Algoritmo de planificación HTN con preferencias.

Alumnos

Cesar Daniel Pérez.
José Ramón Ciacciarelli.

Directores

Dr. Andrés Díaz Pace.
Dr. Luis Berdun.

Facultad de Ciencias Exactas
Universidad Nacional del Centro de la Provincia de Buenos Aires
UNCPBA

Mayo 2013

1- Motivación

La necesidad de los agentes de interactuar con el mundo con el fin de cumplir con objetivos impuestos ha hecho que los investigadores en Inteligencia Artificial investiguen distintos algoritmos para la generación de secuencias apropiadas de acciones.

En particular, los algoritmos de planning se utilizan cuando es necesario ejecutar un número de acciones en un orden coherente para alcanzar metas, o cuando las acciones interactúan de manera compleja [Weld 94].

En este sentido, Hierarchical Task Network (HTN) es un paradigma de planning ampliamente utilizado, del cual existen varias implementaciones (por ejemplo SHOP2, SIPE-2, I-X/IPLAN, O-PLAN) [Ghallab 04]. A diferencia de los algoritmos de planning convencionales en HTN, el planner es provisto con un conjunto de tareas a ser ejecutadas, posiblemente con restricciones sobre las mismas. En este contexto, un plan es formulado mediante la descomposición repetida de tareas en sub-tareas más pequeñas hasta llegar a lo que se denomina tareas primitivas (tareas ejecutables, las cuales pueden ser vistas como las acciones básicas de los algoritmos convencionales). Una de las razones del éxito de HTN radica en que la descripción de los métodos de descomposición de tareas captura el conocimiento del dominio, lo que puede reducir de forma significante el espacio de búsqueda para un plan, ya que se introduce información del dominio en la misma forma de resolver el problema.

Si bien es posible obtener un conjunto de planes satisfactorios que resuelven el problema, no siempre la calidad del plan obtenido puede estar acorde a lo deseado por el usuario. Es en este contexto donde aparecen los algoritmos de planning basados en preferencias. Estos algoritmos aumentan el problema de planning especificando propiedades que constituyen un plan de calidad. Por ejemplo, si se está generando un plan de viaje, atributos de calidad del plan podrían ser: que se minimice el costo, que no se utilicen escalas intermedias, o que se prefiera pasar por una determinada ciudad, entre otros. Los algoritmos de planning basados en preferencias intentan optimizar la satisfacción de estas preferencias mientras alcanzan los objetivos del plan.

En [Berdun 12] se presenta una extensión del algoritmo de planning UCPOP que permite la incorporación de preferencias y restricciones en la formulación del problema de planning. En este trabajo los autores toman como base una definición de preferencias centrada en las acciones y no en los estados de un plan. Como también se menciona en [Shorabi 09] las preferencias especificadas en PDDL 3.0 son expresivas pero se encuentran centralizadas en los estados del plan, identificando estados preferidos a lo largo de la trayectoria del plan.

2- Trabajo Propuesto

El trabajo propuesto consiste en el desarrollo de una herramienta que permita incrementar la personalización de los planes generados por un algoritmo de planning HTN, introduciendo preferencias definidas por el usuario.

La idea es poder generar planes que no solo dependan de las restricciones del problema, sino también poder definir reglas sobre los objetos a ser evaluados por el algoritmo. La definición de las preferencias estará desacoplada de la definición del problema.

Las implementaciones HTN incluyen a planners como:

- Nonlin [Tate 76], que se encuentra escrito en POP-2 (Burstall, Collins y Popplestone, 1971). Este planner hace uso de un sistema de base de datos llamado HBASE (Barrow,1975) y utiliza un formalismo para especificar el dominio (TF) Task Formalism. Este planner asegura tener 2 opciones a la hora de elegir los caminos para llegar al objetivo. Mantiene una estructura objetivo (Gost) *Goal Structure*. Esta estructura permite saber el valor de cada nodo en particular en cualquier momento.

- SIPE-2 [Wilkins 93], permite interactuar con el usuario mediante el uso de *Advisable Planner*. Este le permite al usuario ingresar información adicional a la del dominio a la hora de generar un plan. SIPE-2 también se puede utilizar para modificar los planes generados anteriormente, una capacidad esencial para la re planificación en tiempo de ejecución.

- O-PLAN [Tate 95], que se centra en la generación y perfeccionamiento de varios cursos de acción (COA). Su uso principal fue en la intervención militar, teniendo en cuenta cambios repentinos y la necesidad de readaptar el plan para su pronta solución.

- UMCP [Kutluhan 94], ofrece búsqueda automática e interactiva. Interfaz gráfica de usuario para navegar a través del espacio de búsqueda.

- SHOP2 [Nau 03], reduce la complejidad de los razonamientos mediante la eliminación de una gran cantidad de incertidumbre acerca del mundo. Puede hacer inferencia axiomática, cálculos simbólicos/numérico mixtos, y llamadas a programas externos. Incorpora características de PDDL, como cuantificadores, efectos y precondiciones. Además puede manejar dominios de planificación temporal traduciendo los operadores temporales de PDDL.

Pero hay que tener en cuenta que los planificadores mencionados anteriormente utilizan implementación de bajo nivel para poder obtener mayor velocidad de procesamiento. Por

lo general utilizan implementaciones pensadas a optimizar el hardware sobre el que se va trabajar y no tanto pensando en cambiar a futuro o la necesidad de modificar alguno de sus componentes.

Es en este sentido que decidimos utilizar la programación orientada a objetos para implementar la herramienta, con el objetivo de generar un framework flexible para configurar y experimentar con distintas políticas de preferencias.

La programación orientada a objetos es un paradigma que utiliza objetos como elementos fundamentales en la construcción de la solución. Surge en los años 80. Un objeto es una abstracción de algún hecho o ente del mundo real que tiene atributos que representan sus características o propiedades y métodos que representan su comportamiento o acciones que realizan. Todas las propiedades y métodos comunes a los objetos se encapsulan o se agrupan en clases. Una clase es una plantilla o un prototipo para crear objetos; por eso se dice que los objetos son instancias de clases. Las virtudes de este paradigma, en comparación con implementaciones de bajo nivel, incluyen:

- Reusabilidad. Cuando hemos diseñado adecuadamente las clases, se pueden usar en distintas partes del programa y en numerosos proyectos.
- Mantenibilidad. Debido a la sencillez para abstraer el problema, los programas orientados a objetos son más sencillos de leer y comprender, pues nos permiten ocultar detalles de implementación dejando visibles sólo aquellos detalles más relevantes.
- Modificabilidad. La facilidad de añadir, suprimir o modificar nuevos objetos nos permite hacer modificaciones de una forma muy sencilla.
- Fiabilidad. Al dividir el problema en partes más pequeñas podemos probarlas de manera independiente y aislar mucho más fácilmente los posibles errores que puedan surgir.

Para poder desarrollar la herramienta emplearemos técnicas de *frameworks orientados a objetos*, es decir, un esqueleto de partes de software que interactúan entre sí, pero sin tener que depender cada uno de la implementación del otro. Con esto logramos independizar las partes de la herramienta y solo se comunicarían por las entradas y salidas que comparten.

Una vez implementada la herramienta se planea testearla con los distintos dominios utilizados para las diferentes competencias de planning. En este sentido es necesaria primeramente una adaptación de los mismos a HTN, para luego poder probar el desempeño del algoritmo teniendo en cuenta las preferencias. El foco del trabajo está puesto en la calidad de los planes y en un soporte flexible de la herramienta, para su posterior utilización. Este es un aspecto a considerar ya que se busca la calidad del plan por sobre el tiempo utilizado para alcanzar una solución.

La herramienta a desarrollar toma como entrada los datos propios del problema, junto con los datos propios de las preferencias de los usuarios y permite obtener planes que

resuelven el problema, pero ajustándose a lo que el usuario prefiera, siempre teniendo en cuenta que no se violen las restricciones del dominio en el que se está trabajando.

Con este trabajo pretendemos mejorar la calidad de los planes obtenidos por los algoritmos HTN. Mejorar la calidad en la misma gestación del plan, ya que se pueden ir eliminando resultados parciales a la hora de generar un plan y no tener que llegar a la solución final para saber, por ejemplo que la misma no cumple con las preferencias del usuario.

3- Cronograma de Actividades

Actividad	**Duración**
Recopilación bibliográfica.	6 semanas
Análisis de las diferentes implementaciones de planners HTN	5 semanas
Diseño del framework orientado a objetos de planning y sus hotspots para preferencias	5 semanas
Implementación del algoritmo	12 semanas
Evaluación del Algoritmo	6 semanas
Escritura de informe de tesis, reportando el trabajo realizado (en paralelo con las Actividades anteriores).	18 semanas

4- Bibliografía

[Baier 06] Baier, J., Hussell, J., Bacchus, F., & McIlralth, S. (2006). Planning with temporally extended preferences by heuristic search. In 5th international planning competition booklet (IPC-2006), Lake District, England (pp. 20–22).

[Benton 06] Benton, J., Kambhampati, S., & Do, M. B. (2006). YochanPS: PDDL3 simple preferences and partial satisfaction planning. In 5th international planning competition booklet (IPC-2006), Lake District, England (pp. 54–57).

[Berdun 12] Berdun, L., Amandi, A. & Campo, M. (2012) An Agent specifc Planning Algorithm. Expert Systems with Applications, ISSN: 0957-4174, ELSEVIER, Volume 39, Issue 5, Pages 4860-4873.

[Chih-Wi 06] Chih-Wi, H., Wah, B., Ruoyun, H., & Yixin, C. (2006). New features in sgplan for handling preferences and constraints in PDDL3.0. In 5th international planning competition booklet (IPC-2006), Lake District, England (pp. 39–41).

[Edelkamp 06] Edelkamp, S. (2006). Optimal symbolic PDDL3 planning with MIPS-BDD. In 5th international planning competition booklet (IPC-2006), Lake District, England (pp. 31–33).

[Jabbar 06] Edelkamp, S., Jabbar, S., & Naizih, M. (2006). Large-scale optimal PDDL3 planning with MIPS-XXL. In 5th international planning competition booklet (IPC-2006), Lake District, England (pp. 28–30).

[Ghallab 04] Ghallab, M., Nau, D., & Traverso, P. (2004). Automated Planning, Theory and practice. 1558608567. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

[Sohrabi 09] Sohrabi, S., Baier, J. A., & McIlraith, S. A. (2009). HTN planning with preferences. In Proceedings of the 21st international joint conference on artificial intelligence (IJCAI-09), Pasadena, CA, USA, July (pp. 1790–1797).

[]Weld 94] Weld, D. S. (1994). An introduction to least commitment planning. AI Magazine, 15, 27–61.

[Tate 76] Tate, A. (1976) "Project Planning Using a Hierarchic Non-linear Planner", D.A.I. Research Report No. 25, August 1976, Department of Artificial Intelligence, University of Edinburgh.

[Wilkins 93] D. E. Wilkins. *Using the SIPE-2 Planning System: A Manual for Version 4.3*. SRI International Artificial Intelligence Center, Menlo Park, CA, August 1993.

[Tate 95] Tate, A., Drabble. B. and Dalton, J., An Engineers Approach to the Application of Knowledge Based Planning and Scheduling Techniques to Logistics - O-Plan Final Technical Report RL-TR-95-235 December 1995.

[Kutluhan 94] Kutluhan Erol, Dana Nau, and James Hendler. "UMCP: A Sound and Complete Planning Procedure for Hierarchical Task-Network Planning". In AIPS-94, Chicago, June, 1994.

[Nau 03] Nau, D. S., Au, T. C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., & Yaman, F. (2003). SHOP2: An HTN planning system. J. Artif. Intell. Res. (JAIR), 20, 379-404.

http://es.wikipedia.org/wiki/Programación_orientada_a_objetos

http://es.wikipedia.org/wiki/Framework