Ingeniería del conocimiento: IA

Práctica 2: Strips

Jorge Bárcena
Miguel Ángel Gil

ÍNDICE

INTRODUCCIÓN 3
ANÁLISIS DE RESULTADOS 3
PROBLEMAS ENCONTRADOS 5

INTRODUCCIÓN

El objetivo de esta práctica era aplicar el algoritmo planificador de “Strips”, en el laberinto. Lo que el algoritmo debía hacer era buscar un plan adecuado para la resolución del puzle planteado. La diferencia entre este mapa y el de la practica anterior de pathfinding era que, en este mapa, para obtener la salida, se debían activar una serie de botones previamente. Lo que tendría que hacer el algoritmo seria buscar un plan correcto que desbloqueara la salida

ALGORITMO STRIP

El algoritmo de “strips” consiste en obtener un resultado meta, para alcanzar este estado se deberán alcanzar otros sub-estados en primer lugar. De esta manera, se obtiene una solución adecuada a nuestro problema, pero esto no quiere decir que sea la óptima. Strips solamente garantiza que la solución es viable.
Hemos aplicado este algoritmo en el proyecto, funcionando correctamente. Para realizar este algoritmo nos hemos basado en el trabajo que hicimos en clase en C# de strips, y para hacer el modelo de datos nos basamos en lo que hicimos en clase el ultimo dia.

ALGORITMO STRIP REGRESIVO

Hemos conseguido aplicar una variante del algoritmo de “Strips” en el proyecto. Para hacer este algoritmo obteníamos el estado final, y a partir de las precondiciones necesarias, construíamos el camino plan.
El pseudocódigo del algoritmo que hemos aplicado es el siguiente:

```
Obtenemos estado Meta del array de operaciones
Ordenamos las precondiciones por el número de (sub)precondiciones que tengan
Funcion aplicar operador(operacion){
        For(pc en operacion.precondiciones)
                Opc = Buscamos en la lista de operaciones, la pc
                Aplicar_operador(opc)
        If(PlanActual no contiene operacion)
                Añadimos operación a planActual
}
```

ANÁLISIS DE RESULTADOS

ALGORITMO STRIP

Hemos realizado una serie de comparaciones entre el número de operadores que se aplican, el número de objetivos que se proponen en la meta y la semilla seleccionada. Los resultados son los siguientes.

SEMILLA	NUMERO OPERACIONES	OBJETIVOS DE LA META
199	**2**	**1**
245	**3**	**2**
898	**2**	**1**
-555	**5**	**3**
458	**3**	**1**
986	**1**	**0**
1235	**1**	**0**
2001	**3**	**1**
14	**4**	**3**
951	**2**	**1**
999	**3**	**2**
1235	**1**	**0**
456	**3**	**1**
789	**2**	**1**
321	**2**	**1**
6544	**2**	**1**
987	**2**	**1**
444	**3**	**2**

Estos resultados indican que cuanto mas compleja es la acción mas operaciones hace, como es lógico. Vamos a comprobar el siguiente algoritmo para ver las diferencias.

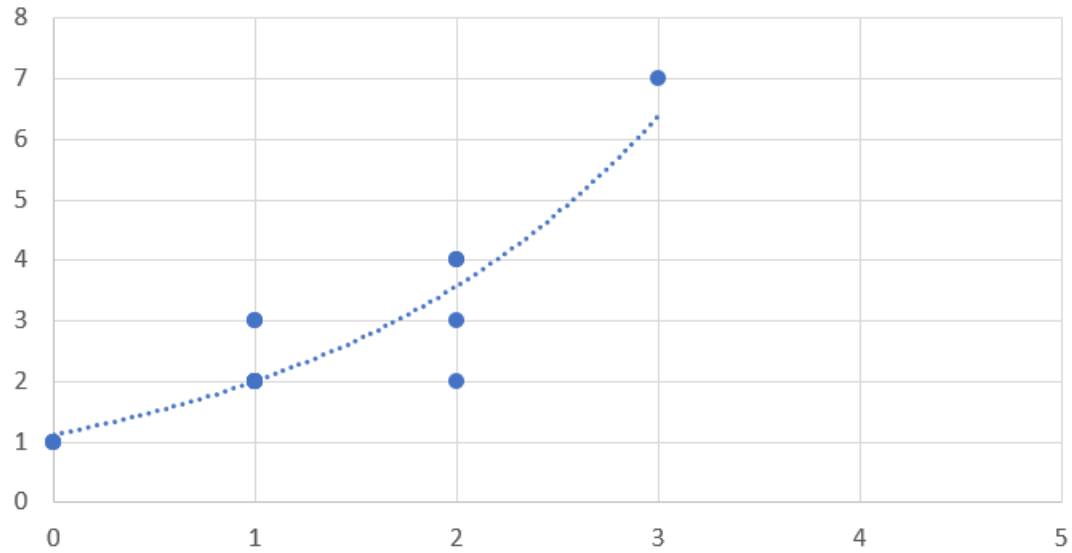
ALGORITMO STRIP REGRESIVO

Hemos realizado una serie de comparaciones entre el numero de operadores que se aplican, el numero de objetivos que se proponen en la meta y la semilla seleccionada. Los resultados son los siguientes.

SEMILLA	NUMERO OPERACIONES	OBJETIVOS DE LA META
199	**2**	**2**
245	**4**	**2**
898	**2**	**1**
-555	**7**	**3**
458	**3**	**1**
986	**1**	**0**
1235	**1**	**0**
2001	**3**	**1**
14	**5**	**3**
951	**2**	**1**
999	**3**	**2**
1235	**1**	**0**
456	**3**	**1**

789	**2**	**1**
321	**2**	**1**
6544	**2**	**1**
987	**2**	**1**
444	**4**	**2**

A raíz de los resultados de los dos algoritmos obtenemos la siguiente gráfica:



Esto muestra, que la línea de tendencia (azul) es exponencial, cuantos más objetivos de la meta más operaciones tiene que hacer el algoritmo. Pero si comparamos los algoritmos nos damos cuenta que el primero es un poco mas eficiente que el segundo, ya que cuando se aumentan el numero de requisitos para llegar a la meta, realiza menos operaciones que el anterior.

PROBLEMAS ENCONTRADOS

GENERACIÓN DE MAPA

Cuando aplicamos el algoritmo, esté funcionaba correctamente, exceptuando las generaciones de mapa cuando, por ejemplo, para desbloquear la casilla de meta las precondiciones eran las siguientes:

Objeto_01 → **Goal** → Objeto_02

En este caso, el algoritmo no encontraba una solución viable, ya que cuando analizaba las precondiciones de la casilla meta, se volvía a encontrar con las precondiciones que ya habían salido con anterioridad.
Debido a que esto era un fallo de la generación del mapa, lo arreglamos añadiendo una

comprobación al generar las precondiciones de la meta, para que no se pueda poner como una precondición de la meta, llegar a la meta.

```
if (generatedObjects[i].Type != PlaceableItem.ItemType.Goal)
    item.Preconditions.Add(generatedObjects[i]);
```

Respecto al algoritmo tuvimos un problema ya que es ocasiones, volvía a meter posiciones que ya se encontraban dentro del planificador. Lo resolví, comprobando si esa posición ya estaba incluida en el plan.

```
if (!init.Plan.Contains(operador))
{

    init.Estado = operador.Aplicar(init.Estado);
    init.Plan.Add(operador);
}
else
    init.Estado = operador.Aplicar(init.Estado);
```