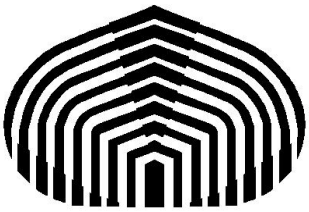

CI-5437
Inteligencia Artificial I
Proyecto I
Búsqueda

Kelwin Fernández y Alejandro Machado
Universidad Simón Bolívar

Junio de 2010

1. Decisiones de implementación

Representación de estados

BFS

Como se cuenta con un máximo de 250 candidatos y un número muy elevado de posibles preferencias distintas, se ha escogido una representación compacta para cada estado, lo que permitirá ahorrar recursos para que el algoritmo pueda concluir su ejecución, en ciertos casos, sin agotar la memoria del computador.

En lugar de almacenar un perfil completo, para cada estado se guarda cuál fue el último cambio elemental realizado (cuál candidato, en cuál preferencia), y un apuntador al estado padre.

Es necesario mantener una lista de estados "cerrados"(ya visitados por el algoritmo) para una correcta implementación de búsqueda en amplitud. En este caso, los estados visitados se mantuvieron en un vector ordenado por un entero asociado a cada estado, que forma también parte de su representación.

Este atributo de cada nodo es calculado con una función de clasificación, que se obtuvo a partir de la heurística sugerida para el algoritmo IDA*. La intuición detrás de esta decisión es la siguiente: dos estados que evalúan a un diferente valor de la función de clasificación deben ser distintos, y por lo tanto no hay que obtener los perfiles asociados y compararlos, lo cual consume tiempo. En algunos casos, una búsqueda binaria sobre el vector de nodos visitados puede arrojar el resultado que se necesita sin ningún cómputo adicional.

Finalmente, también se almacena la profundidad del estado, a fin de no seguir expandiendo fronteras del último nivel si ya se ha hallado una solución en éste.

IDA*

Para este algoritmo se utilizó un solo perfil, y sobre éste se aplican y desaplican cambios elementales para explorar el espacio de estados.

Explicar lista de cerrados.

Obtención de sucesores

BFS

Para obtener los sucesores de un estado, debe construirse primero el perfil asociado y luego aplicar todos los posibles cambios elementales sobre éste.

IDA*

Se aplica un cambio elemental sobre el perfil actual, y se explora el perfil hijo. Posteriormente, se deshace este cambio elemental; este proceso se repite para cada transición posible.

Optimizaciones

General

1. Agregar un votante a una preferencia existente dentro de un perfil es $O(log(n))$, donde n es el número de preferencias. Esto se logra manteniendo las preferencias ordenadas dentro de cada perfil.

- Optimizaciones: BFS: - Mantener dos perfiles: padre y un hijo, para no recomputar el padre en toda una frontera - Se guarda el nivel de profundidad para no expandir nodos más allá de una meta - Función de clasificación: dado que es una funcion, si dos nodos son iguales sus funciones serán iguales, ahorrar comparaciones pesadas.

/bin/bash: xclip: command not found

- Detalles de agregar preferencia (en el sentido de hacer una agregacion logaritmica en ciertos casos)
- Nodos generados almacenados en un vector ordenados (búsqueda logarítmica)
- Explicar mejora de considerar que para los hijos de un mismo padre se computa este una sola vez.
- Se precomputa para cada nodo del bfs una funcion de clasificacion que permite no tener que generar todos los estados de la tabla de visitados para comparar con el nuevo a estado a expandir. Presentar informalmente ciertos resultados que se obtuvieron con respecto a las proporciones.

2. Discusión de resultados

3. Recomendaciones