la función de costo (DxD -> R+), D_s D, es el conjunto de estados iniciales, $D_g \subseteq$ D conjunto de estados meta.
Se dice que s de D, genera directamente el estado s' de D ($s \Rightarrow s'$) si para algún operador o de R se tiene que o (s) =s' ; < s, s'> es un movimiento o transición legal en P. s' es el sucesor inmediato de s (o descendiente) y s es el predecesor inmediato de s' (o ancestro).
Una estrategia S_p es una función que devuelve, para un estado dado en algún dominio problema, una secuencia de estados sobre el dominio del problema. Considerando el problema $p= (D,R,c,D_s,D_g)$ sea S_p una estrategia para P y sea S_p (s_0) la secuencia $s_1, s_2, ..., s_n$ de estados de D para algún estado s_0 de D. s_n es el estado terminal y s_i i=1,...n -1 son estados intermedios.
Se dice que s_0 genera legalmente los estados s_i en D i=1,...,k,k < n, bajo la estrategia S_p ($s_0 \Rightarrow_{(Sp)} s_i$). si $<s_{i-1}, s_i>$ son transiciones legales en P. Se dice que s_0 genera estratégicamente los estados s_i, i=1,..k. Si $<s_i, s_{i+1}>$ es una transición ilegal en P, $0 \leq j < n$, se dice que s_j es un estado fracaso o que S_p falla en s_j.
S_p puede ser una estrategia que no esté definida para algún estado en D.
Por otro lado, para un estado dado puede haber múltiples estrategias sucesoras, correspondientes a caminos estratégicos diferentes.
Para el caso del "8-puzzle", una estrategia podría ser

```
if blanco no está en el centro
then
    s <- resultado de intercambiar el blanco con la ficha cuya posición ocupa el blanco
else
    s <- resultado de intercambiar el blanco con la primera ficha fuera de su lugar.
```

Búsqueda "Estratégica - primero"

El procedimiento para ejecutar esta búsqueda es básicamente el algoritmo de búsqueda mejor-primero donde el criterio de selección se basa en darle preferencia a las transiciones estratégicas sobre las transiciones no-estratégicas.
Con este criterio se puede mejorar, de acuerdo con el autor, el algoritmo estándar mejor-primero, dividiendo la lista de estados que están listos para expansión, en dos listas separadas: una representando sucesores generados, estratégicamente y otra representando los sucesores generados no-estratégicamente. Los estados de la primera lista, ABIERTA, se pueden expandir inmediatamente mientras que los de la segunda lista, "HELD", tienen temporalmente negada la expansión.
Solamente después de que todas las transiciones estratégicas se hayan utilizado, los estados de la lista HELD se abren para la selección. Se pueden alcanzar mejoras adicionales en eficiencia si se retarda la generación de sucesores no-estratégicas para un estado seleccionado hasta el momento en el cual estos se transfieren a la lista ABIERTA.
Para el ejemplo se consideran las fichas numeradas en el sentido de

las manecillas del reloj comenzando por la esquina superior izquierda. El estado meta es aquel en el cual el número de cada ficha corresponde con su posición y la ficha blanca está en el centro.
Se considera la estrategia representada por la siguiente regla:
Estrategia: Si el blanco no está en el centro, intercambiar el blanco con la ficha cuya posición ocupa el blanco; en otro caso intercambiar el blanco con la primera ficha que está fuera de su lugar.
En esta estrategia se intenta correctamente una ficha, cuando es posible, en un solo movimiento. La función heurística se evalúa para cada nodo que se representa en la figura anterior con un asterisco (*).
En el caso de estratégico-primero se reduce el número de evaluaciones de la función. De esta manera, de acuerdo con el autor, la estratégica- primero es más eficiente que la búsqueda heurística.
El enfoque de búsqueda de "Estratégico-primero" se implementó en Pascal introduciéndole un nuevo procedimiento para calcular estados estratégicos, en el caso en que existan. La estrategia implementada genera caminos de longitud 1 y trata de poner en el lugar correcto una ficha en un solo movimiento, siempre y cuando esto sea posible.
Los casos de prueba con los cuales se ejecutó el programa corresponden con la definición y características de la estrategia propuesta por el autor, en el sentido de que el estado meta es aquel en el cual el número de cada ficha corresponde con su posición y el blanco está en el centro. Los resultados de las ejecuciones se presentan en el apartado siguiente.



5. COMPARACIÓN DE ESTRATEGIAS IMPLEMENTADAS

El siguiente cuadro recoge la información relativa a los resultados de la implementación, para el caso particular del "8-puzzle", de las técnicas de búsqueda primero en profundidad, primero en anchura y mejor primero para un total de 10 ensayos por estrategia. En este caso, los estados inicial y final fueron, en cada ejecución, diferentes. Como se podía prever, la técnica mejor-primero, es más eficiente en términos de tiempo de ejecución. El asterisco (*), para el caso del MP significa que el algoritmo terminó por exceder los tamaños definidos para los vectores y matrices utilizados en la implementación. Es importante anotar, que fue necesario modificar la función de evaluación incorporando el nivel de profundidad de los nodos. De esta manera se obtuvieron los resultados.
Para el caso de ESTRATEGICO-PRIMERO y con el propósito de facilitar su comparación se requirió modificar la función de evaluación del MP, de manera que asignara el

ESTRATEGIA UTILIZADA	ENSAYOS O EXITOSOS	ENSAYOS EXITOSOS	TIEMPO PROMEDIO (CENT.SEG.)
ANCHURA	1	9	0.17
PROFUNDIDAD	1	9	0.062
MP	1 (*)	9	0.024

valor 0 si el estado generado era estratégico y 1 en caso contrario. Debido a las características de la estrategia, todas las configuraciones del estado META son iguales y corresponden al estado (1.2.3.4.5.6,7,8,0).

Para el caso de las dos estrategias de búsqueda informada, el siguiente cuadro detalla los resultados de la ejecución.

Para este caso, la estrategia "Estratégico-primero, es más eficiente que la estrategia Mejor-primero", tanto en términos de tiempo de ejecución como en espacio de almacenamiento requerido. La diferencia básica se encuentra, tal y como lo expresa el autor, en que para el caso del Mejor-primero, se generan todos los sucesores del nodo escogido, de acuerdo con el valor de la función, y por lo que se requiere evaluarla para cada uno de ellos. Para el caso Estratégico-primero, solo se generan sucesores estratégicos, a menos que ABIERTA esté vacía, es decir, los sucesores no-estratégicos de un nodo se generan solamente en caso necesario.

ESTADO INICIAL	TOTAL NODOS MP	GENERADOS EP	TIEMPO MP	EJECUCION EP
	(Seg.,cent.seg.)			
482356701	33	32	0.55	0.16
134250786	14	12	0.16	0.05
234568710	31	14	0.50	0.06
123458670	12	10	0.16	0.06
283456701	12	12	0.17	0.05
120364785	15	14	0.22	0.05
123405786	5	3	0.05	0.0
142350786	14	6	0.22	0.0
813457026	15	7	0.22	0.0
812356740	20	7	0.27	0.0
PROMEDIO			0.252	0.043

Utilizando los mismos casos de prueba y para el mismo estado meta (123456780) se obtuvieron los siguientes resultados para cada una de las estrategias implementadas:

ESTRATEGIA UTILIZADA	TIEMPO PROMEDIO (Seg.,cent. de seg.)
ANCHURA	0.388
PROFUNDIDAD	0.692
MP	0.252
EP	0.043

También en este caso las estrategias de búsqueda informada fueron más eficientes que las no-informadas.

6. BIBLIOGRAFIA

[1]. Georgeff M.P., **Strategies in Heuristic Search. Artificial Inteligence 20** (1983)

[2]. Nilson. Nils, **Problem-Solving Methods in Artificial Intelligence.** Mc. Graw-Hill, New York, 1971

[3]. Pazos S. Juan, **Inteligencia Artificial.** Editorial Paraninfo, 1987.

[4]. Pearl, J. **Heuristics Intelligence Search Strategies for Computer Problem Solving.** Addison Wesley Reading, 1984.

[5]. Nilsson Nils, **Principios de Inteligencia Artificial.** Ediciones Díaz de Santos, Madrid, 1987.

[6]. Zaks Ronald. **Programación en Pascal. Turbo Pascal.** Editorial Anaya 1991