Reducción de la Planificación Conformante a SAT mediante Compilación a d–DNNF

Héctor Palacios[1] and Héctor Geffner[1]

[1] Universitat Pompeu Fabra
Passeig de Circumval·lació, 8.
Barcelona, España
hector.palacios@upf.edu
[2] ICREA/Universitat Pompeu Fabra
Passeig de Circumval·lació, 8.
Barcelona, España
hector.geffner@upf.edu

Resumen La planificación clásica consiste en encontrar una secuencia de acciones que convierta un estado inicial conocido en un estado final dada una descripción de los estados y de las acciones en términos de un conjunto de variables. La planificación conformante, por otro lado, es una variación de la planificación clásica donde el estado inicial es parcialmente conocido y las acciones pueden tener efectos no-determinísticos sobre las transiciones de estados. Un plan conformante debe convertir cualquier estado inicial posible en un estado final para cualquier posible transición de estados. Con horizontes acotados, la planificación clásica es NP-completa y por tanto reducible a SAT, mientras que la planificación conformante es más compleja y no es reducible polinomialmente a SAT (a menos que P = NP). De hecho, el enfoque lógico utilizado hasta ahora en la planificación conformante se basa en un lazo de generación y testeo, donde tanto la generación como el testeo puede tomar tiempo exponencial en el peor de los casos: posibles planes conformantes son generados uno a uno invocando un motor SAT, y el resultado es verificado a través de una segunda invocación SAT. En este trabajo, reemplazamos el lazo de generación y testeo por un esquema donde la teoría CNF se compila primero en formato d–DNNF (Darwiche 2000). De allí se obtiene en tiempo lineal una teoría cuyas variables son sólo las acciones, y de la cual se obtiene un modelo, a través de una sola invocación SAT, que codifica un plan conformante para el problema. Si bien la compilación a d–DNNF es exponencial en el peor de los casos, los resultados que se obtienen experimentalmente son mucho mejores. De hecho, demostramos que los resultados obtenidos para el planificador conformante óptimo resultante son competitivos con el estado del arte.

1. Introducción

La planificación clásica consiste en encontrar una secuencia de acciones que convierta un estado inicial conocido en un estado final dada una descripción de los estados y de las acciones en términos de un conjunto de variables. La planificación conformante, por otro lado, es una variación de la planificación clásica donde el estado inicial es parcialmente conocido y las acciones pueden

tener efectos no-determinísticos sobre las transiciones de estados. Un plan conformante debe convertir cualquier estado inicial posible en un estado final para cualquier posible transición de estados. Con horizontes acotados, la planificación clásica es NP-completa y por tanto reducible a SAT, mientras que la planificación conformante es más compleja [1] y no es reducible polinomialmente a SAT (a menos que P = NP).

De hecho, Ferraris y Giunchiglia reportan un planificador conformante que usa dos búsquedas anidadas [2]. La primera para generar posibles planes conformantes uno a uno como modelos de la teoría, y la segunda para verificar que lo son. El proceso para generar planes candidatos es exponencial. A su vez, puede haber un número exponencial de candidatos. La verificación de que un candidato es un plan conformante también requiere tiempo exponencial. En este trabajo, reemplazamos ese lazo de generación y testeo, por *proyección* para obtener una teoría que sólo contenga variables de acción, y una llamada a un *motor SAT* para encontrar un plan conformante. Para proyectar eficientemente proponemos *compilar* la teoría a d–DNNF. A pesar de la intratabilidad de las formas compiladas cómo OBDDs [3] y d–DNNFs, han sido usadas con gran éxito.

El estado del arte en planificadores óptimos conformantes esta representado por los trabajos de Rintanen [4] y más recientemente por Palacios *et al.* [5]. Rintanen hace búsqueda heurística en el espacio de creencias, que resumen las posibles situaciones del agente. Cómo heurística usa alcanzabilidad con pares de estados iniciales. Palacios compila a una forma normal llamada d–DNNF [6] y realiza una búsqueda en el espacio de planes parciales, podando cuando muestra por conteo y proyección que no satisface algún estado inicial. De los planificadores que no garantizan optimalidad a fin de resolver más problemas, destaca CFF [7] que también hace búsqueda heurística.

Lo resultados que presentamos mejoran sustancialmente los resultados de Palacios *et al* con un esquema más simple que usa dos herramientas generales disponibles, un compilador a d–DNNF y un motor SAT, sin tener que codificar un algoritmo de búsqueda.

Tanto en Palacios *et al* como en este trabajo asumimos que todas las acciones son determinísticas y que por tanto toda la incertidumbre del problema esta en la situación inicial[3]. En ambos trabajos se reportan resultando permitiendo que las acciones se ejecuten en paralelo, bajo condiciones de no interferencia.

El artículo esta organizado de la siguiente manera. Primero definimos el problema de planificación conformante y su formulación en lógica proposicional. Luego estudiamos como obtener modelos que correspondan a planes conformantes. Para ello nos detenemos en la proyección como operación lógica, y a continuación los d–DNNFs cómo forma compilada que nos permite realizar la proyección de manera eficiente. Finalmente presentamos el algoritmo del planificador conformante, resultados experimentales y una discusión final.

[3] En principio, las acciones no determinísticas pueden ser convertidas a acciones determinísticas con efectos condicionales, donde una de las condiciones es desconocida.

2. Planificación y Lógica Proposicional

La planificación clásica consiste en encontrar una secuencia de acciones que convierta un estado inicial conocido en un estado final dada una descripción de los estados y de las acciones en términos de un conjunto de variables. La planificación conformante, por otro lado, es una variación de la planificación clásica donde el estado inicial es parcialmente conocido y las acciones pueden tener efectos no-determinísticos sobre las transiciones de estados. Un plan conformante debe convertir *cualquier* estado inicial posible en un estado final para *cualquier* posible transición de estados.

Consideramos un lenguaje para describir problemas de planificación conformante donde un problema P es tupla de la forma $P = \langle F, O, I, G\rangle$, F es el conjunto de símbolos de hechos del problema, O es el conjunto de acciones, e I y G son los conjuntos de clausulas definidas sobre los hechos de F que describen las situaciones iniciales y finales, respectivamente. A su vez, cada acción a tiene una precondición $\text{pre}(a)$, un conjunto de literales positivos, y una lista de efectos condicionales $\text{cond}^k(a) \rightarrow \text{effect}^k(a)$, $k = 1, \ldots, n_a$, donde $\text{cond}^k(a)$ y $\text{effect}^k(a)$ son conjunciones de literales de los hechos F. Nótese que estamos tratando con acciones determinísticas solamente. Por ello $\text{effect}^k(a)$ es una conjunción de literales que es el único efecto de la acción si se cumple la condición $\text{cond}^k(a)$.

En el enfoque lógico, el problema de encontrar un plan para P ejecutable en N instantes de tiempos se resuelve a traves de verificar una fórmula. La codificación proposicional para planificación conformante que usamos es un pequeña variación de la usada en SATPLAN [8] para planificación clásica. Esta consta de variables booleanas x_i para los hechos y las acciones x del problema, donde i es el índice temporal que va desde $i = 0$ hasta el horizonte $i = N$ (no se crean variables de acción x_N para el último instante de tiempo). Para una fórmula B usaremos la notación B_i para expresar la fórmula obtenida al reemplazar cada variable x en B por su contraparte x_i del instante de tiempo i. Describimos a continuación la teoría proposicional $T(P)$ que codifica un problema $P = \langle F, O, I, G\rangle$ con horizonte N:

1. **Inicial:** una clausula C_0 por cada clausula de la situación inicial $C \in I$
2. **Meta:** una clausula C_N por cada clausula de la meta $C \in G$.
3. **Acciones:** Para $i = 0, 1, \ldots, N-1$ y $a \in O$:

$$\begin{aligned} a_i &\supset \text{pre}(a)_i && \text{(precondiciones)} \\ \text{cond}^k(a)_i \wedge a_i &\supset \text{effect}^k(a)_{i+1}, \quad k = 1, \ldots, k_a && \text{(efectos)} \end{aligned}$$

4. **Frame:** para $i = 0, 1, \ldots, N-1$, cada literal del hecho $l \in F$

$$l_i \wedge \bigwedge\nolimits_{\text{cond}^k(a)} \neg[\text{cond}^k(a)_i \wedge a_i] \supset l_{i+1}$$

donde la conjunción varía sobre las condiciones $\text{cond}^k(a)$ asociadas con los efectos $\text{effect}^k(a)$ que 'eliminan' l.

5. **Exclusión:**

$$\neg a_i \ \vee \ \neg a'_i$$

para $i = 0, \ldots, N-1$ si a y a' son incompatibles.

Frame se refiere a los axiomas expresan las persistencia de los hechos en la ausencia de acciones que los efecten. *Exclusión* impide la ejecución de acciones incompatibles en un instante de tiempo. En la formulación serial sólo se permite la ejecución de una acción en cada instante. En la formulación paralela permitimos la ejecución simultanea de acciones que no pueden afectar los mismo literales. Llamaremos *Init* a la subteoría que describe la situación inicial y al conjunto de los posibles estados iniciales: $s_0 \in \mathit{Init}$. Un planificador óptimo procede incrementando el horizonte N desde $N = 0$ hasta encontrar un plan. En teorías paralelas el horizonte óptimo es aquel en que logra hacer un plan, lo cual no minimiza el número de acciones, pero si la duración en la ejecución del plan.

3. Planificación Conformante y Modelos

En planificación clásica la relación entre la codificación $T(P)$ y el problema P es tal que los modelos de $T(P)$ corresponden uno-a-uno con los planes que resuelven P. Un modelo de $T(P)$ codifica sólo un plan optimista: un plan que funciona para algún estado inicial, pero es necesario que funcione en todos. Así, dado *Plan*, un conjunto consistente maximal de literales de acción[4], cuando P es un problema de planificación clásica, *Plan* es una es una solución si y sólo si:

$$T(P) \ + \ \mathit{Plan} \qquad \text{es satisfactible} \tag{1}$$

Un plan conformante, sin embargo, debe funcionar para todos los estados iniciales s_0. Por lo tanto *Plan* es un plan conformante para el problema P si y sólo si:

$$\forall \ s_0 \in \mathit{Init} : \ \ T(P) \ + \ \mathit{Plan} \ + \ s_0 \qquad \text{es satisfactible} \tag{2}$$

Para encontrar un *Plan* que satisfaga 1 es suficiente encontrar un modelo de $T(P)$[5]. Esto no sirve para encontrar un *Plan* que satisfaga 2.

Una primera aproximación para reducir el problema de planificación conformante a SAT sería reemplazar 1 con 3:

$$T(P)' \ + \ \mathit{Plan} \qquad \text{es satisfactible} \tag{3}$$

donde $T(P)'$ es una conjunción que toma en cuenta *todos* los estados iniciales posibles:

$$T(P)' \quad = \quad \bigwedge_{s_0 \in \mathit{Init}} T(P) \,|\, s_0 \tag{4}$$

[4] el equivalente sintáctico de una valuación

[5] el plan lo constituyen los literales positivos de ese modelo

donde $T \mid X$ se refiere a la teoría T con las variables x en T reemplazadas por el valor que tiene en X: $x \leftarrow$ *verdadero* si $x \in X$, y $x \leftarrow$ *falso* si $\neg x \in X$. Esto se conoce como *condicionamiento*.

Si las ecuaciones 3 y 4 fueran correctas, podríamos intentar hallar un plan conformante encontrando un modelo para $T(P)'$ y luego extrayendo el *Plan* del valor de las variables de acción en ese modelo.

Sin embargo, la formulación 3-4 *no es correcta*. La razón es que la teoría $T(P)$ contiene variables f_i que no están en *Init* ni en *Plan*: éstas son las variables que se refieren a los hechos en tiempos $i > 0$. Mientras en Eq. 2, para cada s_0, estas variables pueden tomar valores diferentes, en la formulación 4 estas variables están *forzadas* a tomar el mismo valor para cada s_0, tal como las variables de acción.

Es posible modificar la definición 4 de $T(P)'$ en 3 para obtener una formulación SAT correcta para planificación conformante. Todo lo que necesitamos es *eliminar* las variables f_i de $T(P) \mid s_0$ en 4.

La eliminación de un conjunto de variables S de una teoría T (*forgetting* [9]) es la operación dual de la *Proyección* de T sobre el conjunto V de variables que no están en S ($vars(T) = V + S$). La proyección de T sobre V, denotada $\mathsf{project}[\ T\ ;\ V\ ]$, es una teoría sobre las variables V cuyos modelos son exactamente los modelos de T, restringidos a las variables V. Por ejemplo, si $\phi = (a_1 \wedge f_1) \vee a_2$ entonces $\mathsf{project}[\,\phi;\,\{f_1\}\,] = a_1 \wedge a_2$. Si nos interesan los modelos de ϕ que se refieren a las variables a_i, podemos eliminar f_1 haciendo análisis por casos: $\mathsf{project}[\,\phi;\,\{f_1\}\,] = (\phi \mid f_1 = v) \vee (\phi \mid f_1 = f) = ((a_1 \wedge v) \vee a_2) \vee ((a_1 \wedge f) \vee a_2) = (a_1 \vee a_2)$.

En $T(P) \mid s_0$ las variables son las variables de acción *Actions*, o las variables f_i que representan los hecho en los tiempos $i > 0$. La eliminación de las variables f_i en $T(P) \mid s_0$ es entonces la proyección de $T(P) \mid s_0$ sobre el conjunto de variables de acción *Actions*. Con esta modificación, la definición de $T(P)'$ en 3 deviene:

$$T_{\mathrm{cf}}(P) \quad = \quad \bigwedge_{s_0 \in Init} \mathsf{project}[\ T(P) \mid s_0\ ;\ Actions\ ] \tag{5}$$

Theorem 1. *Los modelos de la fórmula $T_{cf}(P)$ (Eq. 5) están en correspondencia con los planes conformantes para el problema P.*

La fórmula 5 sugiere un **esquema** de planificación conformante donde **(1ro)** condicionamos sobre cada s_0: $T(P) \mid s_0$. Luego, **(2do)** proyectamos las acciones $\mathsf{project}[\ T(P) \mid s_0\ ;\ Actions\ ]$ a fin de tener modelos sólo sobre las acciones y, finalmente, **(3ro)** construimos la conjunción de las proyecciones $T_{\mathrm{cf}}(P)$, a fin de obtener el mismo modelo para cada proyección. Los planes conformantes son obtenidos a travez de un motor SAT sobre la teoría $T_{\mathrm{cf}}(P)$ (Eq. 5). El punto crucial es como generar esa teoría de manera efectiva, dado que el problema de planificación conformante no es reducible a SAT.

4. Proyección y d–DNNF

Compilación de conocimiento (*Knowledge Compilation*) es el área de la Inteligencia Artificial que se ocupa del problema de convertir teorías lógicas a fragmentos que permitan que ciertas operaciones sean tratables [6]. De hecho, los OBDDs [3] han sido usados con mucho éxito en verificación formal [10] y en planificación [11], a pesar de que compilar a OBDD es intratable en general.

Un lenguaje de compilación más reciente es el d–DNNF (*deterministic Decomposable Negation Normal Form*) [6]. Los DNNF suportan numerosas consultas y operaciones en tiempo polinomial, en particular la *proyección*, que puede ser exponencial sobre CNF. OBDD es un caso particular de d–DNNF. Más aún, existen OBDDs que son exponencialmente más grandes que sus d–DNNFs equivalentes [6].

4.1. Descomponibilidad y Determinismo en la NNF

Una fórmula esta en NNF (*Negational Normal Form*) si esta hecha de literales usando sólo conjunciones y disyunciones. Una manera de representar las sentencias NNF es mediante un grafo acíclico dirigido (DAG), donde cada hoja en el DAG es un literal, *verdadero* o *falso*; y cada nodo interno es una conjunción $\wedge$ o una disyunción $\vee$.

Definition 1 (Darwiche 2002). *Una* DNNF *es una* NNF *que satisface la propiedad de descomposición: para cualquier conjunción $\wedge_i \alpha_i$ en la fórmula, ninguna variable aparece en mas de un conjunto α_i.*

La descomponibilidad permite verificar satisfactibilidad en tiempo lineal mediante un recorrido de abajo hacia arriba en el DAG. Un nodo $\vee$ ($\wedge$) será satisfactible si alguno (todos) de sus hijos es (son) satisfactible(s).

Definition 2 (Darwiche 2001). *Un d–*DNNF *(deterministic* DNNF*) es un* DNNF *que satisface la propiedad de determinismo: para cualquier disyunción $\vee_i \alpha_i$ en la forma normal, cada par de disjuntos α_i es mutuamente inconsistente.*

Para una teoría proposicional Δ cualquiera se cumple la siguiente ecuación de análisis por casos sobre la variable a

$$\Delta \quad \equiv \quad (\Delta \,|\, a \ \wedge \ a) \quad \vee \quad (\Delta \,|\, \neg a \ \wedge \ \neg a) \tag{6}$$

dónde $\Delta|a$ se refiere a simplificar la teoría Δ reemplazando a por *verdadero*, y en $\Delta|\neg a$ reemplazando a por *falso*. Nótese que esta fórmula cumple localmente la propiedad de descomposición y determinismo. Si asumimos que $\Delta \,|\, a$ y $\Delta \,|\, \neg a$ son d–DNNFs, la descomposición se cumple porque las conjunciones $\Delta \,|\, a \ \wedge \ a$ no comparten variables siendo que $\Delta \,|\, a$ no contiene la variable a. También se verifica el determinismo porque la disyunción tiene a ambos disjuntos inconsistentes, puesto que en uno aparece a y en el otro $\neg a$. Si se aplica 6 recursivamente sobre una teoría Δ se obtendrá una compilación a d–DNNF equivalente a Δ.[6]

[6] Evidentemente se pueden hacer muchas optimizaciones, como no resolver subproblemas dos veces, lo que hace que los d–DNNF sean grafos.

4.2. Proyección y Condicionamiento en d–DNNF

En este trabajo queremos generar una teoría equivalente a $T_{\mathrm{cf}}(P)$ (Eq. 5), y verificar sus satisfactibilidad. Para generar $T_{\mathrm{cf}}(P)$ podemos condicionar sobre los s_0 y proyectar sobre las acciones *Actions*. Para condicionar cualquier d–DNNF sobre una asignación de variable X es suficiente, como en cualquier teoría proposicional, sustituir los literales por su valor de verdad en X: $x \leftarrow$ *verdadero* si $x \in X$, y $x \leftarrow$ *falso* si $\neg x \in X$. Proyectar un d–DNNF sobre un conjunto de variables V se logra sustituyendo los literales de *las demás variables* por *verdadero* [6], con lo que desaparecen de la teoría y no se pierden ni se agregan modelos nuevos. Ambas operaciónes toman tiempo lineal en la formula d–DNNF.

5. Planificador Conformante

Integrando las observaciones anteriores, obtenemos un esquema de planificación conformante basado en compilar-proyectar-SAT:

1. Obtener $T(P)$ como CNF a partir de una extensión del lenguaje de especificación de problemas PDDL que permite incertidumbre en la situación inicial
2. **Compilar** la teoría $T(P)$ obteniendo la teoría T_{c} en d–DNNF
3. Condicionar para cada $s_0 \in$ *Init*: $T_{\mathrm{c}} \,|\, s_0$
4. **Proyectar** las acciones para cada $s_0 \in$ *Init*, y construir el CNF equivalente a la conjunción de cada proyección[7]

$$T_{\mathrm{cf}}(P) \quad = \quad \bigwedge_{s_0 \in Init} \mathsf{project}[\, T_{\mathrm{c}} \,|\, s_0 \;;\; Actions \,]$$

5. Usar un **motor de SAT** sobre $T_{\mathrm{cf}}(P)$ para encontrar un modelo que contendrá un plan para P. En caso contrario el problema no es solucionable para ese horizonte.

Los procesos de naturaleza exponencial en este esquema son la compilación (paso 2), y la única invocación al motor SAT (paso 5).

El proceso de generar $T_{\mathrm{cf}}(P)$ a partir de T_{c} puede hacerse de manera eficiente si le indicamos al compilador que: a) al compilar elimine los hechos f_i para $i > 0$, b) el análisis por casos se haga primero sobre las variables de *Init*. En ese caso $\mathsf{project}[\, T(P) \,|\, s_0 \;;\; Actions \,]$ para cada s_0 es un subgrafo de T_{c}. El compilador a d–DNNF usado (`c2d` v2.18, mantenido por Adnan Darwiche), provee soporte para estos requerimientos.

El motor SAT usado es `siege_v4` que ha sido reportado como apropiado para teorías de planificación. En caso de que `siege_v4` no logre cargar la teoría (por poseer demasiadas variables) se usa *zChaff*.

[7] traducir de $T_{\mathrm{cf}}(P)$ en forma de grafo NNF a CNF es una operación lineal

6. Experimentación

Hicimos los experimentos en una PC Intel/Linux a 2.80Ghz con 2Gb de RAM. Los experimentos se limitaron a 2 horas y 1.8 Gb de memoria. Usamos el mismo conjunto de problemas que [5] y [4], donde están descritos con cuidado. Son problemas difíciles que caracterizan el problema de planificación conformante, distinguiéndolo de la planificación clásica. **Anillo-n:** trata de un robot que recorre una circunferencia de n cuartos para cerrar ventanas. **Redes de Ordenamiento-n:** requiere hacer un circuito para ordenar un vector de n entradas. **Centro-del-cuarto-n:** requiere que un robot ciego llegue al centro de un cuatro de $2^n \times 2^n$; la solución pasa por ubicarse en una esquina. **Centro-del-cubo-n:** equivalente al del cuarto. **Bloques-n:** requiere construir una torre específica sin saber como están los bloques. Ninguno de los problemas usa precondiciones. Sólo las redes de ordenamiento, el centro-del-cuarto y el centro-del-cubo admiten soluciones paralelas.

Reportamos los tiempos de compilación de la teoría a d–DNNF y los tiempos de búsqueda. Para cada problema reportamos los tiempos de búsqueda SAT con el horizonte correspondiente a la solución óptima N^*, y para el horizonte inmediato anterior $N^* - 1$. Esto permite evaluar la dificultad de probar que para los horizontes menores a $N^* - 1$ no hay solución, y que para N^* sí la hay.

En el Cuadro 1 mostramos algunos resultados de la compilación para horizontes óptimos en el caso serial. Esta contiene los tiempos de compilación a d–DNNF y el tamaño de los CNFs $T_{\text{cf}}(P)$ sobre los cuales se invoca al motor SAT. Algunos dominios generan CNFs muy grandes y otros pequeños. Esto no predice el comportamiento del motor SAT.

		teoría CNF		teoría d–DNNF			$T_{\text{cf}}(P)$	
problema	N^*	vars	clausulas	nodos	arcos	tiempo	vars-tc	clausulas-tc
anillo-r7	20	1081	3683	1008806	2179064	192.2	976203	3105362
anillo-r8	23	1404	4814	3887058	8340295	1177.1	3779477	11957085
bloques-b3	9	444	2913	5242	20229	0.3	4667	23683
bloques-b4	26	3036	40732	226967	888847	124.5	223260	1104383
centro-del-cuarto-e3	20	976	3642	11566	22081	1.1	9664	27956
centro-del-cuarto-e4	44	4256	16586	90042	174781	47.1	81404	238940
cubo-c9	33	2700	10350	282916	574791	98.9	276474	839027
cubo-c11	42	4191	16227	658510	1330313	371.6	647994	1958472
ordenamiento-s7	16	1484	6679	115258	283278	12.4	112756	390997
ordenamiento-s8	19	2316	12364	363080	895247	77.2	359065	1246236

Cuadro 1. Compilación para teorías secuenciales. N^* = horizonte óptimo. Nodos y arcos se refiere al DAG de los d–DNNF. El tiempo es el usando en compilación para horiz N^* (en segs). Vars-tc y clausulas-tc se refiere al tamaño de los CNFs que serán resueltos con el motor SAT.

En el Cuadro 2 mostramos los resultados de ejecutar el motor SAT sobre $T_{\text{cf}}(P)$ indicando el tiempo que tomó en encontrar la solución, tanto teorías seriales como paralelas. El principal resultado es que superamos el rendimiento

reportado en Palacios *et al.* [5], logrando resolver una instancia más en centro-del-cuarto y de ordenamiento, lo cual representa una *mejora de un orden de magnitud* en esos dominios. En los bloques, en cambio, el rendimiento es similar. En los anillos más grandes tenemos algunas dificultades cargando los CNFs generados. Nuestros resultados son comparables a los de Rintanen [4], sin que uno domine sobre el otro, aunque nuestro esquema es mucho más simple al usar sólo dos herramientas generales disponibles.

El planificador propuesto no es directamente comparable con los planificadores subóptimos. Aun así, tenemos mejor rendimiento que CFF [7] en problemas donde es necesario hacer implícitamente razonamientos epistemológicos, como centro-del-cuadro. Nuestro planificador no funciona bien en dominios más parecidos a planificación clásicas, donde CFF tiene mejor rendimiento (ver más abajo).

			búsqueda con horizonte k			b. con horizonte $k-1$	
problema	N^*	$\#S_0$	tiempo	decisiones	#act	tiempo	decisiones
teorías seriales							
anillo-r7	20	15309	° 2.1	2	20	° 0.8	0
anillo-r8	23	52488	> 1.8Gb			° 2.4	0
bloques-b3	9	13	0.1	1665	9	0.2	3249
bloques-b4	26	73	$> 2h$			$> 2h$	
centro-del-cuarto-e3	20	64	18.8	52037	20	207.4	207497
centro-del-cuarto-e4	44	256	5184.4	1096858	44	$> 2h$	
cubo-c7	24	343	3771.5	578576	24	5574.2	736567
cubo-c9	33	729	$> 2h$			$> 2h$	
ordenamiento-s5	9	32	0.0	352	9	22.0	35053
ordenamiento-s6	12	64	40.0	34451	12	$> 2h$	
ordenamiento-s7	16	128	3035.6	525256	16	$> 2h$	
ordenamiento-s8	19	256	$> 2h$			$> 2h$	
teorías paralelas							
centro-del-cuarto-e3	10	64	0.5	2737	20	0.3	1621
centro-del-cuarto-e4	22	256	423.1	244085	44	1181.5	439532
cube-c7	8	343	6.1	4442	24	2.9	1892
cubo-c9	11	729	114.6	27058	33	156.0	32760
cubo-c11	14	1331	> 1.8Gb			181.5	13978
ordenamiento-s7	6	128	46.1	18932	18	355.4	48264
ordenamiento-s8	6	256	° 4256.6	533822	23	$> 2h$	

Cuadro 2. Datos para consultas al motor SAT sobre teorías con horizonte óptimo N^* (izq) y subóptimo $N^* - 1$ (derecha). Se da el horizonte, el numero de estados iniciales, el tiempo que tomó el SAT, el número de decisiones que hizo y el número de acciones en el plan. Entradas con '$> 2h$' significa que agoto el tiempo. Con ° indica que el motor usado fue *zChaff*. Tiempos en segundos.

7. Discusión

En este trabajo hemos presentado un **esquema** compilar-proyectar-SAT para obtener planes conformantes. El esquema es simple al usar herramientas disponibles tanto para compilar a d–DNNF como para verificar si la fórmula obtenida es satisfactible. Los resultados son competitivos y necesariamente mejorables.

La existencia de un plan conformante con horizonte polinomial es Σ_2^p [1], cuya problema representivo (como SAT para NP-completo) es un QBF (*Quantified Boolean Formula*) de la forma $\exists x \forall y \exists z \phi$. Un trabajo previo de Rintanen [12] resuelve planificación conformante por medio de QBFs de la forma: $\exists \text{plan}\, \forall s_0\, \exists F_i\ \phi$. Existen motores QBF que expanden los cuantificadores y simplifican la teoría hasta obtener un CNF .

Actualmente exploramos una variación del **esquema** compilar-proyectar-SAT que pueda lidiar más fácilmente con problemas más cercanos a planificación clásica como los considerados por CFF, cuando se tiene un número limitado de estados iniciales. En tal caso la compilación no luce conveniente, y una proyección basada en la introducción de variables adicionales puede que sea preferible. Es interesante que este enfoque alternativo se reduce a una sola llamada a SAT en el caso de un sólo estado inicial, como SATPLAN en la planificación clásica.

Referencias

[1] Turner, H.: Polynomial-length planning spans the polynomial hierarchy. In Flesca, S., Greco, S., Leone, N., Ianni, G., eds.: JELIA. Volume 2424 of Lecture Notes in Computer Science., Springer (2002) 111–124

[2] Ferraris, P., Giunchiglia, E.: Planning as satisfiability in nondeterministic domains. In: Proceedings AAAI-2000. (2000) 748–753

[3] Bryant, R.E.: Symbolic Boolean manipulation with ordered binary-decision diagrams. ACM Computing Surveys **24** (1992) 293–318

[4] Rintanen, J.: Distance estimates for planning in the discrete belief space. In: Proc. AAAI-04. (2004) 525–530

[5] Palacios, H., Bonet, B., Darwiche, A., Geffner, H.: Pruning conformant plans by counting models on compiled d-DNNF representations. In: Proc. of the 15th Int. Conf. on Planning and Scheduling (ICAPS-05), AAAI Press (2005) To appear.

[6] Darwiche, A., Marquis, P.: A knowledge compilation map. Journal of Artificial Intelligence Research **17** (2002) 229–264

[7] Brafman, R., Hoffmann, J.: Conformant planning via heuristic forward search: A new approach. In: Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04). (2004)

[8] Kautz, H., Selman, B.: Pushing the envelope: Planning, propositional logic, and stochastic search. In: Proceedings of AAAI-96, AAAI Press / MIT Press (1996) 1194–1201

[9] Lin, F., Reiter, R.: Forget it! In: Working Notes, AAAI Fall Symposium on Relevance, American Association for Artificial Intelligence (1994) 154–159

[10] Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press (2000)

[11] Cimatti, A., Roveri, M.: Conformant planning via symbolic model checking. Journal of Artificial Intelligence Research **13** (2000) 305–338

[12] Rintanen, J.: Constructing conditional plans by a theorem-prover. Journal of Artificial Intelligence Research **10** (1999) 323–352