Agentes inteligentes cooperativos para planificación en paralelo

Diego García Paola Moguillansky Alejandro García

Departamento de Ciencias de la Computación
Universidad Nacional del Sur
Av. Alem 1253, (8000) Bahía Blanca, Argentina
agarcia@cs.uns.edu.ar

Resumen

En este trabajo se presenta un sistema de planificación donde dos agentes, trabajando en paralelo, cooperan para obtener una secuencia de acciones (plan) que permita llegar desde un estado inicial a un estado final. Cada agente utiliza un método diferente para planificar. Uno de ellos planifica "hacia adelante", partiendo del estado inicial y avanzando hacia el estado final. El otro agente planifica "hacia atrás", comenzando desde el estado final y retrocediendo hacia el estado inicial. La tarea de planificación termina cuando ambos agentes obtienen un plan parcial para llegar desde su estado de partida a un mismo estado intermedio.

Cada agente decide el curso a tomar, en cada instancia de la planificación, utilizando una función heurística basada en el estado del otro agente. Un mecanismo de comunicación entre los agentes permite que estos intercambien información sobre los estados alcanzados. El plan completo se construye a partir de los planes parciales generados individualmente por cada uno de los agentes. De esta manera, se logra reducir el tiempo y el espacio de búsqueda asociado a la planificación.

Palabras claves: Inteligencia Artificial, Agentes Inteligentes, Planificación

1 Introducción

El objetivo de este trabajo es desarrollar un sistema de planificación donde dos agentes, trabajando en paralelo, cooperan para obtener un plan. Cada agente utiliza un método diferente para planificar. Uno de ellos planifica "hacia adelante", partiendo del estado inicial y avanzando hacia el estado final. El otro agente planifica "hacia atrás", comenzando desde el estado final y retrocediendo hacia el estado inicial. La tarea de planificación termina cuando ambos agentes obtienen un plan parcial para llegar desde su estado de partida a un mismo estado intermedio. Viendo a la planificación como una búsqueda, el sistema aquí desarrollado realiza una *búsqueda bidireccional* [3].

En cada instancia de la planificación, cada agente decide el curso a tomar, utilizando una función heurística basada en el estado del otro agente. Un mecanismo de comunicación entre los agentes permite que estos intercambien información sobre los estados alcanzados. El plan completo se construye a partir de los planes parciales generados individualmente por cada uno de los agentes. De esta manera, se logra reducir el tiempo y el espacio de búsqueda asociado a la planificación. A pesar que la coordinación de los agentes insume tiempo, la sobrecarga estará

limitada a un tiempo constante. Dicho de otra manera, si sólo uno de los agentes es ejecutado, entonces hará una búsqueda uni-direccional mas la sobrecarga de observar siempre el mismo estado del otro agente. La concurrencia también permite que el sistema sea más robusto, ya que aunque uno de los agentes falle, el otro continuará la búsqueda independientemente.

A fin de estudiar la implementación de agentes inteligentes que planifican en forma conjunta, se eligió un dominio restringido y bien estudiado de la literatura, conocido como el "mundo de bloques". Los métodos que utilizan los agentes para planificar en forma concurrente fueron desarrollados utilizando las ideas de STRIPS. Sin embargo, ambos definen una manera particular de recorrer el espacio de búsqueda, eligiendo las acciones a realizar en base a una heurística que tiene en cuenta el estado del otro agente. Gracias a que STRIPS posee una manera general de representar los estados y describir las acciones a partir de precondiciones y poscondiciones, las heurísticas fueron definidas de forma general y pueden ser utilizadas para otros dominios de planificación.

El sistema de planificación aquí descripto fue implementado en *Jinni* (Java INference engine and Networked Interactor). La implementación de la comunicación entre los agentes de planificación se realizó utilizando un blackboard remoto donde los agentes dejan y recuperan información de diferente tipo. Al implementar los métodos de planificación concurrente, se detectaron ciertos aspectos en la elección de las acciones, que permitieron realizar optimizaciones en ambos métodos. Una de estas optimizaciones es aplicable también a STRIPS.

El presente trabajo está organizado de la siguiente manera. A continuación se presenta el dominio de planificación utilizado y se describe brevemente los elementos de STRIPS que se utilizarán en la implementación de la planificación concurrente. En la sección 3 se presentan los métodos de planificación concurrente desarrollados. Luego, en la sección 4 se detallan las heurísticas definidas para los agentes. La sección 5 presenta la implementación de la comunicación entre agentes, y en la sección 6 se introducen optimizaciones realizadas a la implementación. Finalmente en la sección 7 se describen las conclusiones obtenidas.

2 Planificación con STRIPS en el "Mundo de Bloques"

A fin de estudiar la implementación de agentes inteligentes que planifican en forma conjunta, se eligió un dominio restringido y bien estudiado de la literatura, conocido como el "mundo de bloques". En el mundo de bloques se dispone de un conjunto de bloques o cubos etiquetados con letras y una mesa donde estos pueden moverse o apilarse encima de otros (ver Figura 1). Inicialmente los bloques están dispuestos sobre la mesa de alguna manera particular que denominaremos *estado inicial*, y el objetivo es encontrar una secuencia de acciones (*plan*), que lleve a otra disposición o situación especial llamada *estado final*.

La representación de una situación o un estado depende del sistema de planificación que se utilice. En nuestro caso, utilizaremos como punto de partida para nuestro desarrollo a STRIPS, un planificador de propósito general, simple y bien estudiado. A continuación describiremos brevemente la representación de estados, las acciones y el funcionamiento de STRIPS. Un *estado* o *situación* en el mundo de bloques se definirá utilizando las siguientes *condiciones* o *descriptores*:

- $clear(X)$: Indica que el bloque X no tiene bloques sobre él.

- $on(X, Y)$: Indica que el bloque X se encuentra directamente sobre el bloque Y.
- $ontable(X)$: Indica que el bloque X esta sobre la mesa.

Una situación o estado S será descripto simplemente indicando el conjunto de condiciones que se cumplen en S. Por ejemplo el estado inicial de la Figura 1 se describe con el conjunto $\{ontable(A)$, $ontable(B)$, $ontable(C)$, $clear(A)$, $clear(B)$, $clear(C)\}$, y el estado final con $\{ontable(A)$, $on(B, A)$, $on(C, B)\}$.

Figura 1: Ejemplos de estados inicial y final en el mundo de bloques

Cada dominio de planificación define un conjunto particular de *acciones* que pueden ejecutarse sobre un estado del dominio y permiten pasar a un nuevo estado. En STRIPS una acción a se describe en términos de un conjunto de precondiciones $PrecList(a)$ que deben cumplirse para que la acción pueda ser ejecutada en un estado, una lista de condiciones que se agregan al estado $AddList(a)$ y una lista de elementos que se eliminarán de la descripción del estado $DelList(a)$. Dada una situación S y una acción a, diremos que a *es posible en* S si y solo si $PrecList(a) \subseteq S$, es decir, toda precondición de la acción a se cumple en S. El resultado de ejecutar una acción posible en una situación S, es un nuevo estado $S' = (S - DelList(a)) \cup AddList(a)$

En nuestro caso particular del mundo de bloques, el conjunto de acciones será:

- $stack(A, B)$: mueve el bloque A, que se encuentra sobre la mesa, sobre el bloque B.
- $unstack(A, B)$: mueve el bloque A, que se encuentra sobre B, a la mesa.
- $move(A, B, C)$: mueve el bloque A, que se encuentra sobre B, sobre el bloque C.

Dichas acciones se describen en STRIPS como sigue:

a	$PrecList(a)$	$AddList(a)$	$DelList(a)$
$move(A, B, C)$	$clear(A)$ $on(A, B)$ $clear(C)$	$on(A, C)$ $clear(B)$	$on(A, B)$ $clear(C)$
$stack(A, B)$	$clear(B)$ $ontable(A)$ $clear(A)$	$on(A, B)$	$clear(B)$ $ontable(A)$
$unstack(A, B)$	$on(A, B)$ $clear(A)$	$ontable(A)$ $clear(B)$	$on(A, B)$

Dado un estado S, pueden existir varias acciones posibles en S que llevarán a otros estados. Esto genera un espacio de estados, donde la transición entre estados está dada por las acciones

posibles. STRIPS construye la secuencia de acciones, o plan, a partir del estado final y trabajando hacia atrás (backward chaining). Esto es, comenzando por la ultima acción y agregando acciones que logran las precondiciones de las acciones generadas anteriormente en el proceso de planificación (esto es, posteriores en el plan).

Describiremos a continuación el espacio de búsqueda asociado a STRIPS [2]. Los nodos del espacio de búsqueda, serán etiquetados con un par (P, S) donde el primer elemento del par es una secuencia de acciones, o plan parcial, para llegar desde S al estado final, y el segundo elemento es un conjunto de metas que no han sido alcanzadas aún. Dado un estado o situación inicial I, y un estado final o conjunto de metas F , el espacio de búsqueda de STRIPS asociado a I y F es el siguiente:

1. El nodo raíz del espacio de búsqueda es etiquetado con $([\,], F)$.
2. Un nodo $([a_1, a_2, \ldots, a_n], Z)$ es un *nodo meta*, si Z esta incluido en I, es decir, todas las metas restantes (Z) se cumplen en la situación inicial. En un nodo meta, la secuencia de acciones $[a_1, a_2, \ldots, a_n]$ corresponde al plan encontrado por STRIPS.
3. Sea $N = ([a_1, a_2, \ldots, a_n], Z)$ un nodo, y a una acción que logra al menos una meta del conjunto $(Z - I)$, es decir, que $AddList(a) \cap (Z - I) \neq \emptyset$, y que no borra ningún elemento de Z, esto es, $DelList(a) \cap Z = \emptyset$. Un nodo sucesor de N se obtiene de la siguiente manera: $([a, a_1, a_2, \ldots, a_n], (Z - AddList(a)) \cup PrecList(a))$. En el nuevo nodo, la acción a se ubica al frente de la secuencia de acciones, las metas que a agrega se quitan de Z y las precondiciones de a se agregan a Z como nuevas submetas.

   Cabe destacar, que la acción a es elegida de tal manera que logre metas de Z - I, es decir, no se consideran las metas de Z que ya se cumplen en el estado inicial I. Además dicha acción no debe borrar elementos de Z, a fin de evitar la elección de acciones que borren una meta que esperamos lograr antes en la secuencia de acciones (esto es, mas tarde en el proceso de planificación).

3 Planificación concurrente con agentes inteligentes

Como se dijo anteriormente, el objetivo de este trabajo es desarrollar un sistema de planificación donde dos agentes, trabajando de manera concurrente, cooperan para obtener un plan. Dado un estado inicial I y un estado final F del mundo de bloques, uno de los agentes busca un plan partiendo de F y el otro partiendo de I. Los agentes planifican en paralelo y se comunican para poder direccionar su planificación en función de una heurística de acercamiento. La tarea de planificación termina cuando ambos agentes obtienen un plan parcial para llegar desde su estado de partida a un mismo estado intermedio.

En lo que sigue del trabajo llamaremos $agente_1$ al agente que planifica "hacia atrás" partiendo del estado final F, y $agente_2$ al agente que realiza la planificación "hacia adelante" partiendo del estado inicial I. Para lograr encontrarse en un estado intermedio cada agente elige la acción que mas lo acerque al estado en que se encuentra el otro. Para esto cada agente comunica su último estado alcanzado S, y el otro agente utiliza una función heurística, basándose en dicho estado S, para elegir la próxima acción.

A continuación se describirá en detalle el método de planificación utilizado por cada agente. Para entender el funcionamiento de los mismos se debe tener en cuenta que ambos planifican

en forma concurrente, y que además existe un medio de comunicación a través del cual los agentes pueden intercambiar información sobre sus estados. La descripción se dará en forma independiente de la heurística y del mecanismo de comunicación. Sin embargo, en la sección 5 se describe un mecanismo de comunicación entre agentes, la cual fue adoptada para nuestra implementación, y puede utilizarse directamente en los algoritmos de planificación. Las heurísticas desarrolladas para la elección de las acciones serán introducidas en la sección 4.

3.1 STRIPS*: planificación concurrente hacia atrás

El $agente_1$ planifica partiendo desde el estado final F, hacia el estado inicial I. Básicamente utiliza el método de planificación de STRIPS, pero define una manera particular de recorrer el espacio de búsqueda, eligiendo las acciones a realizar en base a una heurística que tiene en cuenta el estado del otro agente. Llamaremos a este método de planificación STRIPS*.

En STRIPS*, al igual que en la descripción del espacio de búsqueda de STRIPS, un nodo será etiquetado con un par $(P, S$, donde P es un plan parcial, para llegar desde el estado S al estado final F. La función H_1 representará la heurística utilizada por el $agente_1$, y será explicada en detalle en la sección 4. A continuación describiremos el procedimiento utilizado por el $agente_1$ para recorrer el espacio de búsqueda, partiendo del estado final F:

1. Sea L una lista de nodos del espacio de búsqueda que contiene inicialmente al nodo raíz ([ ],F), y sea V un conjunto, inicialmente vacío, de nodos que han sido visitados por el $agente_1$.

2. Si el $agente_2$ ha informado que se ha encontrado un plan, el $agente_1$ da por finalizada su tarea de planificación.

3. Sea S_2 el último estado informado por el $agente_2$. Los agentes habrán encontrado un plan en forma conjunta, si en el conjunto V existe un nodo $([a_1, a_2, \ldots, a_n], S)$, visitado por el $agente_1$, tal que $S \subseteq S_2$. Si el plan parcial obtenido por el $agente_2$ para llegar a S_2 es $[a'_1, a'_2, \ldots, a'_n]$, entonces el plan completo se forma concatenando $[a'_1, a'_2, \ldots, a'_n] + [a_1, a_2, \ldots, a_n]$. En este caso el $agente_1$ informa que se ha encontrado el plan, y finaliza la búsqueda. Los estados S y S_2 se denominan puntos de encuentro.

4. En el caso que no se halla encontrado un plan en forma conjunta, el $agente_1$ debe elegir las acciones a realizar. Sea S_2 el último estado informado por el $agente_2$. Extraer de la lista L el primer nodo $N = ([a_1, a_2, \ldots, a_n], S_1)$, e insertarlo en V para registrarlo como visitado. Sea $B = \{b_i \mid AddList(b_i) \cap (S_1 - S_2) \neq \emptyset \wedge DelList(b_i) \cap S_1 = \emptyset\}$ el conjunto de todas las acciones b_i que logran al menos una meta del conjunto $(S_1 - S_2)$, y no borran ningún elemento de S_1. Cada acción $b_i \in B$, $1 \leq i \leq K$ genera un nodo hijo de N, $N_i = ([b_i, a_1, a_2, \ldots, a_n], S_{b_i})$, donde $S_{b_i} = (S_1 - AddList(b_i)) \cup PrecList(b_i)$. Los nodos hijos $N_i = (P, S_i)$ cuyo estado no fue visitado, es decir, no existe en V un elemento (P', S_i), se insertan al comienzo de la lista L, ordenados en forma decreciente de acuerdo al valor de la heurística $H_1(b_i, S_2)$.

5. Si L no está vacía retornar al paso 2, de lo contrario el $agente_1$ finaliza la tarea de planificación sin éxito.

Los pasos 2 y 3 evalúan si se ha encontrado un un plan de manera conjunta. Obsérvese que en el paso 3 es necesario considerar todos los nodos N ya visitados por el $agente_1$, a fin de asegurar que si existe un punto de encuentro, este no será pasado por alto. En el paso 4 del algoritmo se decide cuál es la próxima acción en la planificación. Como el método planifica hacia atrás, en realidad busca el conjunto de acciones (B) que lo llevan al estado actual. La heurística H_1 es utilizada para elegir la mejor de dichas acciones a fin de proseguir la búsqueda en dirección al estado en que se encuentra el $agente_2$.

3.2 F-PLAN: planificación concurrente hacia adelante

La planificación concurrente se completa utilizando un segundo agente que interactúa con el $agente_1$. A diferencia del anterior, el $agente_2$ planifica partiendo desde el estado inicial I, hacia el estado final F, utilizando un método de planificación concurrente hacia adelante que llamaremos F-PLAN. Como el $agente_2$ planifica hacia adelante, si se encuentra en un estado S, elige la próxima acción entre las acciones cuyas precondiciones se cumplen en S (esto es, acciones que pueden ejecutarse en S). A fin de acercarse al estado en que se encuentra el $agente_1$, el $agente_2$ elegirá la acción que maximice una función heurística H_2 que considera el estado del otro agente. La función H_2 será explicada en detalle en la sección 4.

En F-PLAN, un nodo será etiquetado con un par (P, S), donde P es un plan parcial, para llegar desde el estado inicial I al estado S. A continuación describiremos el procedimiento utilizado por el $agente_2$ para recorrer el espacio de búsqueda, partiendo del estado inicial I.

1. Sea L una lista de nodos del espacio de búsqueda que contiene inicialmente al nodo raíz ([ ],I), y sea V un conjunto, inicialmente vacío, de nodos que han sido visitados por el $agente_2$.

2. Si el $agente_1$ ha informado que se ha encontrado un plan, el $agente_2$ da por finalizada su tarea de planificación.

3. Sea S_1 el último estado informado por el $agente_1$. Los agentes habrán encontrado un plan en forma conjunta, si en el conjunto V existe un nodo $([a_1, a_2, \ldots, a_n], S)$, visitado por el $agente_2$, tal que $S_1 \subseteq S$. Si el plan parcial obtenido por el $agente_1$ para llegar a S_1 es $[a'_1, a'_2, \ldots, a'_n]$, entonces el plan completo se forma concatenando $[a_1, a_2, \ldots, a_n] + [a'_1, a'_2, \ldots, a'_n]$. En este caso el $agente_2$ informa que se ha encontrado el plan, y finaliza la búsqueda. Los estados S y S_1 se denominan puntos de encuentro.

4. En el caso que no se halla encontrado un plan en forma conjunta, el $agente_2$ debe elegir las acciones a realizar. Sea S_1 el último estado informado por el $agente_1$. Extraer de la lista L el primer nodo $N = ([a_1, a_2, \ldots, a_n], S_2)$, e insertarlo en V para registrarlo como visitado. Sea $B = \{b_i | PrecList(b_i) \subseteq S_2\}$ el conjunto de todas las acciones posibles en S_2, es decir, todas las acciones b_i cuyas precondiciones se cumplen en S_2. Cada acción $b_i \in B$, $1 \leq i \leq K$ genera un nodo hijo de N, $N_i = ([a_1, a_2, \ldots, a_n, b_i], S_{b_i})$, donde $S_{b_i} = S_2 - DelList(b_i) \cup AddList(b_i)$. Los nodos hijos $N_i = (P, S_i)$ cuyo estado no fue visitado, es decir, no existe en V un elemento (P', S_i), se insertan al comienzo de la lista L, ordenados en forma decreciente de acuerdo al valor de la heurística $H_2(b_i, S_1)$.

5. Si L no está vacía retornar al paso 2, de lo contrario el $agente_1$ finaliza la tarea de planificación sin éxito.

4 Heurísticas de acercamiento

A continuación se definirán las heurísticas H_1 y H_2 utilizadas por los agentes para decidir que acción elegir en su búsqueda del plan. Ambas heurísticas fueron definidas de forma general a partir de las precondiciones ($Preclist$) y poscondiciones ($Addlist$) de las acciones. De esta manera, las heurísticas podrán ser utilizadas para otros dominios de planificación.

Las heurísticas definidas miden el grado de similitud entre el estado S_a generado por la ejecución de una acción a a partir del estado S de uno de los agentes, y el estado actual S' del otro agente. Dicho de otra manera, miden el grado de acercamiento al estado S', que produce la ejecución de una acción a sobre el estado S.

Heurística H_1

Sea a una acción y S_2 el estado en que se encuentra el $agente_2$. En el caso del $agente_1$ (STRIPS*), la heurística H_1 se define de la siguiente manera:

$$H_1(a, S_2) = |PrecList(a) \cap S_2| - |AddList(a) \cap S_2|.$$

Esto es, la cantidad de precondiciones de a que se cumplen en S_2 menos la cantidad de post-condiciones de a presentes en S_2.

Figura 2: Heurística de acercamiento

Ejemplo 4.1 : Supóngase que el $agente_1$ se encuentra en el estado $S_1 = \{ontable(A), on(B, A), clear(B), ontable(C), clear(C)\}$ (ver Figura 2), y que el $agente_2$ se encuentra en el estado $S_2 = \{ontable(A), ontable(B), ontable(C), clear(A), clear(B), clear(C)\}$ El objetivo de ambos agentes es encontrar un plan que los lleve al estado del otro. El conjunto de acciones consideradas por el $agente_1$ en el estado S_1 (ver paso 4 de STRIPS*) es $B = \{stack(B, A), move(B, C, A), unstack(C, B)\}$. A continuación figura el resultado de aplicar la heurística H_1 a cada una de estas acciones con respecto al estado S_2.

$H_1(stack(B, A), S_2) =$
$= |PrecList(stack(B, A)) \cap S_2| - |AddList(stack(B, A)) \cap S_2|$
$= |\{clear(B), clear(A), ontable(B)\} \cap S_2| - |\{on(B, A)\} \cap S_2|$
$= |\{clear(B), clear(A), ontable(B)\}| - |\{\}| = 3 - 0 = 3$

$H_1(move(B, C, A), S_2) =$
$= |PrecList(move(B, C, A)) \cap S_2| - |AddList(move(B, C, A)) \cap S_2|$
$= |\{clear(B), clear(A), on(B, C)\} \cap S_2| - |\{on(A, B), clear(C)\} \cap S_2|$
$= |\{clear(B), clear(A)\}| - |\{clear(C)\}| = 2 - 1 = 1$

$H_1(unstack(C,B), S_2) =$
$= |PrecList(unstack(C,B)) \cap S_2| - |AddList(unstack(C,B)) \cap S_2|$
$= |\{clear(C), on(C,B)\} \cap S_2| - |\{ontable(C), clear(B)\} \cap S_2|$
$= |\{clear(C)\}| - |\{clear(B)\}| = 1 - 1 = 0$

Como puede verse, la heurística indica que la acción $stack(B,A)$ produce un mejor acercamiento al estado S_2, que la acción $unstack(C,B)$ o $move(B,C,A)$. De esta forma si el $agente_1$ selecciona la acción $stack(B,A)$ para aplicar sobre S_1, su próximo estado estaría dado por: $S_1 - (AddList(stack(B,A)) \cup PrecList(stack(B,A)) = \{ontable(A), ontable(B), ontable(C), clear(A), clear(B), clear(C)\}$.

En este caso, la acción $stack(B,A)$ aplicada sobre S_1 lo lleva al $agente_1$ directamente al estado S_2 del $agente_2$, lo cual significa que los agentes se encontraron en un estado intermedio y que han encontrado un plan.

Heurística H_2

Sea a una acción y S_1 el estado actual del $agente_1$, en el caso del $agente_2$ (F-PLAN), la heurística H_2 se define como la cantidad de elementos de $AddList(a)$ que se cumplen en S_1 menos la cantidad de elementos de $DelList(a)$ presentes en S_1.

$$H_2(a, S_1) = |AddList(a) \cap S_1| - |DelList(a) \cap S_1|$$

Ejemplo 4.2 : Sean $S_1 = \{ ontable(A), on(B,A), clear(B) \}$ y $S_2 = \{ ontable(A), ontable(B), clear(A), clear(B)\}$ el estado actual del $agente_1$ y el $agente_2$ respectivamente (ver Figura 3).

Figura 3: Heurística de acercamiento

Las acciones posibles en el estado S_2 son $stack(B,A)$ y $stack(A,B)$:

- $stack(B,A)$, ya que $PrecList(stack(B,A)) = \{ontable(B), clear(B), clear(A)\} \subseteq S_2$
- $stack(A,B)$, ya que $PrecList(\text{stack(A,B)}) = \{ontable(A), clear(B), clear(A)\} \subseteq S_2$

Al aplicar la función H_2 sobre ambas acciones resulta:

$$\begin{aligned} H_2(stack(B,A), S_2) &= |AddList(stack(B,A)) \cap S_1| - |DelList(stack(B,A)) \cap S_1| \\ &= |\{on(B,A)\} \cap S_1| - |\{clear(A), ontable(B)\} \cap S_1| \\ &= |\{on(B,A)\}| - |\{\}| = 1 - 0 = 1 \end{aligned}$$

$$\begin{aligned} H_2(stack(A,B), S_2) &= |AddList(stack(A,B)) \cap S_1| - |DelList(stack(A,B)) \cap S_1| \\ &= |\{on(A,B)\} \cap S_1| - |\{clear(B), ontable(A)\} \cap S_1| \\ &= |\{\}| - |\{clear(B), ontable(A)\}| = 0 - 2 = -2 \end{aligned}$$

Esto indica que la acción $stack(B,A)$ en el estado S_2 produce un mejor acercamiento al estado S_1 que la acción $stack(A,B)$. De esta forma si el $agente_2$ selecciona la acción $stack(B,A)$, su próximo estado estaría dado por : S_2 - $DelList(stack(B,A) + AddList(stack(B,A)) = \{ontable(A), on(B,A), clear(B)\}$.

5 Implementación y comunicación entre agentes

El sistema de planificación aquí descripto fue implementado en *Jinni* (Java INference engine and Networked Interactor). Jinni es un lenguaje de programación en lógica "multi-threaded" diseñado especialmente como una plataforma para el desarrollo de aplicaciones con agentes inteligentes autónomos. Jinni provee una forma de comunicación y coordinación entre agentes utilizando *blackboards*. Los blackboards funcionan como servidores locales o remotos donde los agentes pueden consultar o insertar (assert) términos con la sintaxis de Prolog. Referimos al lector interesado en los detalles de Jinni a [5].

La implementación de la comunicación entre los agentes de planificación se realizó utilizando un blackboard remoto donde los agentes dejan y recuperan información de diferente tipo. A continuación se definen las primitivas que fueron utilizadas para el intercambio de información a través del blackboard.

1. $Informar(A, S)$: mediante esta primitiva un agente A deja en el blackboard su estado actual S. Como efecto secundario, el estado anterior informado por el agente A (si existe) es quitado del blackboard. De esta manera en el blackboard siempre se encuentra sólo el estado actual de cada agente.
2. $Recuperar(A, S)$: consulta al blackboard por el último estado S informado por el agente A. Utilizando esta primitiva un agente puede conocer el estado en que se encuentra el otro agente.
3. $InformarPlanEncontrado$: registra en el blackboard que se ha encontrado un plan.
4. $ObtenerPlanParcial(A, S, P)$: obtiene el plan parcial P generado por el agente A para llegar al estado S. Mediante esta primitiva un agente puede armar el plan completo utilizando el plan parcial que él a construído y el plan parcial construído por el otro agente.

6 Optimizaciones

Al implementar los métodos de planificación concurrente, se detectaron ciertos aspectos en la elección de las acciones, que permitieron realizar optimizaciones en ambos métodos. Para F-PLAN se desarrolló un criterio adicional para poder elegir entre acciones que tengan el mismo valor al aplicar la función heurística. En el caso de STRIPS*, puede ocurrir que se elijan acciones que produzcan un estado absurdo o inconsistente. En la sección 6.2 se define un criterio para evitar caer en dicho problema.

6.1 Elección de acciones con igual valor heurístico

Al aplicar la función heurística a un conjunto de acciones B (paso 4 de los métodos de búsqueda), puede darse el caso que no exista una única acción cuyo valor sea máximo. Es decir, que exista un subconjunto $B' \subseteq B$ de acciones que tengan el máximo valor. En este caso cualquiera de las acciones de B' podría ser elegida como la próxima acción en el plan. En F-PLAN, en lugar de elegir una de las acciones de B' al azar, se definió un criterio para decidir entre ellas.

El criterio consiste en seleccionar (si es que existe) una acción a que al ser ejecutada permita que el resto de las acciones de B' se realicen posteriormente, es decir que no elimine ninguna precondición de estas acciones. De esta manera se elegirá una acción $a \in B'$ tal que $\forall b \in B' - \{a\}$, se cumpla que $DelList(a) \cap PrecList(b) = \emptyset$

Ejemplo 6.1 : Sean $S_2 = \{ontable(A),\ ontable(B),\ ontable(C),\ clear(A),\ clear(B),\ clear(C)\}$ y $S_1 = \{ontable(A),\ on(B,A),\ on(C,B),\ clear(C)\}$, los estados actuales del $agente_2$ y el $agente_1$ respectivamente (ver Figura 4).

Figura 4: Estados del $agente_2$ y del $agente_1$

El conjunto de acciones consideradas por el $agente_2$ en el estado S_2 es: $\{stack(B,A), stack(A,B), stack(A,C), stack(B,C), stack(C,A), stack(C,B)\}$. Las acciones con mayor valor en la heurística son: $stack(C,B)$ y $stack(B,A)$, con $H_2(stack(C,B), S_1) = H_2(stack(B,A), S_1) = 1$. Ambas tienen el mismo valor de heurística y cualquiera podría ser elegida como la próxima acción. Sin embargo, si se elige la acción $stack(C,B)$ (ver Figura 5-a), se impedirá la elección posterior de la acción $stack(B,A)$, ya que su ejecución en S_2 elimina precondiciones necesarias para poder ejecutar $stack(B,A)$ en el estado resultante: $DelList(stack(C,B)) \cap PrecList(stack(B,A)) = \{clear(B)\}$

Figura 5: (a) ejecutando $stack(C,B)$ en S_2, (b) ejecutando $stack(B,A)$ en S_2

Si en cambio el criterio anterior es utilizado para elegir entre las acciones, se elegirá $stack(B,A)$, ya que esta no impide que se realicen posteriormente el resto de las acciones con igual valor en la heurística, pues no elimina ninguna precondición de estas: $DelList(stack(B,A)) \cap PrecList(stack(C,B)) = \emptyset$. El resultado de ejecutar $stack(B,A)$ puede verse en la Figura 5-b.

6.2 Acciones que producen estados inconsistentes

Al planificar utilizando el esquema de STRIPS (como en el caso de STRIPS*) pueden llegar a elegirse acciones que produzcan estados inconsistentes. Explicaremos esta situación mediante el siguiente ejemplo.

Ejemplo 6.2 : Considérese el estado $S = \{ontable(A), on(B,A), on(C,B), clear(C)\}$ (ver Figura 6) las acciones que STRIPS puede elegir son:

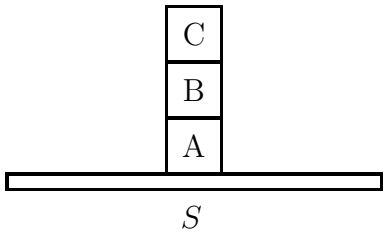

Figura 6: un estado en la planificación con strips

- $stack(C, B)$, ya que $AddList(stack(C, B)) \cap S \neq \emptyset$ y $DelList(stack(C, B)) \cap S = \emptyset$
- $stack(B, A)$, ya que $AddList(stack(B, A)) \cap S \neq \emptyset$ y $DelList(stack(B, A)) \cap S = \emptyset$

En caso de elegir la acción $stack(B, A)$, el estado S' resultante sería:
$$\begin{aligned} S' &= (S - AddList(stack(B, A))) \cup PrecList(stack(B, A)) = \\ &= (S - \{on(B, A)\}) \cup \{ontable(B), clear(B), clear(A)\} = \\ &= \{ontable(A), on(C, B), clear(C), clear(B), clear(A), ontable(B)\} \end{aligned}$$

Si se observa con atención, las condiciones $clear(B)$ y $on(C, B)$ están presentes en el nuevo estado S', lo cual resulta inconsistente, ya que si se cumple $on(B, C)$, entonces no puede cumplirse $clear(B)$. Dicho de otra manera, S' es un estado donde B no tiene bloques encima y C se encuentra sobre B.

Este fenómeno se produce debido que se selecciona una acción (en el ejemplo $stack(A, B)$) para lograr una meta incorrecta (en este caso, $on(B, A)$), es decir, una meta que debería lograse antes en el plan (más tarde en el proceso de planificación). Cabe recordar que las acciones en STRIPS se eligen en función de las metas que estas logran ($AddList$) y las metas que no han sido logradas aún.

Para evitar la elección de acciones que lleven a estados inconsistentes, se deben elegir primero las metas adecuadas y luego, a partir de estas, seleccionar las acciones que las cumplen. Sea $P = \{a_i | PrecList(a_i) \subseteq S\}$, el conjunto de todas las acciones cuyas precondiciones se cumplen en S. El conjunto de metas adecuadas M se calcula como la unión de las precondiciones de todas las acciones pertenecientes al conjunto P.

Considerando el estado S descripto en el ejemplo 6.2 se obtiene el conjunto $P = \{a_i | PrecList(a_i) \subseteq S\} = \{unstack(C, B)\}$, con lo cual, el conjunto de metas adecuadas resulta $M = PrecList(unstack(C, B)) = \{clear(C), on(C, B)\}$. Utilizando ahora el conjunto de metas adecuadas M para la elección de las acciones, puede verse que la acción $stack(A, B)$ no será elegida, ya que $Addlist(stack(A, B)) = \{on(B, A)\} \cap M = \emptyset$.

7 Conclusiones y trabajo futuro

El método de planificación propuesto resulta mas robusto que un sistema con un solo agente, ya que aunque uno de los agentes se detenga, el otro continuará con la planificación hacia el último estado informado por el agente detenido. Planificando en paralelo se reduce el espacio de búsqueda y el tiempo asociado a la planificación. La coordinación de los agentes insume tiempo, pero la sobrecarga estará limitada a un tiempo constante. Dicho de otra manera, si sólo uno

de los agentes es ejecutado, entonces hará una búsqueda uni-direccional mas la sobrecarga de observar siempre el mismo estado del otro agente. Aunque el método se desarrolló para el mundo de bloques, puede adaptarse fácilmente a otros dominios que permitan definir las acciones como STRIPS. Las heurísticas y las optimizaciones también fueron descriptas de manera general y pueden aplicarse a otros dominios de planificación.

El estado en que se encuentra uno de los agentes permite guiar las acciones del otro, pero desafortunadamente, también puede desviarlo produciendo un plan poco satisfactorio (con acciones redundantes) e incrementando el tiempo de planificación. Por lo tanto, no hay garantía de encontrar un plan óptimo con respecto a la cantidad de acciones. Como trabajo futuro se tiene pensado considerar dos o más agentes que planifican partiendo del mismo estado. Por ejemplo, varios agentes A_i^F podrían planificar partiendo del estado inicial y avanzando hacia el estado final. Para evitar que todos los agentes sigan el mismo camino, ya que todos elegirían la (misma) mejor opción, los agentes pueden comunicarse y no tomar una acción que tomó otro. De esta forma, la probabilidad de tomar un camino equivocado disminuye dramáticamente (ver [6]). Una dirección a explorar es combinar un conjunto de agentes A_i^F que planifican hacia adelante, y un conjunto de agentes A_i^B que planifican hacia atrás. De esta forma se mejoraría el tiempo de búsqueda, evitando que uno de los agentes desvíe la planificación del otro. Otra solución posible para evitar que los agentes se desvíen, es tener un tercer agente que estime el punto de encuentro entre los agentes, de esta manera, los agentes planificadores podrían tomar este punto de encuentro para direccionar su búsqueda.

Referencias

[1] Edmund Durfee. Distributed problem solving and planning. In Gerhard Weiss, editor, *Multiagent Systems, a Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.

[2] Matt Ginsberg. *Essentials of Artificial Intelligence*. Morgan Kaufmann Publishers, Inc., 1993.

[3] T. Ishida. *Real-time search for learning autonomous agents*. Kluwer Academic Publishers, 1997.

[4] T. Ishida and R. E. Korf. Moving target search. In *Proc. of the 12th IJCAI*, pages 204–210, Sidney, Australia, 1991.

[5] Paul Tarau. Intelligent Mobile Agent Programming at the Intersection of Java and Prolog. In *Proceedings of The Fourth International Conference on The Practical Application of Intelligent Agents and Multi-Agents*, pages 109–123, London, U.K., 1999.

[6] M. Yokoo and T. Ishida. Search algorithms for agents. In Gerhard Weiss, editor, *Multiagent Systems, a Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.