Planificación, preferencias y conocimiento: motivación y problemas abiertos

Jorge Baier

Profesor Asistente Departamento de Ciencia de la Computación, Pontificia Universidad Católica de Chile. Doctor of Philosophy, University of Toronto; Ingeniero Civil Industrial y Magíster en Ciencias de la Ingeniería, Pontificia Universidad Católica de Chile; Áreas de interés: Planificación Automática, Búsqueda en Inteligencia Artificial, Representación y Razonamiento en Mundos Dinámicos.

jabaier@ing.puc.cl

La planificación —que es como suelo traducir la palabra en inglés *planning*— es una actividad que los seres humanos realizamos todos los días cuando pensamos qué se debe hacer para lograr cierto objetivo. Un ejemplo es la planificación de un viaje: si he decidido aceptar hoy dar una charla mañana a las 15:00 hrs. en Viña del Mar, ¿qué debo hacer? La respuesta a esta pregunta es probablemente un conjunto de acciones, algunas de las cuales tienen que ver con la preparación de la charla, otras tienen que ver con cómo llegar a Viña y otras con comunicar mi plan a otro agente (mi señora, por ejemplo).

Más específicamente, el problema implica componer acciones, las que consideramos entes productoras de un cambio en el mundo, de tal forma que al ejecutar esas acciones, se produzca un cambio deseado en éste. El problema es central en el desarrollo de la inteligencia artificial, dado que se considera a la planificación como una actividad que todo agente autónomo deberá en su momento realizar. Es de interés principal en esta área la planificación *independiente del dominio*, que significa que el programa que planifica debe ser capaz de procesar un problema en un dominio que nunca antes ha visto, ojalá, de forma tan hábil como lo hacemos las personas.

En su versión más sencilla —llamada planificación clásica *(classical planning)*— el problema se modela como una tupla *(I, A, G),* donde *I* representa un estado inicial, *A* es un conjunto de acciones, y *G* es una condición que se debe cumplir sobre los estados objetivo, es decir, aquellos estados del mundo a los que nos interesa llegar. Cada acción *a* en *A* se modela como una tupla *(prec(a), eff(a)),* donde *prec(a)* representa la precondición de la acción, es decir, la condición que se debe cumplir antes de ejecutar la acción. Por otro lado *eff(a)* representa los efectos de la acción, es decir, aquellas condiciones que deben satisfacerse en el estado que resulta de ejecutar *a*. Dentro de *eff(a)* se distinguen los efectos positivos y los efectos negativos. Los efectos positivos son aquellos hechos que se harán verdaderos una vez que ejecutamos *a* (por ejemplo, al viajar de Santiago a Viña se hace verdadero que estoy en Viña). Los efectos negativos hablan de las condiciones que se hacen falsas una vez ejecutada la acción (en el ejemplo anterior, al viajar de Santiago a Viña se hace falso que estoy en Santiago).

En planificación clásica se supone que al ejecutar una acción en un estado del mundo *s*, el agente llega a otro estado del mundo *s'*. Más aún, se supone que este estado es único (el mundo es determinístico), y que se conoce exactamente cómo computar *s'* desde *s*. La condición *G* se puede pensar como definiendo un conjunto de estados del mundo a los que nos interesa llegar después de ejecutar una secuencia de acciones tomadas desde *A*.

Así como lo definimos, el problema de planificación se puede entender como una búsqueda de un camino en un grafo. En efecto, lo que se quiere resolver es un problema de alcanzabilidad en el grafo determinado por el estado inicial, y por los estados generados por la aplicación sucesiva de acciones sobre él. Visto desde el punto de vista de la teoría de grafos, este problema se puede resolver usando el algoritmo de Dijkstra, publicado en 1959.

¿Qué hace entonces que este problema aún sea objeto de estudio? La respuesta a esta pregunta es larga y daré algunas ideas. Mientras la respondo, aprovecharé de describir temas que desarrollo en mi investigación. También espero dejar una lista de problemas que aún se consideran abiertos y que, creo, seguirán sin tener una respuesta satisfactoria por algún tiempo.

HEURÍSTICAS: UNA MANERA DE ATACAR ESTE DIFÍCIL PROBLEMA

El gran problema de usar un algoritmo como Dijkstra para planificación, es que simplemente no sirve desde el punto de vista práctico. El espacio de estados del grafo donde buscamos es demasiado grande en general. Desde el punto de vista teórico, alcanzabilidad es un problema NLOGSPACE-complete. Esa complejidad, sin embargo, está expresada en términos del tamaño del grafo, no del tamaño de nuestra tupla *(I, A, G)*. Lamentablemente, el grafo definido por tal tupla puede ser exponencial en el tamaño de ella: la complejidad de planificación clásica (de decidir si un plan existe) es realmente PSPACE-complete, lo que significa que no conocemos algoritmos eficientes para resolver estos problemas. Los mejores algoritmos que tenemos ejecutan, de hecho, en tiempo exponencial en el tamaño del problema.

Para verlo en términos un poco más prácticos, pensemos en el espacio de búsqueda del conocido y sencillo *mundo de bloques*, que consiste en una mesa llena de bloques, donde el agente puede tomar uno sólo a la vez y dejarlo sobre otro bloque o sobre la mesa. El objetivo es dejar los bloques en una configuración dada. Este sencillo problema, hasta hace unos diez años no era resuelto por ningún planificador automático de forma razonablemente rápida. Incluso hoy los mejores planificadores tienen problemas

para resolver ciertas configuraciones. ¿Cuál es el tamaño del espacio de estados de este problema? Si tenemos n bloques, está dado por:

$$\sum_{k=1}^{n} \frac{n!}{k!} \binom{n-1}{k-1}$$

con lo que concluimos que con 20 bloques, tenemos un espacio de alrededor de 3×10^{20} estados (agradezco a mi alumno Nicolás Rivera por deducir la fórmula). Por otro lado, el famoso puzle del cubo Rubik tiene alrededor de 4×10^{19} estados. ¿Diríamos que resolver problemas del mundo de bloques con 20 o más de estos, es más difícil que resolver un cubo Rubik?

Los seres humanos somos capaces de resolver muchos problemas con espacios de estados gigantes sin, aparentemente, requerir de un esfuerzo de cómputo demasiado grande. Una posible explicación es que tenemos la habilidad de distinguir sin mucho esfuerzo cuándo un estado *s* está más o menos cerca del objetivo, al mirar unas pocas características de *s*.

Pero, ¿cómo es que logramos que un computador, se dé cuenta que un problema es fácil, como lo hacemos nosotros? Con búsqueda ciega (por ejemplo, Dijkstra), resolver 20 bloques es *mucho* más difícil que el cubo Rubik. En planificación usamos algoritmos de búsqueda informados, que usan una función —llamada función heurística— para guiar la búsqueda. Más precisamente, la heurística es una función que estima la dificultad de llegar al objetivo desde un estado particular. En el año 2000 recién se desarrollaron los primeros planificadores automáticos que eran capaces de resolver el problema de planificación usando búsqueda heurística, donde la heurística usada es una que se computa *automáticamente*, es decir no es entregada por el usuario. Desde ese momento ha surgido gran interés por desarrollar esta técnica, especialmente porque estos planificadores demostraron ser notablemente mejores que los que existían en el pasado.

Resulta que una de las mejores heurísticas independientes del dominio es tan simple que se puede explicar con una sola oración: para estimar la dificultad de llegar a algún estado de *G* a partir de un estado *s* imaginamos otro problema en donde las acciones sólo tienen efectos positivos y luego resolvemos el problema desde *s* y retornamos el largo de la solución como la estimación. Afortunadamente, el problema sin efectos negativos, que es una relajación del problema original, se puede resolver en tiempo *polinomial* en el tamaño del problema original. Llamaremos a esta forma de relajar el problema la *relajación libre de efectos negativos* (RLEN).

Uno de los temas de mi investigación ha tenido que ver precisamente con esta problemática: sobre cómo construir estas funciones de manera automática. A pesar de que la relajación libre de efectos negativos funciona muy bien en muchos problemas, hay otros en los que conduce al planificador a tomar decisiones tan malas que hacen que un problema trivial para cualquier ser humano no pueda ser resuelto por los mejores planificadores actuales. Encontrar relajaciones mejores que la RLEN ha probado ser un problema muy difícil. Junto a mis colegas hemos propuesto una solución a éste [1]. Las heurísticas que propusimos permiten mejorar la RLEN en muchos casos, pero tienen la desventaja que son más lentas de computar, lo que hace que sean difíciles de aplicar.

HACIA MODELOS MÁS RICOS DE PLANIFICACIÓN

Tal como está descrita, la planificación clásica parece tener aplicaciones limitadas. A simple vista el mundo no parece determinístico (a menos que deseáramos modelar hasta las más mínimas interacciones físicas entre partículas). Tampoco es cierto que los seres humanos estemos interesados sólo en el objetivo, sin importar qué ejecutamos, porque todos tenemos preferencias sobre qué cosas queremos hacer, qué estados nos gustan o cuáles queremos evitar, etc. En mi ejemplo del viaje a Viña, junto con preferir minimizar el tiempo de viaje, podría preferir aprovechar el tiempo de viaje leyendo; esto hace difícil tomar una decisión sobre el tipo de vehículo a utilizar (bus versus mi auto).

Entonces, si tenemos un conjunto de preferencias, ¿cómo es posible construir planificadores automáticos que busquen soluciones preferidas en forma eficiente? Si el problema clásico es difícil, éste parece serlo aún más. En 2006 [2] propusimos un planificador automático que utilizaba heurísticas desarrolladas para planificación clásica para este problema. El planificador tuvo un desempeño aceptable en una competencia internacional en donde participó. Hasta el día de hoy, sin embargo, este problema es motivación para mi investigación. Por ejemplo, junto a mi alumno León Illanes, hemos descubierto que existen muchos problemas de estos, que los humanos consideramos "fáciles" y que son muy difíciles para los planificadores con preferencias actuales. El problema es que las heurísticas para planificación son aún muy malas en presencia de preferencias.

Otro tema que me motiva son los objetivos más complejos. ¿Qué pasa si mi objetivo no es solamente llegar a un estado, sino que me interesa que la trayectoria para llegar a ese objetivo tenga una cierta propiedad?¿Podemos hacer planificación eficiente para este caso? La respuesta a la que llegamos cuando investigamos esto es satisfactoria: resulta que cuando los objetivos son expresados usando lógica temporal lineal, es posible usar resultados de teoría de autómatas para reducir el problema a uno de planificación clásica [3]. El enfoque tiene un peor caso, muy malo, donde el resultado de la reducción es exponencial en el problema original. Sin embargo, para muchas situaciones prácticas este peor caso no se manifiesta.

Aún quedan problemas abiertos: no sabemos realmente si las heurísticas existentes son las mejores cuando se usan objetivos temporales. Actualmente no hay estudios

publicados que se refieran seriamente a este tema.

APLICACIONES NO ESTÁNDARES DE PLANIFICACIÓN

Planificación es un problema PSPACE-completo, lo que significa que podemos reducir todo problema en PSPACE a un problema de planificación. Es decir, podemos usar planificadores para problemas que no han sido necesariamente concebidos como problemas de planificación.

¿Hay algún problema interesante que se pueda resolver usando un planificador de forma más efectiva que usando otro "solver" específico? La respuesta también parece ser positiva. Junto con mis colegas mostramos que cierto tipo de problemas de diagnóstico dinámico (es decir, el problema de encontrar fallas en un sistema dinámico) también se puede resolver usando planificadores en forma efectiva [4]. Esto motivó incluso que propusiéramos un nuevo paradigma de planificación, para el cual aún no conocemos buenos planificadores [5].

INTEGRACIÓN Y ADQUISICIÓN DE CONOCIMIENTO PROCEDURAL

Imaginemos ahora que quiero implementar un agente (un robot o un programa autónomo) y quiero programarlo, pero no quiero hacer un programa gigante con miles de *if-then-else*, sino que quiero dar flexibilidad al agente. Es decir, estoy dispuesto a llenar al programa con constructos no-deterministas, que el agente, dependiendo de las condiciones "complete" durante la ejecución. En muchos casos, el problema de cómo completar estos hoyos no-determinísticos es algo que se puede reducir a planificación clásica [6] y, como consecuencia, es posible aprovechar el poder de los *solvers* actuales.

Claramente los seres humanos no planificamos desde cero cada vez que tenemos un problema de planificación. Yo ya sé cómo ir a la universidad todos los días y es poco lo que planifico todos los días. Incluso si deseo ir a otro lugar es posible que reutilice un pedazo de mi plan para resolver otro problema. En efecto, pareciera que el plan para llegar a la universidad parece como si fuera una especie de programa general, que yo mismo construí. Actualmente, ésta es toda un área de investigación que ha mostrado avances interesantes, pero aún parece estar lejos del rendimiento obtenido por enfoques más sencillos como los que construyen algún tipo de *porfolio* de planificadores.

MEJORES ALGORITMOS DE BÚSQUEDA

Los mejores planificadores que han surgido en los últimos años se han caracterizado no sólo por usar novedosas heurísticas, sino también por introducir importantes modificaciones a los algoritmos de búsqueda existentes. El conocido A*, un algoritmo de búsqueda que puede usar una heurística *h*, tiene muy mal rendimiento en planificación, usándose mejoras sobre éste, como por ejemplo, priorizar aquellas acciones que aparecen en el plan relajado obtenido de resolver la relajación LEN.

No está claro, sin embargo, si los algoritmos actuales son los mejores para el caso, por ejemplo, de preferencias. Cuando se planifica con preferencias el problema parece mucho más uno de optimización, pero en el que, además, sólo encontrar una solución factible (para qué decir la óptima) puede ser muy difícil. En estos casos, parece ser que la construcción de algoritmos incrementales, que retornen soluciones cada vez mejores es la mejor solución. Sin embargo, no está claro cómo aprovechar el esfuerzo realizado en las búsquedas pasadas para obtener buenas soluciones. El algoritmo que mejor rendimiento tiene hasta el momento es uno que descarta todo lo aprendido en el pasado [7], lo que no parece intuitivamente razonable. Cuando hay múltiples objetivos o preferencias, pareciera más razonable aprender qué pedazos del plan satisfacen ciertas preferencias. Estos pedazos se podrían reutilizar en futuras búsquedas. Este es otro tema que actualmente forma parte de mi foco en esta área. BITS

Referencias

[1] Jorge A. Baier, Adi Botea: Improving Planning Performance Using Low-Conflict Relaxed Plans. ICAPS 2009.

[2] Jorge A. Baier, Fahiem Bacchus, Sheila A. McIlraith: A heuristic search approach to planning with temporally extended preferences. Artif. Intell. 173(5-6): 593-618 (2009).

[3] Jorge A. Baier, Sheila A. McIlraith: Planning with First-Order Temporally Extended Goals using Heuristic Search. AAAI 2006: 788-795.

[4] Shirin Sohrabi, Jorge A. Baier, Sheila A. McIlraith: Diagnosis as Planning Revisited. KR 2010.

[5] Sammy Davis-Mendelow, Jorge A. Baier, and Sheila McIlraith: Assumption-Based Planning: Generating Plans and Explanations under Incomplete Knowledge. AAAI 2013. To appear.

[6] Jorge A. Baier, Christian Fritz, Sheila A. McIlraith: Exploiting Procedural Domain Control Knowledge in State-of-the-Art Planners. ICAPS 2007: 26-33.

[7] Silvia Richter, Jordan Tyler Thayer, Wheeler Ruml: The Joy of Forgetting: Faster Anytime Search via Restarting. ICAPS 2010: 137-144.