Extendiendo Graphplan con Técnicas de Aprendizaje

Diego García
dgarcia@cs.uns.edu.ar

Alejandro García
agarcia@cs.uns.edu.ar

Laboratorio de Investigación y Desarrollo en Inteligencia Artificial (LIDIA)
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur.
Av. Alem 1253, (8000) Bahía Blanca, Argentina

Planificar eficientemente requiere de heurísticas de búsqueda específicas a cada dominio de planificación. Sin embargo, construir heurísticas apropiadas para cada nuevo dominio es una tarea difícil. Una alternativa es utilizar técnicas de aprendizaje para adquirir conocimiento automáticamente que permita guiar la búsqueda y mejorar la performance del planificador. Graphplan es uno de los planificadores de propósito general más promisorios, sin embargo, la técnica de planificación que utiliza es simplemente una búsqueda sobre el grafo de planificación.

El propósito de esta línea de investigación es mejorar la performance de Graphplan mediante la utilización de técnicas de aprendizaje. La idea básica es extender el algoritmo de Graphplan de forma tal que durante la resolución de un problema de planificación genere conocimiento (experiencia) que podrá ser utilizado para resolver nuevos problemas del mismo dominio de planificación. El conocimiento generado podría ser utilizado para mejorar el algoritmo de Graphplan en dos aspectos: en la fase de expansión del grafo y en la elección de acciones en la fase de búsqueda del plan.

Aspectos a mejorar de Graphplan

El algoritmo de Graphplan [1] está basado en el paradigma de "análisis del grafo de planificación" y utiliza una estructura llamada *grafo de planificación* que codifica el problema de planificación de manera que muchas restricciones inherentes al problema se manifiestan explícitamente, reduciendo así la búsqueda en el grafo. El algoritmo alterna entre dos fases: expansión del grafo y búsqueda del plan. Durante la fase de expansión, el grafo de planificación se extiende hacia adelante en el "tiempo", hasta lograr una condición necesaria (aunque insuficiente) para la existencia de un plan. En la fase de búsqueda del plan se realiza una búsqueda hacia atrás (backward-chaining) sobre el grafo generado hasta el momento, buscando un plan que resuelva el problema. Si se encuentra un plan, el proceso termina, de lo contrario, el ciclo se repite expandiendo el grafo más niveles.

Un grafo de planificación (ver Figura 1) es un grafo dirigido con dos tipos de nodos organizados en niveles: los niveles pares (representados con un círculo) contienen nodos de proposición, esto es, precondiciones o efectos de acciones. El nivel 0 consiste precisamente de las proposiciones que son verdaderas en el estado inicial del problema de planificación. Los niveles impares contienen nodos de acción (representados con rectángulos) que corresponden a instancias de acciones cuyas precondiciones están presentes en el nivel anterior, y cuyos efectos están en el nivel siguiente. El grafo tiene tres tipos de arcos que representan relaciones entre las acciones y las proposiciones. Los nodos de acción en un nivel de acción i están conectados a sus precondiciones en el nivel de proposición $i-1$ por *arcos de precondición*, y a sus efectos en el nivel $i+1$ por *arcos de agregación* (add-edges) representados por líneas sólidas y *arcos de eliminación*

(delete-edges) representados por líneas punteadas. Existe un tipo especial de acción llamado *no-op* o *frame action* (cuadrado sólido), que representa la posibilidad que una proposición se mantenga inalterada de un nivel de proposición i al siguiente $i + 2$.

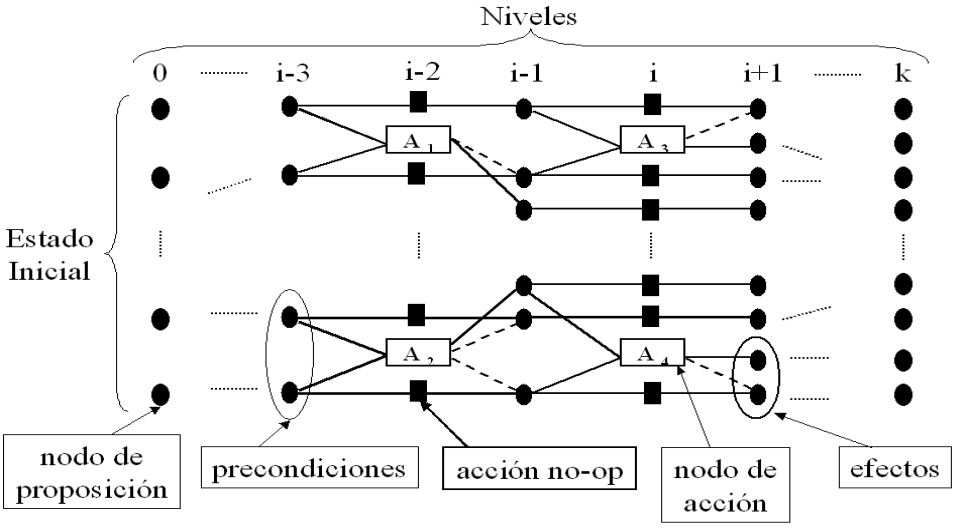

Figura 1: Esquema del grafo de planificación

En Graphplan cuando se resuelve un problema que es similar a otros resueltos con anterioridad, se vuelve a expandir el grafo en su totalidad, sin tener en cuenta la experiencia obtenida de haber resuelto problemas anteriores. Por ejemplo, en el *mundo de bloques* al resolver el problema de apilar tres bloques que están sobre la mesa, el algoritmo de Graphplan construirá un grafo considerando todas las acciones aplicables en cada nivel. Una gran cantidad de estas no contribuirán a la solución, y será descartada por el proceso de búsqueda que seleccionará aquellas acciones que resuelven el problema. Si se toma luego un problema similar como apilar cuatro bloques, el planificador volverá a generar el grafo con todas las acciones aplicables, considerando nuevamente una gran cantidad de acciones que no contribuyen a la solución del problema. Además al ser cuatro bloques, la cantidad de acciones totales aumenta considerablemente. Sin embargo, si la experiencia de cuales acciones fueron relevantes para resolver el problema de apilar tres bloques, es almacenada en forma de conocimiento para el planificador, entonces al intentar resolver el problema para apilar cuatro bloques, la elección de acciones podría estar guiada por este conocimiento.

Como la búsqueda se realiza backward-chaining, en cada nivel de acción i del grafo se debe seleccionar un subconjunto S de las acciones en i que cumplan las metas del nivel $i + 1$. Luego se debe seleccionar un subconjunto S' de acciones del nivel $i - 2$ que cumplan las metas del nivel $i - 1$, representadas por las precondiciones de las acciones en S. Si el conjunto S' existe, se continúa hacia atrás hasta llegar al nivel O. De lo contrario, si S' no existe en $i - 2$ se debe seleccionar un subconjunto alternativo en i (backtraking). El problema es que esta búsqueda hacia atrás es ciega, esto es, el planificador no utiliza ninguna heurística para seleccionar el conjunto de acciones en cada nivel. Sin embargo, para guiar este proceso de selección se podría utilizar conocimiento generado durante la búsqueda de un plan en problemas similares anteriores.

Trabajo previo relacionado

En [2] se aplicaron técnicas de aprendizaje a un planificador sencillo (linear state-based progression planner) con el objetivo de mejorar su performance. El planificador obtenido utiliza la experiencia generada en planificaciones anteriores para resolver un nuevo problema de planificación. El planificador durante la búsqueda de un plan genera conocimiento en la forma de reglas rebatibles "$do(accion) \prec situacion$", para representar información de cuales acciones resultaron convenientes ejecutar para encontrar un plan a partir de una situación particular de la búsqueda. Una regla de la forma "$do(a) \prec s$" indica que si el planificador "se encuentra en una situación s, entonces hay razones para ejecutar la acción a". El planificador también genera reglas de la forma "$\sim do(a) \prec s$" para representar que "si se encuentra en la situación s hay razones para no aplicar (o elegir) la acción a". De esta forma, se genera conocimiento a favor o en contra de ejecutar (o elegir) acciones en determinadas situaciones. Por lo tanto, cuando se desea encontrar un plan para un nuevo problema, en lugar de explorar el árbol de búsqueda exhaustivamente, las reglas rebatibles generadas en planificaciones anteriores son utilizadas para argumentar sobre que acción aplicar en cada estado.

Los resultados obtenidos fueron alentadores, pero al tratarse de un planificador que exploraba el espacio de estados, el tamaño del espacio de búsqueda representaba un problema para problemas no triviales.

Generación de conocimiento y planificación basada en la experiencia

La propuesta de esta línea de trabajo es incorporar técnicas de aprendizaje a un planificador como Graphplan que ha demostrado tener un mejor desempeño para un amplio espectro de dominios de planificación. El conocimiento generado tendrá la forma de reglas a favor o en contra de realizar una acción. De esta manera, un mecanismo de razonamiento sobre el conocimiento generado podrá utilizarse para guiar la búsqueda en problemas futuros.

Durante la búsqueda de un plan, un punto importante a tener en cuenta es cuando se produce backtracking. El backtracking en un nivel de acción i indica que el subconjunto de acciones S elegido en el nivel i no fue el correcto para lograr las metas del nivel $i+1$. Sin embargo, es posible que sólo un subconjunto de acciones de S sea responsable del backtraking. Este subconjunto puede ser identificado a través de las relaciones de exclusión mutua que se producen durante la búsqueda, en niveles de acción anteriores a i. Una vez identificado este subconjunto podrían generarse reglas que indiquen que subconjunto de acciones no conviene elegir para lograr las metas del nivel $i+1$.

En el caso que un conjunto de acciones elegido en un nivel i resulte satisfactorio (forme parte del plan) para lograr las metas del nivel $i+1$, podrían generarse reglas que indiquen que subconjunto de acciones es conveniente elegir para lograr las metas del nivel $i+1$.

De esta forma, cuando el planificador se enfrente a un nuevo problema de planificación, puede utilizar las reglas generadas en búsquedas anteriores para guiar la elección de acciones en cada nivel.

Referencias

[1] A. L. Blum and M. L. Furst. *Fast Planning through planning graph analysis* Journal of Artificial Intelligence, 90 (1-2):281-300, 1997.

[2] Diego García y Alejandro García *Planificación con Aprendizaje de Reglas Rebatibles para elección de Acciones utilizando Argumentación* Workshop de Investigadores en Ciencias de la Computación. WICC'02. Universidad Nacional del Sur, Bahía Blanca, 2002.

[3] R. Kambhampati, E. Lambrecht and E. Parker. *Understanding and Extending Graphplan* In Proc. 4th European Conference on Planning. September 1997.

[4] Daniel S. Weld. Recent advances in AI planning. *AI Magazine*, 20(2):93–123, 1999.