UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Cálculo distribuido de landmarks para sistemas de planificación multiagente

Proyecto Final de Carrera

Ingeniería Informática

Autor: Ana Oropesa Física

Director: Eva Onaindia de la Rivaherrera

Codirector: Alejandro Torreño Lerma

27 de septiembre de 2013

A mis padres,
y a todas esas personas que tras cinco años
me han acompañado en esta aventura.

Resumen

En este Proyecto Final de Carrera se verá la motivación por la que hacer una heurística multiagente utilizando landmarks, la construcción de ésta y unos posteriores resultados y comparativas con la heurística monoagente entre otras.

Índice general

Capítulo 1

Introducción

El proyecto desarrollado se enmarca en el área de la Inteligencia Artificial y más concretamente en la planificación multiagente. En este capítulo veremos la motivación del trabajo, los objetivos a conseguir y un resumen de como está estructurado el documento.

1.1. Motivación

El inicio del proyecto surge del desarrollo e investigación, llevado a cabo por el grupo de planificación del DSIC (GRPS-AI), del modelo de planificación multiagente FMAP, el cual integra la generación y coordinación de planes entre varios agentes. En FMAP, los agentes crean planes mediante un planificador completo de orden parcial hacia delante, guiado por una heurística que está basada en información sobre los cambios de las variables estado en un dominio de planificación, y optimizada para evaluar planes en entornos con información incompleta o privada.

En la versión monoagente de FMAP se utiliza una heurística basada en landmarks (conjunto de proposiciones que tienen que cumplirse en un plan para que éste sea un plan solución del problema). De este modo, surge la idea del presente proyecto, implementar una versión multiagente de la heurística de landmarks y comparar los resultados con la heurística que actualmente está implementada en FMAP.

Obtener una heurística multiagente tiene una dificultad añadida por la coordinación de los agentes, ya que esto supone muchas comunicaciones simultáneas. Por ello, se decide integrar todo el proyecto dentro de la platafor-

ma PlanInteraction [1] (también del grupo de planificación del DSIC), donde la coordinación y gestión de información entre los agentes se puede realizar de una forma más sencilla.

Así pues, el proyecto consta de dos fases: implementar una versión distribuida del algoritmo de cálculo del grafo de landmarks y una vez obtenido dicho grafo para cada uno de los agentes participantes en el problema, implementar una versión multiagente de la heurística basada en landmarks del sistema de planificación LAMA [RW10]. Finalmente, todo ello se integrará dentro de la plataforma PlanInteraction.

1.2. Objetivos

El primer objetivo de este proyecto es diseñar e implementar una heurística multiagente basada en landmarks. Después, compararemos el rendimiento de la heurística diseñada con la heurística multiagente no basada en landmarks de FMAP.

El segundo objetivo de este proyecto es integrar FMAP junto con la nueva heurística diseñada, que desde ahora llamaremos FMAP-land, en la plataforma PlanInteraction. El próposito de esta integración es simplificar y compactar la gran cantidad de comunicaciones entre agentes, lo cual facilitará la implementación.

1.3. Organización del documento

En esta sección, se mostrarán los temas a tratar en cada capítulo para ofrecer una idea general de la estructura del documento. El presente documento se divide en los siguientes capítulos:

- **Capítulo 1: Introducción**

  En este capítulo se verá una introducción sobre el proyecto, su motivación, sus objetivos y un resumen de los temas a tratar.

[1] http://servergrps.dsic.upv.es/planinteraction/

- **Capítulo 2: Estado del Arte**

  Aquí se tratará con más profundidad los antecedentes de la planificación como área dentro de la Inteligencia Artificial, la utilización de landmarks en planificación, y la heurística basada en landmarks del planificador LAMA.

- **Capítulo 3: FMAP y PlanInteraction**

  En esta parte se comentará en profundidad los sistemas en los que está basado el proyecto: el planificador FMAP y la plataforma PlanInteraction.

- **Capítulo 4: Diseño de FMAP-land**

  En este capítulo se hablará del diseño FMAP-land y su uso de la privacidad, además de su implementación multiagente y el cálculo final de la heurística.

- **Capítulo 5: Resultados**

  Aquí se comentarán los resultados obtenidos de la extracción de landmarks en problemas de los dominios Satellite y Rovers, benchmarks utilizados en la Competición Internacional de Planning [2]

- **Capítulo 6: Conclusiones**

  En esta parte, se detallaran las conclusiones del proyecto.

[2] http://ipc.icaps-conference.org/

Capítulo 2

Estado del Arte

La planificación en Inteligencia Artificial se encarga de estudiar algoritmos y técnicas para la construcción de planes. Un plan es un conjunto de acciones que aplicadas en un estado inicial conduce a la obtención de un conjunto de metas deseadas. En la práctica, la planificación se sustenta en el uso de funciones de utilidad, también conocidas como heurísticas, que permiten evaluar la selección de acciones o estados de acuerdo a la utilidad que ofrecen al agente de planificación [GNT04].

2.1. Planificación monoagente

El problema de la planificación clásica se define como sigue [Wel99]:

'Dada una descripción del estado inicial del mundo (en algún lenguaje formal), una descripción de la meta del agente (el comportamiento deseable), y una descripción de las posibles acciones (atómicas) que pueden llevarse a cabo modelada a través de funciones de transformación de estados, el objetivo es obtener un plan. Es decir, un conjunto de acciones que transformen el estado inicial en un estado en el que las metas del agente se cumplan.'

La planificación clásica puede verse como un proceso de búsqueda en el que un único agente sintetiza un conjunto de acciones que le permite conseguir sus objetivos a partir de una situación inicial dada. Los planificadores clásicos se enfrentan a dos importantes escollos: la definición de lenguajes robustos y expresivos para modelar las acciones, y el desarrollo de técnicas eficientes para la resolución del problema.

A lo largo de los años, la investigación en planificación clásica se ha cen-

trado en el estudio de diferentes paradigmas de resolución de problemas. Estos paradigmas pueden clasificarse de acuerdo a los siguientes conceptos [Sap05]:

- **Representación y búsqueda basada en estados:** Los planificadores realizan una búsqueda basada en estados, donde un nodo del árbol de búsqueda representa una situación concreta del mundo.
- **Representación y búsqueda basada en planes:** Los planificadores construyen soluciones parciales como secuencias parcialmente ordenadas de acciones. En este caso, un nodo del árbol de búsqueda representa un plan de orden parcial construido con las acciones existentes.

Los primeros planificadores que se construyeron utilizaban el paradigma de representación y búsqueda basada en estados, de modo que las transiciones entre estados se producen por la aplicación de acciones. Posteriormente, la necesidad de manipular planes de orden parcial durante la búsqueda condujo al desarrollo de algoritmos de búsqueda en el espacio de planes [PW92, BW94, Wel94]. El paradigma de Planificación de Orden Parcial (POP) refina los planes parciales a través de la adición de acciones, enlaces causales y restricciones de orden. También existen enfoques adicionales que aplican otro tipo de refinamientos, como el paradigma Hierarchical Task Network (HTN) [EHN94], que reemplaza acciones abstractas por fragmentos de plan de bajo nivel.

La investigación en planificación ha introducido otros enfoques que incrementan significativamente la eficiencia de los sistemas de planificación. Entre estos modelos, destacan GRAPHPLAN [BF97], SATPLAN [KS96] y la planificación heurística [BG01]. En GRAPHPLAN, se utilizan grafos de planificación, mientras que en SATPLAN se convierte el problema de planificación en un problema de satisfacción de restricciones. Por último, la planificación heurística se desarrolla habitualmente en un espacio de estados. El objetivo de la planificación heurística es buscar funciones de utilidad o heurísticas que permitan guiar la búsqueda hacia el objetivo. Este enfoque ha resultado muy exitoso, como se ha demostrado con los planificadores FF [HN01] y LAMA [RW10].

2.1.1. Modelización de un problema de planificación

A continuación se verá un ejemplo de modelización de un problema de planificación monoagente. En un problema monoagente existe una única entidad de planificación, la cual tiene información completa sobre el entorno en

el que se desarrolla el problema así como del estado inicial del mismo.

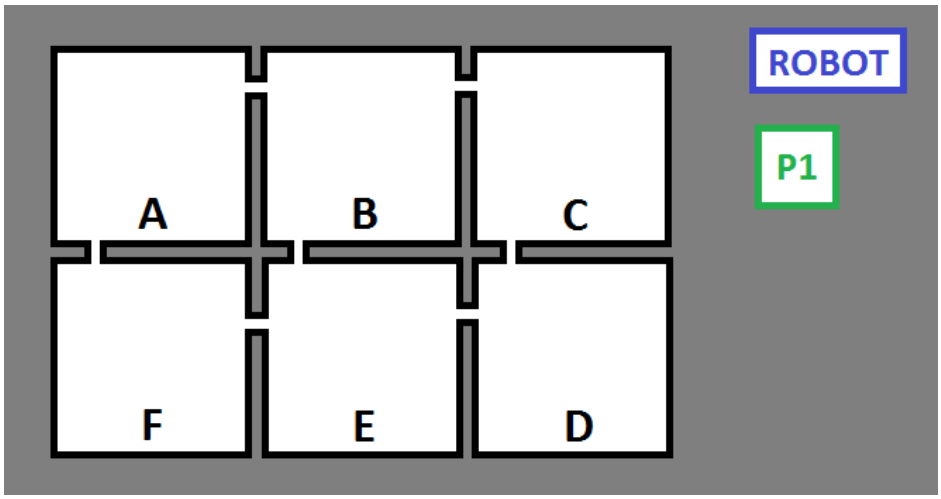

Figura 2.1: Problema1 del dominio Robot

La figura 2.1 muestra el *problema1* del dominio Robot. Este dominio consta de una serie de áreas delimitadas: A, B, C, D, E y F entre las cuales se puede transitar si están conectadas y robots que transportan paquetes de unas áreas a otras. Para este *problema1* existe un paquete y un robot. El robot es el único agente de este ejemplo, el cual puede realizar un conjunto de acciones tales como cargar un paquete, descargar un paquete o moverse de una área a otra.

Existen diversos lenguajes para la modelización de los problemas de planificación. El lenguaje más conocido es PDDL[GHK+98], el cual se utiliza en la mayoría de los planificadores independientes del domino. La versión más reciente de *PDDL* es *PDDL3.1*. El objetivo de esta versión es enriquecer el lenguaje con una representación de problemas al estilo de SAS+ [BN95]. Sin embargo, SAS+ permite usar variables de modo muy limitado al no permitir el anidamiento (sólo se permiten comparaciones y asignaciones con constantes). Por ello, *PDDL3.1* introduce variables objeto, que son variables asociadas a un dominio finito de valores, una solución flexible inspirada por el formalismo *Functional Strips* [Gef00]. Además, *PDDL3.1* permite especificar costes numéricos a las acciones del dominio, de modo que el coste de todas las acciones da una idea de la calidad del plan.

Una tarea de planificación se define mediante dos ficheros: uno que des-

cribe el dominio del problema y otro que describe el problema concreto de planificación a resolver, en el cual se definen los objetos que intervienen en el problema, el estado inicial, y el estado final.

2.1.1.1. Fichero PDDL del dominio

El fichero del dominio contiene la siguiente información:

- El tipo de los elementos del dominio.
- Los predicados del dominio.
- Las variables estado (funciones) del dominio.
- Las acciones del dominio.

La figura 2.2 muestra el código del dominio del problema Robot:

```
(define (domain Robot)
(:requirements :typing :equality :fluents)
(:types asset place - object
     area agent - place
     robot - agent
     box - asset)

(:predicates
     (connected ?l1 - area ?l2 - area))

(:functions
     (pos ?x - asset) - place
     (at ?x - robot)  - location)

(:action go-to
     :parameters (?r - robot ?from ?to - place)
     :precondition (and (= (at ?r) ?from)
                        (connected ?from ?to))
     :effect (and (assign (at ?r) ?to)))

(:action pickup
     :parameters (?r - robot ?bk - asset ?l - place)
     :precondition (and (= (pos ?bk) ?l) (= (at ?r) ?l))
     :effect (and (assign (pos ?bk) ?r)))

(:action drop
     :parameters (?r - robot ?bk - asset ?l - place)
     :precondition (and (= (pos ?bk) ?r) (= (at ?r) ?l))
     :effect (and (assign (pos ?bk) ?l))))
```

Figura 2.2: Fichero del dominio Robot

Para entender mejor el fichero, se explicará parte por parte su significado.

Todos los objetos que intervienen en un problema están asociados a una clase de objeto ó tipo. En este dominio, los tipos se definen de la siguiente manera:

- Los paquetes son del tipo `asset`.
- Los robots son del tipo `agent`.
- Las áreas y los elementos de tipo `agent` son del tipo `place`. Los robots (agentes) son considerados tipo `place` ya que un robot es uno de los lugares donde puede estar localizado un paquete (pueden cargar un paquete).

En la sección `:predicates` se pueden ver los predicados del dominio. Éstos expresan una relación entre tipos de objetos. En este problema se especifica el predicado `connected` para indicar las áreas concretas del problema que están conectadas entre sí.

En la sección `:functions` se describen las funciones del dominio. Una variable estado modela el estado concreto de una característica de un objeto de un problema. En PDDL las variables estado se representan mediante funciones. En este caso, se definen dos variables en el dominio del problema. Una de las variables es `(pos ?x - asset)` e indica el lugar donde se encuentra un paquete. La otra variable es `(at ?x - robot)`, la cual indica el área donde está el robot. Por ejemplo, si el paquete está en el área `A` y el robot está en el área `B`, los valores asociados a las variables serían los siguientes:

- `(= (pos paquete) A)`
- `(= (at robot) B)`

Las acciones en PDDL se definen siempre con los siguientes elementos: parámetros, precondiciones y efectos. Para explicar el significado de cada uno los elementos de una acción se tomará la acción `go-to` de la figura 2.2:

- **Parámetros:** Indican los tipos de objetos que intervienen en la acción. En la acción `go-to` se especifican un robot y dos localizaciones.
- **Precondiciones:** Una precondición es una condición que debe cumplirse en el estado actual para que la acción pueda ejecutarse. En el caso de que existan varias precondiciones deben cumplirse todas para poder ejecutar la acción. La acción `go-to` tiene dos precondiciones: una indica el área donde está el robot y la otra la conexión entre el área a donde quiera dirigirse el robot y el área donde se encuentra.
- **Efectos:** Como su nombre indica, son los efectos resultantes de aplicar la acción. El resultado la acción `go-to` es el cambio de la posición del robot, que se encontrará en una área distinta a la anterior.

Las restantes acciones que puede realizar el agente, definidas en el fichero del dominio 2.2, son las siguientes:

La acción `pickup` puede realizarse siempre que el paquete se encuentre en la misma área en la que se encuentra el robot. El resultado es el cambio de posición del paquete. De este modo, tras la ejecución de la acción el paquete se encontrará situado en un robot.

Por último, la acción `drop` puede realizarse cuando el robot lleva un paquete. Como resultado de esta acción, el robot desposita el paquete en el área donde éste se encuentra.

2.1.1.2. Fichero PDDL del problema

Además de la definición del dominio, el agente debe conocer el problema concreto de planificación a resolver, esto es, los objetos que intervienen en el problema, el estado inicial y el estado final que ha de conseguir. La figura 2.3 muestra el código PDDL del problema a resolver:

```
(define (problem problema1)
(:domain Robot)
(:objects
       robot - robot
       p1 - box
       A B C D E F - place)
```

```
(:init
        (connected A B) (connected B A)
        (connected B C) (connected C B)
        (connected C D) (connected D C)
        (connected D E) (connected E D)
        (connected E F) (connected F E)
        (connected F A) (connected A F)
        (= (pos p1) B)
        (= (at robot) A))

(:global-goal
        (and
                (= (pos p1) C)
                (= (at robot) D))))
```

Figura 2.3: Fichero del *problema1* a resolver

Como se puede ver en el fichero del problema 2.3 en la etiqueta `:objects`, los nombres que identifican los elementos del problema son los siguientes:

- Áreas: `A, B, C, D, E`
- Robot: `robot`
- Paquete: `p1`

El estado inicial del problema, sección `:init` de la figura 2.3, contiene instanciaciones de predicados, denominados literales o proposiciones, y los valores asignados a las variables estado. En el estado inicial de este problema se indican las conexiones entre áreas y las localizaciones iniciales del paquete y el robot.

Por último, el estado final, `:global-goal`, contiene las metas que han de cumplirse para finalizar el problema. En este caso, el paquete debe estar en el área `C` y el robot encontrarse en el área `D` en el estado final.

2.2. Planificación multiagente

La planificación multiagente cooperativa extiende el problema clásico de planificación monoagente a un entorno multiagente, donde cada agente es una entidad con capacidades de planificación (planificador). Una tarea de planificación multiagente consiste en realizar conjuntamente un plan solución entre

diferentes entidades que alcance todas las metas de un problema. De esta manera, la planificación multiagente es el problema de coordinar agentes en un entorno compartido donde la información está distribuida, enfatizando así dos aspectos claves que no están presentes en la planificación monoagente: la coordinación y la distribución de la información entre todos los agentes.

En líneas generales, resolver una tarea de planificación multiagente implica seguir los siguientes pasos:

1. Refinamiento de las metas
2. Distribución de tareas
3. Coordinación antes de planificar
4. Planificación individual
5. Coordinación después de planificar
6. Ejecución del plan

Algunos de los puntos anteriores pueden ser eliminados o fusionados según el tipo de problema o las metas que se quieren conseguir. Por ejemplo, en aquellos problemas donde no se distribuyen las metas explícitamente no se realizaría el punto 2.

2.2.1. Modelización de un problema multiagente

Para mostrar la modelización de un problema multiagente, se analizará una extensión del problema visto anteriormente del dominio Robot. En el nuevo problema, en lugar de tener un solo agente, el problema dispondrá de dos robots y un total de tres paquetes. Se remarcarán los cambios de la extensión multiagente en el código PDDL correspondiente.

Al igual que en el caso monoagente, un problema de planificación multiagente se modeliza mediante la definición de dos tipos de conocimientos: el dominio y el problema en particular a resolver. Sin embargo, al tratarse de un problema multiagente habrá un problema por cada agente. Estos problemas contienen un estado inicial y unas metas a conseguir. La razón principal de definir un problema distinto para cada agente es poder diferenciar a cada agente de los demás mediante el estado inicial.

En el fichero de problema se especifica el estado inicial `:init` que conoce cada agente, el cual dependerá de la distribución de la información del problema. Supongamos una tarea de planificación en la que se tienen dos agentes y dos aviones, el *avión1* y el *avión2*, los cuales pueden transportar agua o comida. En el estado inicial de cada agente se especifica que avión poseen y que elemento transportan.

Respecto al dominio, si hubiesen agentes con capacidades de planificación diferentes habría un fichero del dominio por cada capacidad de planificación. En este caso, los agentes del dominio Robot tienen las mismas capacidades de planificación, por lo que solo habrá un dominio.

En lo referente a las metas, las cuales se especifican en el apartado `:global-goal`, serán iguales para todos los agentes ya que todos ellos cooperarán entre sí para resolver el conjunto de metas globales.

De este modo, un problema de planificación multiagente es aquel en el que dos o más agentes parten de unos datos iniciales y cooperan entre sí para lograr unas metas comunes. A diferencia de la planificación monoagente, un agente no debe encargarse de conseguir todas las metas, sino que todos los agentes han de coordinarse entre sí para resolverlas.

2.2.1.1. Fichero PDDL del dominio multiagente Robot

La figura 2.4 muestra el código PDDL del dominio multiagente Robot:

```
(define (domain Robot)
(:requirements :typing :equality :fluents)
(:types asset place - object
        robot - agent
        area agent - place
        box - asset)

(:predicates
        (connected ?l1 - area ?l2 - area)
        (my-agent ?r - robot))

(:functions
        (pos ?x - asset) - place
        (at ?x - robot)  - location)

(:action go-to
```

```
        :parameters (?r - robot ?from ?to - location)
        :precondition (and (= (at ?r) ?from)
                           (connected ?from ?to)(my-agent ?r))
        :effect (and (assign (at ?r) ?to)))

(:action pickup
        :parameters (?r - robot ?bk - asset ?l - location)
        :precondition (and (= (pos ?bk) ?l) (= (at ?r) ?l)
                           (my-agent ?r))
        :effect (and (assign (pos ?bk) ?r)))

(:action drop
        :parameters (?r - robot ?bk - asset ?l - location)
        :precondition (and (= (pos ?bk) ?r) (= (at ?r) ?l)
                           (my-agent ?r))
        :effect (and (assign (pos ?bk) ?l)))
```

Figura 2.4: Fichero del dominio multiagente Robot

Como se puede observar en la figura 2.4, la sección `:predicates` contiene un predicado adicional. Este predicado, **`(my-agent ?r - robot)`**, es necesario para diferenciar el robot asociado a cada agente. Además, todas las acciones utilizan dicho predicado para especificar el robot que realiza la acción.

2.2.1.2. Fichero PDDL del problema del dominio multiagente Robot

Como se ha comentado anteriormente, por cada agente existe un problema. Este problema contiene un estado inicial y las metas a conseguir, que son comunes a todos los agentes. En este problema multiagente hay dos agentes robot, el `robot1` y el `robot2`. Solo se mostrará el problema del `robot1` y después se comentará que diferencias existentes con respecto al problema del `robot2`.

```
(define (problem problema1)
(:domain Robot)
(:objects
     robot1 robot2 - robot
     package1 package2 package3- box
     A B C D E F - location)

(:shared-data
  ((pos ?x - asset) - place)
  (at ?x - robot)  - location) - robot2)
```

```
(:init
     (my-agent robot1)
     (connected A B) (connected B A)
     (connected B C) (connected C B)
     (connected C D) (connected D C)
     (connected D E) (connected E D)
     (connected E F) (connected F E)
     (connected F A) (connected A F)
     (= (pos package1) B)
     (= (pos package2) B)
     (= (pos package3) B)
     (= (at robot1) A)
     (= (at robot2) B))

(:global-goal
     (and
          (= (pos package1) C)
          (= (pos package2) C)
          (= (pos package3) D)
          (= (at robot1) D)
          (= (at robot2) F))))
```

Figura 2.5: Fichero del problema del robot1

En la sección `:init` de la figura 2.5 se encuentra el estado inicial del problema para el `robot1`. En él se indican las áreas que están conectadas entre sí, la posición de los tres paquetes del problema (todos ubicados en el área `B`) y las posiciones del `robot1` y `robot2`, área `A` y área `B` respectivamente.

La sección `:shared-data` muestra la información que el `robot1` comparte con el `robot2`. En este caso, tal y como se puede ver en la figura 2.5, el `robot1` comparte con el `robot2` la posición de los paquetes del problema y la posición de los robots.

En este caso, lo que cambia en el fichero del problema del `robot2` respecto al de la figura 2.5 es el predicado que especifica el robot asociado al agente. En la especificación del estado inicial del `robot2` en vez del literal `(my-agent robot1)` estará el literal `(my-agent robot2)`, y en la sección `:shared-data` ahora mostrará la misma información pero compartible con el `robot1`.

2.2.2. Privacidad

Uno de los mayores problemas en la planificación multiagente es el manejo de la información privada. Al tratarse de información distribuida, puede

darse el caso que ninguna entidad tenga una visión total de todos los datos del problema. La privacidad se refiere a datos que no son visibles por uno o varios agentes y que al menos son visibles por un agente diferente a los anteriores.

Hay varios tipos de privacidad: nula, parcial o total. Para entender los tres tipos de privacidad, se mostrará un ejemplo con el dominio Robot visto anteriormente. Para ello, utilizaremos el valor de las variables estado `(pos package1)` y `(pos package2)`. Supongamos que el robot `robot1` no tiene visión sobre las áreas `E y F` y que el robot `robot2` tiene visión nula del paquete `package1` por una especificación nueva del problema . Por lo tanto:

- **Privacidad nula**: El dato es visible por los dos agentes. Por ejemplo, los dos robots pueden saber en cada momento donde está el paquete `package2`.
- **Privacidad parcial**: El agente no conoce el valor de una determinada variable en todo momento. En este ejemplo, el `robot1` a veces no sabe donde está el paquete `package1`. Esto es así porque puede que el paquete `package1` esté en áreas donde el robot `robot1` no tiene visibilidad, como las áreas `E y F`.
- **Privacidad total**: El agente no conoce el valor de una determinada variable en ningún momento. En este caso, el `robot2` no conocerá nunca la posición del paquete `package1`, ya que tiene visión nula de él.

Existe un conjunto de literales y valores de las variables estado que son visibles a todos los agentes, los cuales forman una excepción. Este conjunto son los literales y los valores de las variables del estado inicial y final, que son visibles para todos los agentes.

En el caso de la privacidad parcial o total, cuando un agente requiere el valor de una variable que no puede saber por sí mismo se establece un mecanismo de comunicación con los demás agentes en el que si es posible compartir el valor de esa variable, se comparte con él.

2.3. Heurísticas en planificación

Las heurísticas de planificación son una parte importante del proceso de resolución, ya que son las funciones que permiten orientar la búsqueda del

planificador hacia los objetivos. Existen múltiples trabajos e investigaciones en planificación heurística, habiéndose desarrollado numerosas y diferentes heurísticas con diferentes propiedades (monotonicidad, optimalidad, etc.).

En este trabajo nos centramos en las heurísticas basadas en *landmarks*. Un *landmark* es una variable con un valor determinado que ha de cumplirse en algún momento en todo plan solución de un problema. Existen diferentes tipos de landmarks pero los que se utilizan habitualmente para el diseño de heurísticas son landmarks que se calculan a partir de estados de planificación. Por ese motivo, las heurísticas basadas en landmarks se han utilizado frecuentemente en planificadores basados en estados. No obstante, también pueden utilizarse en otro tipo de planificadores como, por ejemplo, *planificadores de orden parcial* donde se calculan estados únicamente para la aplicación de heurísticas.

2.3.1. Landmarks

Unos de los trabajos más relevantes sobre el cálculo y extracción de *landmarks* son la tesis doctoral [SO03] de la investigadora del DSIC Laura Sebatiá Tarín, y el trabajo *Ordered Landmarks in Planning* publicado en la revista JAIR[1] donde ella misma participó [HPS04].

Un *landmark* es un hecho que ha de ser cierto en algún punto en todo plan solución del problema. En los trabajos citados, se muestra un algoritmo para el cálculo y la extracción de *landmarks*. Dicho algoritmo extrae todos los *landmarks* de un problema mediante una serie de validaciones y establece un orden entre los *landmarks* extraidos. Más adelante se explicará con profundidad este algoritmo en la sección 4.2 del capítulo 4.

Hay que destacar que si no aparece algún *landmark* en un plan éste nunca sería un plan solución, ya que sin dicho *landmark* no se podrían alcanzar todas las metas del problema. Además por definición, todos los literales meta e iniciales son landmarks.

A continuación se presenta un ejemplo para entender mejor el concepto de *landmark*:

[1] http://www.aaai.org/Press/Journals/jairvol22.php

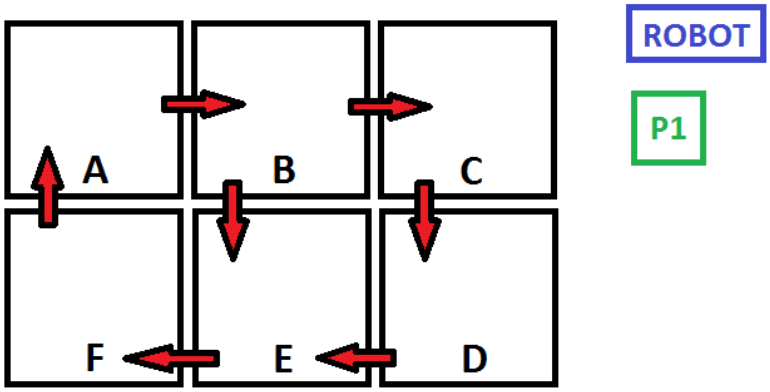

Figura 2.6: Problema2 del dominio Robot

El dominio de este *problema2* es el dominio Robot visto anteriormente. Consta de una serie de áreas delimitadas: `A, B, C, D, E` y `F` en las cuales se puede transitar de una a otra siguiendo las flechas de la figura 2.6. Además, el *problema2* consta de un paquete, `P1`, que puede estar en alguna de las áreas nombradas anteriormente o portado por un robot. El robot será el único agente de este ejemplo, el cual puede realizar el conjunto de acciones vistas anteriormente:

- **go-to:** Ir de un área a otra.
- **pickup:** Recoger un paquete de un área.
- **drop:** Dejar un paquete en un área.

El estado inicial del *problema2* es:

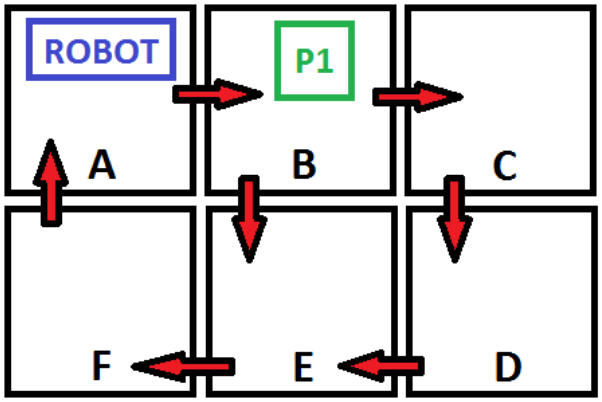

Figura 2.7: Estado inicial del *problema2* del dominio Robot

y el estado final que se quiere alcanzar se muestra en la figura 2.8:

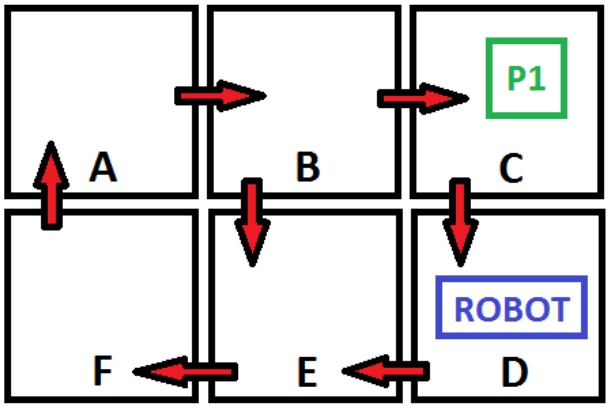

Figura 2.8: Estado final del *problema2* del dominio Robot

Uno de los posibles planes, constaría de las siguientes acciones:

- Robot: `go-to` desde el área `A` al área `B`.
- Robot: `go-to` desde el área `B` al área `E`.
- Robot: `go-to` desde el área `E` al área `F`.
- Robot: `go-to` desde el área `F` al área `A`.
- Robot: `go-to` desde el área `A` al área `B`.
- Robot: `pick-up P1` del área `B`.
- Robot: `go-to` desde el área `B` al área `C`.
- Robot: `drop P1` al área `C`.
- Robot: `go-to` desde el área `C` al área `D`.

Otro posible plan constaría de las siguientes acciones:

- Robot: `go-to` desde el área `A` al área `B`.
- Robot: `pick-up P1` del área `B`.

- Robot: `go-to` desde el área `B` al área `C`.
- Robot: `drop P1` al área `C`.
- Robot: `go-to` desde el área `C` al área `D`.

Fijándose en los planes anteriores, se puede observar que si se eliminan las acciones

- `go-to` desde el área `E` al área `F`
- `go-to` desde el área `E` al área `F`.

no afecta para que el plan siga siendo un plan solución. Esto es porque los efectos de estas acciones, `(= (at robot) E)` y `(= (at robot) F)` no son landmarks. Conseguir o no esos efectos no influye en que el plan sea un plan solución del problema.

En este problema, se puede identificar un conjunto de landmarks del estado inicial y de las metas:

- `(= (pos p1) B)`
- `(= (at robot) A)`
- `(= (pos p1) C)`
- `(= (at robot) D)`

Y otro conjunto de landmarks que no están ni en el estado inicial ni en las metas. Sin estos landmarks, no se conseguirían las metas:

- `(= (at robot) B)`
- `(= (at robot) C)`
- `(= (pos p1) robot)`

Así pues, pueden existir más landmarks que los iniciales y finales, siendo estos tan importantes como los anteriores. Si alguno de estos landmarks no está en un plan, éste nunca será un plan solución.

2.3.2. Heurística basada en landmarks usada por el planificador LAMA

LAMA[RW10] es un planificador clásico de búsqueda hacia delante basado en una pseudo-heurística derivada de los landmarks. En vez de utilizar una sola heurística, LAMA usa una búsqueda multi-heurística combinando la heurística de landmarks con una variante de la heurística FF[HN01]. Para combinar estas heurísticas, LAMA utiliza la heurística de FF en primera instancia y cuando la aplicación de la heurística FF devuelve el mismo valor heurístico para varios planes, utiliza la heurística de landmarks para desempatar y elegir una entre ellas. Con la estimación dada por los landmarks, LAMA se decanta por los estados que han conseguidos más landmarks que los demás.

La razón por la que se decanta por los estados con más landmarks es porque esos estados están más cercanos a ser plan solución que los demás. Por ejemplo, si hubiese que desempatar entre dos planes, *p1* y *p2*, que han alcanzado cinco y siete landmarks respectivamente se elegiría el plan *p2*, ya que es el que más cerca está de conseguir todos los landmarks de un problema y por lo tanto, de ser un plan solución del problema a considerar.

Capítulo 3

FMAP y PlanInteraction

Este trabajo está implementado en el planificador FMAP, del cual recoge los elementos básicos para la realización de la heurística FMAP-land, e integrado la plataforma PlanInteraction, pieza clave para coordinar a los agentes con los muchos pasos de mensajes que se realizan.

3.1. FMAP

FMAP es un sistema de planificación multiagente cooperativo que permite a un conjunto de entidades de planificación trazar, de forma coordinada, un curso de acción o plan para alcanzar un conjunto de objetivos o metas a partir de una situación inicial dada.

FMAP está basado en técnicas de planificación de orden parcial (POP) [PW92]. A diferencia del paradigma clásico POP, que plantea la construcción de los planes de forma regresiva a partir de las metas, FMAP realiza una búsqueda progresiva, de modo similar a los planificadores basados en estados, manteniendo al mismo tiempo la filosofía de menor compromiso que caracteriza a los planificadores de orden parcial.

El algoritmo de FMAP puede interpretarse como una búsqueda en el espacio de planes. Los agentes participantes construyen, de forma cooperativa, un árbol de búsqueda donde cada nodo es un plan de orden parcial cuyas acciones han sido aportadas por distintos agentes. El nodo raíz del árbol es un plan vacío, y cada nodo del árbol refina a su nodo padre introduciendo una nueva acción sobre el mismo.

FMAP aplica una estrategia de búsqueda heurística para explorar el árbol de planes multiagente. Cada plan Π del árbol se evalúa mediante una

función de utilidad $f(\Pi) = g(\Pi) + h(\Pi)$, donde $g(\Pi)$ mide el coste invertido en construir el plan Π desde el plan vacío inicial, mientras que $h(\Pi)$ estima el coste de alcanzar un plan solución desde Π. FMAP realiza estas estimaciones mediante una novedosa función heurística basada en Grafos de Transición de Dominio [cita].

Al basarse en una estrategia de búsqueda heurística, el rendimiento de FMAP depende en gran medida de la calidad de las estimaciones obtenidas por la función heurística. Aunque la heurística utilizada por FMAP ofrece un rendimiento notable en dominios de la Competición Internacional de Planificación, es posible aplicar técnicas como el cálculo de landmarks para mejorar la precisión de sus estimaciones.

FMAP ya extrae y utiliza landmarks en un contexto monoagente, lo que ha mejorado el rendimiento del planificador. Sin embargo, hasta la fecha no se ha desarrollado ningún algoritmo multiagente de extracción de landmarks. Por tanto, uno de los objetivos del presente proyecto se centra en calcular landmarks en tareas multiagente y utilizarlos para mejorar el rendimiento de FMAP.

3.2. PlanInteraction

PlanInteraction [1] es un trabajo coordinado entre las universidades UPV [2], UC3M [3] y [4]. El interés del proyecto PlanInteraction es desarollar nuevas técnicas y tecnologías basadas en dinámicas sociales para el diseño e implementación de una plataforma de planificación multiagente compuesta de entidades de planificación autónomas.

El diseño de PlanInteraction, desde la visión más general, cuenta con un entorno en el que coexisten un conjunto de agentes con capacidades de planificación y/o ejecución, una serie de elementos de control, un simulador y una interfaz de interacción con el mundo real. Esta arquitectura permite crear múltiples configuraciones según el tipo de problema que se vaya a tratar. En concreto, la aproxiación general que se ha adoptado para la resolución de los problmeas de planificación supone la división de las tareas de planificación y ejecución, teniendo así un conjunto de agentes de planificación y un conjunto

[1] http://servergrps.dsic.upv.es/planinteraction/
[2] Universidad Politécnica de Valencia
[3] Universidad Carlos III de Madrid
[4] Universidad de Granada

de agentes de ejecución.

La especificación y control de las comunicaciones entre agentes viene heredado de los mecanismos proporcionados por Magentix2 [5]. El broker de comunicaciones que se utiliza es Apache Qpid y los mensajes intercambiados entre los agentes siguen el estándard FIPA-ACL [6]. Este broker de comunicaciones establece una dirección única para cada agente a partir de su identificador de agente, por lo que cada agente debe tener un identificador único. Los mensajes recibidos por el broker son redirigidos a la cola de mensajes del receptor correspondiente, donde cada agente procesará los mensajes de su cola de forma secuencial y asíncrona.

Los protocolos de comunicación/ejecución que van a seguir los agentes se definen como grafos de estados. Magentix2 proporciona toda una serie de estados con funcionalidades predefinidas y las herramientas para su definición. Una de las apartaciones de PlanInteraction ha sido simplificar el proceso de definición de estos estados y transiciones mediante unas APIs de más alto nivel. Los tipos de estados permitidos son los que siguen:

- BEGIN: Estado inicial.
- FINAL: Estado final.
- ACTION: Estado de propósito general.
- WAIT: Estado de espera de recepción de mensajes.
- RECEIVE: Estado de recepción y manejo de mensajes.
- SEND: Estado de construcción y envío de un mensaje.
- NAM[7]: Estado de absorción de mensajes.

Para definir correctamente el flujo de ejecución de un protocolo de comunicación se han de establecer las transiciones entre estos estados, que pueden ser multiples de entrada o de salida.

El objetivo de integrar FMAP-land en esta plataforma es aprovechar su capacidad de sintetizar el paso de mensajes y la coordinación de los agentes

[5]http://www.gti-ia.upv.es/sma/tools/magentix2/
[6]http://www.fipa.org/repository/aclspecs.html
[7]Not Accepted Message

para facilitar la implementación de comportamiento que extrae todos los landmarks de un problema dado. Este comportamiento seguirá el algoritmo explicado en el capítulo 4.

Capítulo 4

Diseño de FMAP-Land

El diseño de FMAP-Land se basa en el algoritmo presentado en la Tesis Doctoral de Laura Sebastián Tarín [SO03]. En dicho trabajo se define el mecanismo para identificar los landmarks de un problema de planificación y establecer órdenes necesarios entre ellos, donde un orden necesario indica que un landmark debe aparecer necesariamente antes que otro en cualquier plan solución. El presente trabajo se centra en el cálculo de landmarks simples, excluyendo el cálculo de landmarks disyuntivos, que será desarrollado como trabajo futuro.

El presente capítulo se estructura como sigue:

- La sección 4.1 introduce y formaliza los conceptos fundamentales que van a utilizarse a lo largo del capítulo (tarea de planificación, acción, literal, landmark, etc).
- La sección 4.2 resume el algoritmo monoagente de extracción de landmarks [SO03].
- La sección 4.3 expone los cambios realizados para adaptar el algoritmo de extracción de landmarks al contexto multiagente.
- Finalmente, la sección 4.4 detalla la aplicación del árbol de landmarks obtenido para mejorar el rendimiento del planificador FMAP-land.

4.1. Formalización de conceptos

Definición 1. *(**Tarea de planificación multiagente**) Una **tarea de planificación multiagente** es una tupla* $\mathcal{T} = \langle \mathcal{AG}, \mathcal{O}, \mathcal{V}, \mathcal{A}, \mathcal{I}, \mathcal{G}, \rangle$. $\mathcal{AG} = \{1, \ldots, n\}$ *es un conjunto finito no vacío de agentes de planificación.* $\mathcal{O}$ *es*

un conjunto finito de objetos, que modelan los elementos del dominio de planificación sobre los que actúan las acciones de planificación. $\mathcal{V}$ es un conjunto finito de variables de estado que modelan los estados del mundo. Cada variable de estado $v \in \mathcal{V}$ está asociada a un dominio finito de valores mutuamente exclusivos $\mathcal{D}_v$. Cada valor en el dominio de una variable corresponde a un objeto del dominio de planificación, esto es, $\forall v \in \mathcal{V}$, $\mathcal{D}_v \subseteq \mathcal{O}$. Cuando un valor es asignado a una variable de estado, el par variable-valor actúa como un átomo instanciado en planificación proposicional. $\mathcal{A}$ es el conjunto de acciones deterministas de los agentes. $\mathcal{I}$ es el conjunto de valores asignado a las variables de estado en $\mathcal{V}$ y representa el estado inicial de la tarea de planificación multiagente $\mathcal{T}$. $\mathcal{G}$ es el conjunto de metas de la tarea de planificación multiagente que los agentes deben satisfacer; $\mathcal{G}$ representa los valores que las variables de estado deben adquirir en el estado final.

La información sobre los estados del mundo que poseen los agentes se modela mediante un conjunto de variables instanciadas. Esto incluye el estado inicial, $\mathcal{I}$, y las metas, $\mathcal{G}$. A diferencia de los modelos basados en *STRIPS* [FN71], que aplican negación por fallo, el modelo que se utiliza permite la representación explícita de la información verdadera y falsa. Por tanto, este modelo adopta la asunción de mundo abierto, considerando que la información que no está explícitamente almacenada en el modelo interno de los agentes es desconocida para ellos.

Definición 2. *(**Variable instanciada o literal**) Una **variable instanciada** o **literal** del problema es una tupla de la forma $\langle v, d \rangle$, donde $v \in \mathcal{V}$, $d \in \mathcal{D}_v$. Una variable instanciada negativa toma la forma $\langle v, \neg d \rangle$. Una variable instanciada positiva $\langle v, d \rangle$ indica que la variable v toma el valor d, mientras que una variable instanciada negativa $\langle v, \neg d \rangle$ indica que la variable v no toma el valor d. Todo literal l tiene una lista asociada $a_l \subseteq \mathcal{AG}$ que indica los agentes capaces de conseguir el literal l.*

Los agentes en este modelo son heterogéneos, dado que pueden tener diferentes conocimientos y habilidades de planificación. Además, pueden tener información incompleta acerca de la tarea de planificación multiagente, dado que ésta está distribuida entre los agentes. En dicho caso, los agentes deben cooperar para resolver la tarea de planificación multiagente. Aunque la información esté distribuida entre los agentes, debe haber un subconjunto de variables de estado susceptible de ser compartido entre los agentes, de modo que éstos puedan interactuar adecuadamente. Para denotar las acciones, metas, etc, de un agente $i \in \mathcal{AG}$ se usará la notación de superíndice x^i para cada aspecto x.

Del conjunto de variables $\mathcal{V}$ de la tarea de planificación multiagente, $\mathcal{V}^i$ es el conjunto de variables gestionadas por el agente i, lo que incluye las variables privadas que sólo i conoce, y las variables públicas compartidas con otros agentes. Por tanto, $\mathcal{V} = \{\mathcal{V}^i\}_{i=1}^n$. $D_v^i \subseteq D_v$ es el conjunto de valores de una variable $v \in \mathcal{V}^i$ que son visibles para el agente i. La información del estado inicial de la tarea de planificación multiagente, $\mathcal{I}$, se modela mediante un conjunto de variables instanciadas positivas y negativas. Esta información está distribuida entre los agentes bajo la asunción de que el conocimiento parcial de los agentes sobre $\mathcal{I}$ es consistente, es decir, no hay información contradictoria entre los agentes. Por tanto, $\mathcal{I}$ puede definirse como $\mathcal{I} = \bigcup_{\forall i \in \mathcal{AG}} \mathcal{I}^i$. Es posible definir tareas de planificación multiagente en las que todos los agentes tienen una visión completa del estado inicial $\mathcal{I}$, es decir, $\forall i \in \mathcal{AG}$, $\mathcal{I}^i = \mathcal{I}$.

Cada agente $i \in \mathcal{AG}$ tiene un conjunto asociado de acciones $\mathcal{A}^i$, de modo que el conjunto de acciones de una tarea de planificación multiagente se define como $\mathcal{A} = \bigcup_{\forall i \in \mathcal{AG}} \mathcal{A}^i$. Una acción α es pública si dos o más agentes la comparten, esto es, $\alpha \in A^i \wedge \alpha \in A^j$, $i \neq j$. $\alpha \in \mathcal{A}^i$ es privada para un agente i si y sólo si $\alpha \notin \mathcal{A}^j, \forall j \neq i$. Una acción $\alpha \in \mathcal{A}^i$ denota que el agente i posee la capacidad expresada en α. Si α forma parte de un plan solución, el agente i es también responsable de ejecutar α.

Definición 3. *(**Acción**) Una **acción** $\alpha \in \mathcal{A}$ es una tupla $\langle PRE(\alpha), EFF(\alpha)\rangle$. $PRE(\alpha) = \{p_1, \ldots, p_n\}$ es un conjunto de variables instanciadas que representan las precondiciones de α, mientras que $EFF(\alpha) = \{e_1, \ldots, e_m\}$ es un conjunto de operaciones de la forma $(v = d)$ o $(v \neq d)$, $v \in \mathcal{V}$, $d \in \mathcal{D}_v$, que representan las consecuencias de ejecutar α.*

Una acción α puede pertenecer a diferentes agentes, es decir, $\alpha \in \mathcal{A}^i$ y $\alpha \in \mathcal{A}^j$, $i \neq j$. El resultado de ejecutar α en S es un nuevo estado del mundo S' que surge de la revisión de S por $EFF(\alpha)$, es decir, S' se genera actualizando las variables instanciadas en S de acuerdo a los efectos de α:

- Una operación $(v = d) \in EFF(\alpha)$ implica la adición de una variable instanciada $\langle v, d\rangle$ y un conjunto de variables instanciadas $\langle v, \neg d'\rangle, \forall d' \in \mathcal{D}_v \mid d' \neq d$ al estado S'. Si $\langle v, d'\rangle \in S$ o bien $\langle v, \neg d\rangle \in S$, $d' \neq d$, la operación $(v = d)$ implica también el borrado de las variables instanciadas $\langle v, \neg d\rangle$ y $\langle v, d'\rangle$ de S'.

- Una operación $(v \neq d) \in EFF(\alpha)$ implica la adición de una variable instanciada $\langle v, \neg d\rangle$ al estado S'. Si $\langle v, d\rangle \in S$, la operación $(v \neq d)$ implica también el borrado de la variable instanciada $\langle v, d\rangle$ en S'. Nótese que la sola existencia de una variable instanciada $\langle v, \neg d\rangle$ en un estado S indica que el valor de la variable v es desconocido en S.

El conjunto de precondiciones de una acción α, $PRE(\alpha)$, indica qué variables instanciadas deben figurar en un estado S para que α sea aplicable en ese estado. Una precondición positiva de la forma $\langle v, d\rangle$ indica que la variable instanciada $\langle v, d\rangle$ debe aparecer en S, mientras que una precondición negativa $\langle v, \neg d\rangle$ indica que la variable instanciada $\langle v, \neg d\rangle$ debe figurar en S. Nótese que la existencia de una variable instanciada positiva $\langle v, d\rangle$ implica también la existencia de una variable instanciada negativa $\langle v, \neg d'\rangle$ para el resto de valores en el dominio de la variable, es decir, $(\exists\langle v, d\rangle \in S) \Rightarrow (\forall d' \in \mathcal{D}_v,\ d' \neq d,\ \exists\langle v, \neg d'\rangle \in S)$.

Definición 4. *(**Plan de orden parcial**) Un **plan de orden parcial o plan parcial** es una tupla $\Pi = \langle \Delta, \mathcal{OR}, \mathcal{CL}\rangle$. $\Delta \subseteq \mathcal{A}$ es el conjunto de acciones en Π. $\mathcal{OR}$ es un conjunto de restricciones de orden ($\prec$) en Δ. $\mathcal{CL}$ es un conjunto de enlaces causales sobre Δ. Un enlace causal toma la forma $\alpha \overset{\langle v,d\rangle}{\rightarrow} \beta$ o bien $\alpha \overset{\langle v,\neg d\rangle}{\rightarrow} \beta$, donde $\alpha \in \mathcal{A}$ y $\beta \in \mathcal{A}$ son acciones en Δ. $\alpha \overset{\langle v,d\rangle}{\rightarrow} \beta$ indica que hay una operación $(v = d)$ tal que $v \in \mathcal{V}$, $d \in \mathcal{D}_v$, $(v = d) \in EFF(\alpha)$ y una variable instanciada $\langle v, d\rangle \in PRE(\beta)$. $\alpha \overset{\langle v,\neg d\rangle}{\rightarrow} \beta$ indica que hay una variable instanciada $\langle v, \neg d\rangle$ tal que $v \in \mathcal{V}$, $d \in \mathcal{D}_v$, $\langle v, \neg d\rangle \in PRE(\beta)$ soportada por una operación $(v \neq d) \in EFF(\alpha)$ o una operación $(v = d') \in EFF(\alpha)$, $d' \in \mathcal{D}_v$, $d' \neq d$.*

Esta definición de plan parcial muestra que un plan puede verse como un grafo dirigido acíclico, donde Δ representa los nodos del grafo (acciones) y $\mathcal{OR}$ y $\mathcal{CL}$ son conjuntos de aristas dirigidas que representan las precedencias y enlaces causales entre acciones, respectivamente.

Un plan parcial *vacío* se define como $\Pi_0 = \langle \Delta_0, \mathcal{OR}_0, \mathcal{CL}_0\rangle$, donde Δ_0 contiene α_0 y α_f, las acciones inicial y final del plan, respectivamente. α_0 y α_f son acciones ficticias que no pertenecen al conjunto de acciones de ningún agente. $\mathcal{OR}_0$ contiene la restricción de orden $\alpha_0 \prec \alpha_f$ y $\mathcal{CL}_0$ es un conjunto vacío. De este modo, un plan Π para una tarea de planificación multiagente $\mathcal{T}$ contendrá siempre las dos acciones ficticias, de forma que $PRE(\alpha_0) = \emptyset$, $EFF(\alpha_0) = \mathcal{I}$, $PRE(\alpha_f) = \mathcal{G}$, y $EFF(\alpha_f) = \emptyset$; es decir, α_0 representa la situación inicial de la tarea de planificación multiagente $\mathcal{T}$, y α_f representa las metas globales de $\mathcal{T}$.

Asumiendo que $\mathcal{G} \neq \emptyset$, un plan vacío es incompleto si las precondiciones de α_f no están soportadas aún por un enlace causal.

Definición 5. *(**Plan solución**) Un plan Π es un **plan solución** para una tarea de planificación multiagente $\mathcal{T}$ si soporta todas las precondiciones de la acción final α_f, es decir, si resuelve todas las metas de la tarea (Π es un plan completo).*

Nótese que requerimos que Π sea un plan completo, de modo que no puede tener metas pendientes. En consecuencia, las precondiciones de la acción ficticia final α_f estarán también resueltas, lo que garantiza que Π resuelve la tarea de planificación multiagente $\mathcal{T}$.

Definición 6. *(**Landmark**) Un literal **l** es un **landmark** si y sólo si **l** es cierto en algún punto en todos los planes solución del problema a resolver.*

Por definición, todas las metas del problema son **landmarks**, ya que deben satisfacerse en todo plan solución. Además, se consideran también **landmarks** todos los literales del estado inicial $\mathcal{I}$.

Definición 7. *(**Orden necesario**) Sean dos literales **l** y **l'**. Un **orden necesario** entre **l** y **l'**, denotado como **l** $\leq_n$ **l'**, implica que siempre que se consiga **l'** en el plan, **l** deberá ser cierto en el estado inmediatamente anterior.*

Definición 8. *(**Grafo de landmarks**) Se define **grafo de landmarks** como $LG = (N, E)$, donde N es el conjunto de landmarks de $\mathcal{T}$, y E es el conjunto de órdenes necesarios establecidos entre pares de landmarks.*

4.2. Algoritmo extracción landmarks

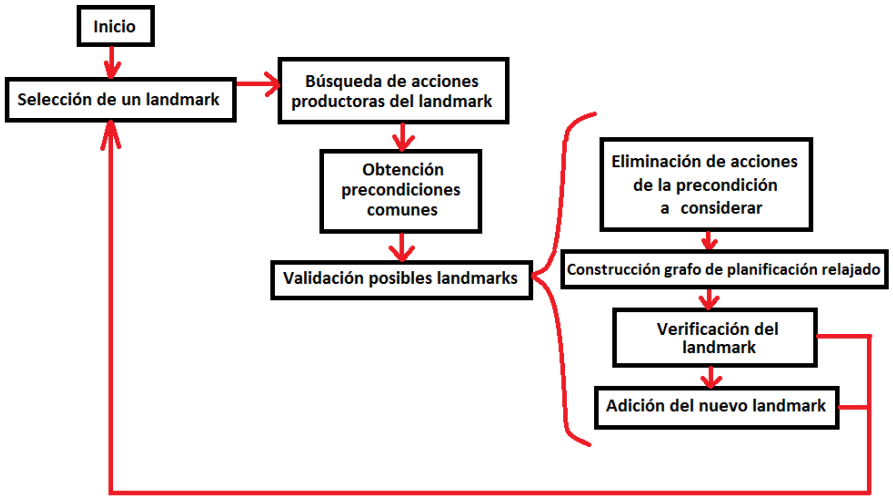

Figura 4.1: Algoritmo de extracción de landmarks

El algoritmo de extracción de landmarks original [SO03] en el que se basa la implementación multiagente desarrollada se muestra en la Figura 4.1. La estructura que se usará en este algoritmo es la siguiente:

Estructura de landmarks por niveles: Se define una *estructura de landmarks por niveles (LL)* como una tabla donde se inserta cada landmark según el nivel mínimo **n** en el que es encontrado por un agente en su grafo del planificación relajado [ZNK07]. El nivel mínimo **n** de un landmark l se define de la siguiente manera:

$$n = \text{mín}^a(nivel(l)),\ a \in \mathcal{AG},\ l \in landmarks(\mathcal{T})$$

El algoritmo de extracción de landmarks consta de las siguientes fases:

- **Inicio**: Se calcula un grafo de planificación relajado [BF97] con todas las acciones $\mathcal{A}$ del problema. Ya que por definición todos los literales del estado inicial y las metas son landmarks, se añaden directamente al grafo de landmarks LG. Los literales del estado inicial se almacenan en el nivel 0 de la estructura auxiliar LL, mientras que las metas se almacenan de acuerdo al nivel que ocupan en el grafo de planificación relajado.

- **Selección de un landmark**: Se escoge un landmark ℓ del conjunto de landmarks de LL con el mayor nivel posible y que no se haya escogido aún. No se escogerán los landmarks iniciales (nivel 0), pues de ellos no se pueden obtener más landmarks.

- **Búsqueda de acciones productoras del landmark** ℓ: De entre todas las acciones del problema, $\mathcal{A}$, se selecciona el subconjunto $\lambda_l \subseteq \mathcal{A}$ de acciones que tienen ℓ como efecto, es decir, $\gamma \in \lambda_l \leftrightarrow \ell \in EFF(\gamma)$.

- **Obtención de precondiciones comunes**: Se calculan las precondiciones comunes a todas la acciones del conjunto λ_l, lo que da lugar a un conjunto de literales ρ. Las precondiciones en ρ aparecerán en cualquier plan en que figure también ℓ.

- **Validación de posibles landmarks**: En cada iteración de esta fase, se analiza un literal $\upsilon \in \rho$ para validar si es un landmark. Esta validación se lleva a cabo mediante la construcción de un grafo de planificación relajado. Este procedimiento consta de las siguientes etapas:

  - **Eliminación de acciones productoras del literal** υ: De todas las acciones del problema, $\mathcal{A}$, se eliminan aquellas acciones que

producen el literal v como efecto, es decir, el subconjunto $\lambda_v \subseteq \mathcal{A}$. Esto da lugar a un subconjunto de acciones $\mathcal{A}_{\lambda v} = (\mathcal{A} \setminus \lambda_v)$ que se utilizará para construir el grafo de planificación relajado.

- **Construcción grafo de planificación relajado**: Se construye un grafo de planificación relajado [BF97] usando sólo aquellas acciones que no tienen como efecto el literal v, $\mathcal{A}_{\lambda v}$.
- **Verificación del landmark**: Si todas las metas figuran en el grafo de planificación relajado, el literal v no será considerado landmark. Esto se debe a que al haberse alcanzado todas las metas sin contar con el literal v, éste no es necesario para obtener un plan solución, por lo que no cumple la definición de landmark. Si, por el contrario, no se alcanza alguna de las metas en el grafo, el literal v será considerado landmark.
- **Adición del literal** v: Si el literal v es un landmark, se guardará en la estructura auxiliar LL de acuerdo a su nivel en el grafo relajado. Del mismo modo, v se añadirá al grafo de landmarks LG, estableciéndose también un orden necesario $v \leq_n \ell$, dado que v aparecerá necesariamente antes que ℓ en cualquier plan solución.

4.2.1. Ejemplo de aplicación del algoritmo

Esta sección desarrolla un ejemplo de aplicación del algoritmo de extracción de landmarks. Este ejemplo se basa en el dominio *Satellite*, uno de los benchmarks utilizados en la Competición Internacional de Planning [1].

El dominio Satellite consta de una serie de satélites de comunicaciones (agentes) que incorporan un conjunto de instrumentos para fotografiar estrellas u otros fenómenos. Cada agente satélite dispone de una serie de acciones que le permiten orientarse y operar sus instrumentos. Este problema de ejemplo incluye un único agente satélite.

4.2.1.1. Fichero PDDL del dominio Satellite

A continación, se muestra el código del fichero PDDL correspondiente al dominio Satellite.

[1] http://ipc.icaps-conference.org/

```
(define (domain satellite)
(:requirements :typing :equality :fluents)
(:types agent direction instrument mode - object
        satellite - agent)

(:predicates
  (power-avail ?s - satellite)
  (power-on ?i - instrument)
  (calibrated ?i - instrument)
  (have-image ?d - direction ?m - mode))

(:functions
(pointing ?s - satellite) - direction
(calibration-target ?i - instrument) - direction)

(:multi-functions
(on-board ?s - satellite) - instrument
(supports ?i - instrument) - mode)

(:action turn-to
 :parameters (?s - satellite ?d-new - direction
              ?d-prev - direction)
 :precondition ((= (pointing ?s) ?d-prev))
 :effect (and (assign (pointing ?s) ?d-new)))

(:action switch-on
 :parameters (?i - instrument ?s - satellite)
 :precondition (and (member (on-board ?s) ?i)
                 (power-avail ?s))
 :effect (and (power-on ?i) (not (calibrated ?i))
              (not (power-avail ?s))))

(:action switch-off
 :parameters (?i - instrument ?s - satellite)
 :precondition (and (member (on-board ?s) ?i)
                 (power-on ?i))
 :effect (and (not (power-on ?i)) (power-avail ?s)))

(:action calibrate
 :parameters (?s - satellite ?i - instrument ?d - direction)
 :precondition (and (member (on-board ?s) ?i)
                    (= (pointing ?s) ?d)
                    (= (calibration-target ?i) ?d)
                    (power-on ?i))
 :effect (calibrated ?i))

(:action take-image
 :parameters (?s - satellite ?d - direction ?i - instrument
```

```
              ?m - mode)
 :precondition (and (calibrated ?i) (member (on-board ?s) ?i)
                (member (supports ?i) ?m) (power-on ?i)
                (= (pointing ?s) ?d))
 :effect (have-image ?d ?m)))
```

Figura 4.2: Fichero PDDL del dominio Satellite

Para facilitar la comprensión de este dominio, a continuación se expondrá el significado de las principales secciones del código:

En la sección `:types` de la figura 4.2 se muestra la jerarquía de tipos del dominio Satellite. Como se aprecia en el código, los satélites están definidos como agentes, mientras que el resto de tipos se definen como sigue:

- **Direction:** direcciones (fenómenos espaciales) hacia las que puede orientarse un satélite.
- **Instrument:** instrumentos fotográficos a bordo de los satélites.
- **Mode:** modos de disparo que soporta un instrumento.

La sección `:predicates` de la figura 4.2 se muestra la definición de los predicados del problema, que proporcionan la siguiente información:

- **power-avail:** indica si se puede encender o no el instrumento del satélite.
- **power-on:** su valor muestra si está encendido el instrumento.
- **calibrated:** muestra si el instrumento está calibrado o no.
- **have-image:** indica que se ha obtenido una imagen con un determinado modo de disparo.

La sección `:functions` de la figura 4.2 muestra las variables del dominio. Estas variables tienen el siguiente significado:

- **pointing:** indica en qué dirección está orientado el satélite.
- **calibration-target:** indica la dirección de calibración de un instrumento, esto es, la dirección a la que debe orientarse el satélite para calibrar dicho instrumento.

En la sección `multi:functions` de la figura 4.2 se muestran las multi-funciones del dominio. Estas variables difieren de las anteriores en que pueden contener más de un valor. En este caso, la variable `'on-board'` indica los instrumentos a bordo de un satélite, mientras que la variable `'supports'` devuelve los modos de disparo que soporta un instrumento.

Las diferentes acciones que puede realizar un agente satélite, que pueden verse en la figura 4.2, son las siguientes:

- La acción **`turn-to`** orienta a un satélite `?s`, previamente orientado en una dirección `?d-prev`, a otra dirección nueva `?d-new`.
- En el caso de la acción **`switch-on`**, es necesario que el instrumento `?i` esté a bordo del satélite `?s` y se pueda encender. Los efectos de la acción dejan el instrumento `?i` encendido y sin calibrar.
- La acción **`switch-off`** es opuesta a la anterior. Para que se pueda ejecutar, el instrumento `?i` a bordo del satélite `?s` debe estar encendido. Como efecto de la acción, el instrumento `?i` queda apagado.
- Para ejecutar la acción **`calibrate`** el instrumento `?i` a bordo del satélite `?s` debe estar orientado hacia el fenómeno `?d`, dado que esa es la dirección de calibración para el instrumento `?i`. Como resultado de la acción, `?i` queda calibrado.
- Por último, la acción **`take-image`**, tiene como efecto la obtención de una fotografía de un fenómeno `?d` con un modo `?m`. Para que esta acción pueda llevarse a cabo es necesario que el instrumento `?i` a bordo del satélite `?s` esté calibrado, encendido y apuntando a la dirección `?d`. Asimismo, el instrumento `?i` debe soportar el modo de fotografía `?m`.

4.2.1.2. Fichero PDDL del problema Satellite

A continuación, se muestra el fichero de problema PDDL, que define el estado inicial del problema y las metas a alcanzar.

```
(define (problem strips-sat-x-1)
(:domain satellite)
(:objects
 satellite0 - satellite
 instrument0 - instrument
 infrared0 - mode
```

```
 star1 star4 - direction)

(:init
 (power-avail satellite0)
 (not (power-on instrument0))
 (not (calibrated instrument0))
 (= (calibration-target instrument0) star1)
 (not (have-image star1 infrared0))
 (= (pointing satellite0) star4)
 (= (on-board satellite0) instrument0)
 (= (supports instrument0) infrared0))

(:global-goal (and
 (have-image star1 infrared0))))
```

Figura 4.3: Fichero PDDL del problema del dominio Satellite

Este problema ejemplo incluye un único agente satélite, `satellite0` y un instrumento de fotografía, `instrument0`, que soporta un modo de operación `infrared0`. Se definen dos fenómenos espaciales en el problema, `star1` y `star4`.

El estado inicial del problema contiene la siguiente información:

- Se puede encender el instrumento `instrument0` del satélite `satellite0`.
- El instrumento `instrument0` no está encendido ni calibrado.
- La dirección de calibración del instrumento `instrument0` es el fenómeno `star1`.
- Al comienzo no se dispone de la foto del fenómeno `star1` con el modo `infrared0`.
- El satélite `satellite0` está inicialmente orientado a `star4`.
- El instrumento `instrument0` está a bordo del satélite `satellite0`.
- El instrumento `instrument0` soporta el modo `infrared0`.

Por último, la meta del problema consiste en la obtención de una fotografía del fenómeno `star1` en modo `infrared0`.

4.2.1.3. Traza de la primera iteración del algoritmo

Una vez estudiado el dominio del problema y el estado inicial y las metas, se muestra la traza del algoritmo. Se seguirá el algoritmo descrito con anterioridad y mostrado en la figura 4.1.

- **Inicio:** Se añaden al conjunto de landmarks de LL y al grafo de landmarks LG todos los literales iniciales y finales. Así pues, la estructura LL quedaría:

0	1	2	3
(= (power-avail satellite0) true)			(= (have-image star1 infrared0) true)
(= (power-on instrument0) false)			
(= (calibration-target instrument0) star1)			
(= (have-image star1 infrared0) false)			
(= (pointing satellite0) star4)			
(= (on-board satellite0) instrument0)			
(= (supports instrument0) infrared0)			

  Además, el conjunto de aristas del grafo de landmarks E contendrá el conjunto vacío inicialmente, $E = \emptyset$.

- **Selección de un landmark:** Se comprueba el último nivel de la estructura LL y se escoge el único landmark en ese nivel, `(= (have-image star1 infrared0) true)`. Así pues, se marca como seleccionado y se pasa a la siguiente fase.

- **Búsqueda de acciones productoras del landmark `(= (have-image star1 infrared0) true)`:** Para este landmark solo hay una acción productora, `(take-image satellite0 star1 instrument0 infrared0)`.

- **Obtención de precondiciones comunes:** Dado que sólo se dispone de una acción productora, todas las precondiciones de dicha acción son comunes, y por tanto, candidatas a landmark. Por tanto, se procede a la **validación de posibles landmarks**. La acción consta de cuatro precondiciones:

  - `(= (on-board satellite0) instrument0)`

- `(= (calibrated instrument0) true)`
- `(= (power-on instrument0) true)`
- `(= (pointing satellite0) star1)`

Como la precondición `(= (on-board satellite0) instrument0)` ya ha sido identificada como landmark (al pertenecer al estado inicial del problema), el procedimiento se centra en las otras tres precondiciones. En primer lugar, se selecciona `(= (power-on instrument0) true)` para su validación.

- **Eliminación de acciones productoras de (= (calibrated instrument0) true):** En primer lugar, se eliminan del conjunto de acciones del agente `satellite0` aquellas que tienen como efecto `(= (calibrated instrument0) true)`. La única acción que cumple esta condición es `(calibrate satellite0 instrument0 star4)`. De este modo se construye un grafo de planificación relajado utilizando el conjunto de acciones $\mathcal{A}$ / `(calibrate satellite0 instrument0 star4)`.

- **Expansión de literales:** A partir del nuevo conjunto de acciones definido, se expande el grafo de planificación relajado desde el estado inicial.

- **Comprobación valor de landmark:** Una vez expandidos todos los literales, se comprueba si se han alcanzado todas las metas. En este ejemplo, no ha sido posible alcanzar la meta ya que, como se ha explicado anteriormente, el literal `(= (calibrated instrument0) true)` es necesario para obtener la fotografía descrita en la meta. Por tanto, `(= (calibrated instrument0) true)` queda validado como landmark.

- **Adición del literal `(= (calibrated instrument0) true)`:** Como el literal `(= (calibrated instrument0) true)` ha resultado ser un landmark, se añade al nivel correspondiente de la estructura LL , quedando ésta así:

0	1	2	3
(= (power-avail satellite0) true)		(= (calibrated instrument0) true)	(= (have-image star1 infrared0) true)
(= (power-on instrument0) false)			
(= (calibration-target instrument0) star1)			
(= (have-image star1 infrared0) false)			
(= (pointing satellite0) star4)			
(= (on-board satellite0) instrument0)			
(= (supports instrument0) infrared0)			

Además, se añade al conjunto de aristas E del grafo de landmarks LG, hasta ahora vacío, la arista `(= (calibrated instrument0) true)` $\leq_n$ `(= (have-image star1 infrared0) true)`, quedando éste así:

Conjunto de aristas E
`(= (calibrated instrument0) true)` $\leq_n$ `(= (have-image star1 infrared0) true)`

- Después de este punto, el algoritmo pasa a validar el siguiente candidato del conjunto de precondiciones comunes. Se elige **(= (power-on instrument0) true)**.

- Siguiendo los pasos anteriores, el literal **`(= (power-on instrument0) true)`** será validado como landmark, pues sin él no se consigue la meta `(= (have-image star1 infrared0) true)` en el grafo relajado. Por tanto, se añade al grafo de landmarks LG y al nivel correspondiente de la estructura de landmarks LL, quedando esta estructura de la siguiente manera:

0	1	2	3
(= (power-avail satellite0) true)	(= (power-on instrument0) true)	(= (calibrated instrument0) true)	(= (have-image star1 infrared0) true)
(= (power-on instrument0) false)			
(= (calibration-target instrument0) star1)			
(= (have-image star1 infrared0) false)			
(= (pointing satellite0) star4)			
(= (on-board satellite0) instrument0)			
(= (supports instrument0) infrared0)			

También se añade al conjunto de aristas E la arista `((power-on instrument0) true)` $\leq_n$ `(= (have-image star1 infrared0) true)`, quedando éste así:

Conjunto de aristas E
`(= (calibrated instrument0) true)` $\leq_n$ `(= (have-image star1 infrared0) true)`
`(= (power-on instrument0) true)` $\leq_n$ `(= (have-image star1 infrared0) true)`

- A continuación, se verifica el último candidato del conjunto de precondiciones comunes, **`(= (pointing satellite0) star1)`**, que queda validado como landmark. Por tanto, el literal se añade también a las estructuras LG y LL, quedando esta última de la siguiente manera:

0	1	2	3
`(= (power-avail satellite0) true)`	`(= (power-on instrument0) true)`	`(= (calibrated instrument0) true)`	`(= (have-image star1 infrared0) true)`
`(= (power-on instrument0) false)`	`(= (pointing satellite0) star1)`		
`(= (calibration-target instrument0) star1)`			
`(= (have-image star1 infrared0) false)`			
`(= (pointing satellite0) star4)`			
`(= (on-board satellite0) instrument0)`			
`(= (supports instrument0) infrared0)`			

El conjunto de aristas E queda como sigue:

Conjunto de aristas E
`(= (calibrated instrument0) true)` $\leq_n$ `(= (have-image star1 infrared0) true)`
`(= (power-on instrument0) true)` $\leq_n$ `(= (have-image star1 infrared0) true)`
`(= (pointing satellite0) star1)` $\leq_n$ `(= (have-image star1 infrared0) true)`

En este punto, la traza volvería a la fase de **Selección de un landmark**, donde se comienza a explorar el nivel 2 de la estructura LL, extrayendo las precondiciones de cada landmark e identificando posibles nuevos landmarks. El proceso iterativo continúa hasta que no quedan landmarks por analizar en la estructura LL.

4.3. Adaptación Multiagente

Al llevar a cabo la adaptación multiagente del grafo de landmarks se encontraron diferentes problemas derivados de la información privada de la que disponen los agentes y del hecho de que cada agente pueda ejecutar acciones distintas al resto.

El algoritmo de extracción de landmarks multiagente parte de la construcción de un grafo de planificación distribuido [ZNK07]. Cada agente calcula un grafo de planificación relajado individualmente, y a continuación, los agentes se intercambian los literales compartibles `(:shared-data)` que figuran en sus grafos individuales. Tras el intercambio, los agentes vuelven a extender sus grafos individualmente. Este proceso se repite hasta que no aparecen nuevos literales en ninguno de los grafos de los agentes.

Como resultado, cada agente dispone de un grafo de planificación distinto, que refleja su visión incompleta de la información del problema. Como se indica en la sección 4.1, cada literal l del problema dispone de una etiqueta a_l que muestra qué agentes pueden alcanzarlo. Los literales alcanzados por más de un agente se ubican en el menor nivel posible de los grafos de dichos agentes. Es decir, si un agente i consigue un literal l en el nivel 1 de su grafo, y un agente j lo consigue en el nivel 2, ambos guardarán el literal en el nivel 1.

Al igual que en el caso del grafo de planificación relajado, cada agente dispone de su propia versión del grafo de landmarks, cuyos contenidos dependen de la visión parcial del problema que mantiene el agente.

Esta sección describe los cambios realizados a las fases del algoritmo de extracción de landmarks presentado en la sección anterior (ver Figura 4.1) para llevar a cabo su adaptación a un contexto multiagente.

4.3.1. Selección de un landmark

Al igual que en el procedimiento monoagente, el estado inicial y las metas se almacenan por niveles. Los literales del estado inicial se guardan en el nivel 0 de la estructura LL, mientras que las metas se almacenan según su nivel en el grafo de planificación multiagente. El algoritmo de extracción de landmarks multiagente explora la estructura LL de forma regresiva, comenzando por el último nivel.

Para garantizar la adecuada sincronización entre agentes, uno de los agentes asumirá el rol de *coordinador*, encargándose de liderar el proceso. Este rol rotará entre los agentes participantes en cada iteración del algoritmo, lo que garantiza que todos los agentes asumirán en algún momento el rol de coordinador.

El agente *coordinador* comprueba el último nivel de su estructura LL donde quedan landmarks por analizar (en adelante nivel n), seleccionando un landmark l. A continuación, el *coordinador* aplica el proceso de *obtención de precondiciones comunes* para buscar más landmarks a partir de l. Tras analizar l, todos los agentes que consiguen l lo marcan como visitado en sus estructuras.

Una vez el agente haya terminado de explorar el nivel n de su estructura LL, el rol de agente *coordinador* rotará entre los agentes, de modo que el siguiente agente *coordinador* procederá a analizar los landmarks en el nivel n de su estructura.

El rol de *coordinador* seguirá rotando entre los agentes hasta que el primer agente que asumió dicho rol lo recupere. Esto indica que todos los agentes han explorado el nivel n de su estructura LL, y por tanto, se puede comenzar a analizar el nivel inmediatamente anterior.

Una vez se alcance el nivel 0 de la estructura LL, donde están ubicados los landmarks correspondientes al estado inicial, el algoritmo de extracción de landmarks multiagente concluye devolviendo, para cada agente i, su versión del grafo de landmarks, LG^i.

4.3.2. Obtención de precondiciones comunes

En este proceso, el agente *coordinador* calcula las precondiciones comunes a las acciones que producen el landmark l. Inicialmente, el agente *coordinador* verifica únicamente las acciones productoras que él mismo puede ejecutar, obteniendo así un conjunto de precondiciones comunes λ^c.

Si el landmark l es generado por otros agentes además del *coordinador* (esto es, si l está etiquetado con más de un agente), éste se comunicará con dichos agentes para solicitar su colaboración. Cada agente i productor del landmark l calculará las precondiciones comunes a sus acciones productoras. El conjunto de precondiciones comunes, λ^i, será enviado de vuelta al agente

coordinador.

Una vez el agente *coordinador* reciba todos los conjuntos de precondiciones comunes λ^i, éste calculará la intersección de dichos conjuntos, obteniendo como resultado el conjunto λ_l de precondiciones comunes al landmark l. Por tanto, $\lambda_l = \bigcap_{\forall i \in \mathcal{AG} | i \in a_l} \lambda^i$.

En el caso de los landmarks correspondientes a metas del problema, el cálculo varía ligeramente, dado que las metas son visibles para todos los agentes, pero no necesariamente son alcanzables por todos ellos. Para asegurar que se calculan las precondiciones comunes correctamente, sólo se intersecan los conjuntos λ^i no vacíos. Por tanto, dado un landmark g tal que $g \in \mathcal{G}$, $\lambda_g = \bigcap_{\forall i \in \mathcal{AG} | i \in a_l \wedge \lambda^i \neq \emptyset} \lambda^i$.

4.3.3. Validación de posibles landmarks

Una vez obtenido el conjunto de literales comunes, λ_l, los agentes proceden a validar cada literal candidato $lc \in \lambda_l$. Para ello, cada agente i elimina de su conjunto de acciones $\mathcal{A}^i$ el conjunto de acciones productoras del literal lc, $\Upsilon^i(lc) \subseteq \mathcal{A}^i$, obteniendo como resultado un subconjunto de acciones $\mathcal{A}^i_{lc} = \mathcal{A}^i \setminus \Upsilon^i(lc)$.

A continuación, los agentes desarrollan un grafo de planificación multiagente como el descrito en la sección 4.2, de modo que cada agente i utiliza únicamente las acciones en el conjunto $\mathcal{A}^i_{lc}$.

Los agentes expanden, por turnos, sus respectivos grafos de planificación relajados. A continuación, se intercambian los literales compartibles que han introducido en sus grafos. Cuando ninguno de los agentes consiga introducir nueva información en el grafo, el proceso de construcción finaliza.

Una vez completado el grafo de planificación multiagente, los agentes comprueban que ninguno de ellos haya alcanzado todas las metas del problema. En tal caso, el literal lc es validado como landmark, dado que resulta imprescindible para alcanzar un plan solución.

4.3.4. Adición del literal v

Si el literal lc es un landmark, el agente *coordinador* solicitará al resto de agentes etiquetados como productores del landmark lc que lo añadan

a sus respectivos grafos de landmarks LG . Del mismo modo, los agentes productores introducirán un orden necesario $lc \leq_n l$ en el conjunto de aristas E de su grafo de landmarks LG, dado que lc debe aparecer necesariamente antes que l en cualquier plan solución.

4.3.5. Ejemplo de aplicación del algoritmo multiagente

Esta sección presenta una versión multiagente del problema ejemplo mostrado en la sección 4.2.1. En este caso, el problema incluye dos agentes satélite.

Dado que ambos agentes son satélites, y por tanto poseen las mismas capacidades, el problema incluye un único fichero de dominio. Cada agente, a su vez, dispone de su propio fichero de problema, que especifica su situación inicial (orientación, instrumentos a bordo del satélite, modos de operación soportados por cada instrumento, etc).

Además de los objetos, estado inicial y metas, los ficheros de problema incluyen una sección adicional `:shared-data` en la que se define la información compartible, esto es, los literales que el agente puede compartir con el resto de agentes involucrados en la tarea.

4.3.5.1. Fichero PDDL de dominio multiagente

A continuación, se analizan los principales cambios introducidos en la versión multiagente del fichero de dominio PDDL.

```
(define (domain satellite)
(:requirements :typing :equality :fluents)
(:types agent direction instrument mode - object
   satellite - agent)

(:predicates
   (power-avail ?s - satellite)
   (mySatellite ?s - satellite)
   (power-on ?i - instrument)
   (calibrated ?i - instrument)
   (have-image ?d - direction ?m - mode))

(:functions
   (pointing ?s - satellite) - direction
   (calibration-target ?i - instrument) - direction)

(:multi-functions
   (on-board ?s - satellite) - instrument
   (supports ?i - instrument) - mode)

(:action turn-to
   :parameters (?s - satellite ?d-new - direction
                ?d-prev - direction)
   :precondition (and (mySatellite ?s - satellite)
                 (= (pointing ?s) ?d-prev))
   :effect (and (assign (pointing ?s) ?d-new)))

(:action switch-on
   :parameters (?i - instrument ?s - satellite)
   :precondition (and (mySatellite ?s - satellite)
                      (member (on-board ?s) ?i)
                      (power-avail ?s))
   :effect (and (power-on ?i) (not (calibrated ?i))
                (not (power-avail ?s))))

(:action switch-off
   :parameters (?i - instrument ?s - satellite)
   :precondition (and (mySatellite ?s - satellite)
                      (member (on-board ?s) ?i) (power-on ?i))
   :effect (and (not (power-on ?i)) (power-avail ?s)))

(:action calibrate
 :parameters (?s - satellite ?i - instrument ?d - direction)
 :precondition (and (mySatellite ?s - satellite)
                    (member (on-board ?s) ?i)
                    (= (pointing ?s) ?d)
                    (= (calibration-target ?i) ?d)
                    (power-on ?i))
```

```
  :effect (calibrated ?i))

(:action take-image
 :parameters (?s - satellite ?d - direction ?i - instrument
                  ?m - mode)
 :precondition (and (mySatellite ?s - satellite)
                    (calibrated ?i)
                    (member (on-board ?s) ?i)
                    (member (supports ?i) ?m)
                    (power-on ?i) (= (pointing ?s) ?d))
 :effect (have-image ?d ?m)))
```

Figura 4.4: Fichero PDDL del dominio multiagente Satellite

El predicado **(mySatellite ?s - satellite)** indica qué objeto satélite corresponde a cada agente. Todas las acciones utilizan el predicado anterior como precondición para garantizar que cada agente ejecuta únicamente las acciones asociadas a su satélite.

4.3.5.2. Fichero PDDL de problema multiagente

El fichero de problema de cada agente introduce una serie de cambios respecto al fichero monoagente homónimo. Por simplicidad, en esta sección se ilustran dichos cambios usando únicamente el fichero de problema asociado al `satellite0`.

```
(define (problem strips-sat-x-1)
(:domain satellite
(:objects
     satellite0 satellite1 - satellite
     instrument0 instrument1 - instrument
     infrared0 spectrograph2 - mode
     star1 star4 phenomenon7 - direction)
(:shared-data
     ((pointing ?s - satellite) - direction)
     (have-image ?d - direction ?m - mode) - satellite1)
(:init (mySatellite satellite0)
     (power-avail satellite0)
     (not (power-on instrument0))
     (not (calibrated instrument0))
     (= (calibration-target instrument0) star1)
     (= (calibration-target instrument1) star4)
     (not (have-image star1 infrared0))
     (not (have-image phenomenon7 spectrograph2))
     (= (pointing satellite0) star4)
     (= (on-board satellite0) instrument0)
     (not (= (onboard satellite0) instrument1))
```

```
    (= (supports instrument0) infrared0)
    (not (= (supports instrument0) spectrograph2))
(:global-goal (and
    (have-image star1 infrared0)
    (have-image phenomenon7 spectrograph2))))
```

Figura 4.5: Fichero PDDL del problema del dominio multiagente Satellite

Cada fichero de problema introduce una sección adicional `:shared-data` donde se define qué literales puede compartir el agente y con quién. Como se puede ver en la figura 4.5, se comparten los literales relativos a la dirección de los satélites y las fotografías que se han tomado. Del mismo modo, en el fichero de problema del agente `satellite1`, se especifica la información que este agente comparte con el `satellite0`.

El estado inicial del agente `satellite0` se puede ver en la sección `:init` de la figura 4.5. Como se muestra en el código, el literal `(mysatellite satellite0)` asocia al agente con el objeto `satellite0`. Asimismo, el estado inicial sólo especifica la información relativa al `satellite0` (orientación inicial, instrumentos a bordo, etc), dado que la configuración del `satellite1` no es conocida por el `satellite0`. A su vez, en el fichero de problema del `satellite1` se especifica la información relativa a dicho agente.

Las metas del problema, definidas en la sección `:global-goal` de la figura 4.5, son compartidas por ambos agentes, y consisten en la obtención de una foto del fenómeno `star1` con el modo `infrared0` y otra foto del fénomeno `phenomenon7` con el modo `spectrograph2`.

4.3.5.3. Traza del algoritmo multiagente

Una vez detallada la información del dominio, del estado inicial y las metas a conseguir, esta sección desarrolla una traza de la primera iteración del bucle para la obtención de los landmarks de este problema.

- **Inicio**: Los satélites `satellite0` y `satellite1` construyen un grafo de planificación multiagente, y añaden a sus estructuras LL y al grafo de landmarks LG todos los literales iniciales y las metas. Como por definición las metas son conocidas por todos los agentes (pese a que alguno de los agentes no las pueda conseguir), se añaden a los grafos de landmarks de ambos agentes. Así pues, las dos estructuras LL^0 (estructura del `satellite0`) y LL^1 (estructura del `satellite1`) se muestran en los cuadros 4.1 y 4.2, respectivamente:

0	1	2	3
[s0](= (power-avail satellite0) true)			[s0,s1](= (have-image star1 infrared0) true)
[s0](= (power-on instrument0) false)			[s0,s1](= (have-image phenomenon7 spectrograph2) true)
[s0](= (calibration-target instrument0) star1)			
[s0](= (on-board satellite0) instrument0)			
[s0](= (supports instrument0) infrared0)			
[s0,s1](= (pointing satellite0) star4)			
[s0,s1](= (pointing satellite1) star1)			
[s0,s1](= (have-image star1 infrared0) false)			
[s0,s1](= (have-image phenomenon7 spectrograph2) false)			

Cuadro 4.1: LL^0 del `satellite0`

Por otra parte, tanto E^0 como E^1 se inicializan al conjunto vacío $\emptyset$. Se asume que el agente `satellite0` actúa como *coordinador* inicialmente.

- **Selección de un landmark**: El agente *coordinador* `satellite0` comprueba el último nivel de su estructura LL^0, seleccionando el landmark `(= (have-image star1 infrared0) true)`.

- **Búsqueda de acciones productoras del landmark `(= (have-image star1 infrared0) true)`:** La única acción productora que puede realizar el agente `satellite0` para conseguir este landmark es `(take-image satellite0 star1 instrument0 infrared0`.

- **Obtención de precondiciones comunes**: Dado que sólo se dispone de una acción productora, todas las precondiciones de dicha acción son comunes, y por tanto, candidatas a landmark. Es necesario intersecar dichas precondiciones con las que el `satellite1` envíe al *coordinador*. Por ello, el `satellite0` pide al `satellite1` que calcule sus precondiciones comunes.

  En este caso, no hay ninguna acción que el `satellite1` pueda realizar para obtener el landmark, puesto que `instrument0`, el único instrumento capaz de obtener imágenes en modo `infrared0`, no se halla a bordo del `satellite1`. Puesto que el `satellite1` no dispone de ninguna acción productora, sus precondiciones comunes son el conjunto vacío $\emptyset$.

0	1	2	3
[s1](= (power-avail satellite1) true)			[s0,s1](= (have-image star1 infrared0) true)
[s1](= (power-on instrument1) false)			[s0,s1](= (have-image phenomenon7 spectrograph2) true)
[s1](= (calibration-target instrument1) star4)			
[s1](= (on-board satellite1) instrument1)			
[s1](= (supports instrument1) spectrograph2)			
[s0,s1](= (pointing satellite0) star4)			
[s0,s1](= (pointing satellite1) star1)			
[s0,s1](= (have-image star1 infrared0) false)			
[s0,s1](= (have-image phenomenon7 spectrograph2) false)			

Cuadro 4.2: LL^1 del satellite1

Dado que el satellite1 devuelve al satellite0 el conjunto vacío $\emptyset$, no se interseca este conjunto con el obtenido previamente por el satellite0. Así pues, el conjunto de precondiciones a validar es el siguiente:

- (= (on-board satellite0) instrument0)
- (= (calibrated instrument0) true)
- (= (power-on instrument0) true)
- (= (pointing satellite0) star1)

Como la precondición (= (on-board satellite0) instrument0) ya ha sido identificada como landmark (al pertenecer al estado inicial del problema), el procedimiento se centra en las otras tres precondiciones.

En primer lugar, se selecciona (= (power-on instrument0) true) para su validación.

- **Eliminación de acciones productoras de (= (calibrated instrument0) true):** En primer lugar, el agente satellite0 elimina de su conjunto de acciones aquellas que tienen como efecto (= (calibrated instrument0) true). La única acción que cumple esta condición es (calibrate satellite0 instrument0 star4).

  Dado que el agente satellite0 juega el rol de coordinador, éste pide al satellite1 que elimine de su conjunto de acciones aquellas

que tengan como efecto `(= (calibrated instrument0) true)`. Como el satélite `satellite1` no puede conseguir como efecto el literal `(= (calibrated instrument0) true)`, no elimina ninguna acción.

- **Expansión de literales:** A partir de los nuevos conjuntos de acciones calculados, ambos agentes construyen un grafo de planificación multiagente.
- **Comprobación del valor de landmark por el *coordinador*:** Una vez expandidos todos los literales, se comprueba si se han alcanzado todas las metas. En este caso, no ha sido posible alcanzar la meta `(= (have-image star1 infrared0) true)`, por lo que la precondición `(= (calibrated instrument0) true)` es un landmark.
- **Adición del literal `(= (calibrated instrument0) true)`:** El literal `(= (calibrated instrument0) true)` se añade a las estructuras de los agentes que pueden conseguirlo. En este caso, solo el agente `satellite0` lo consigue, por lo que añade el literal a sus estructuras LG^0 yLL^0, así como el correspondiente orden necesario a su estructura E^0.

0	1	2	3
[s0](= (power-avail satellite0) true)		[s0](= (calibrated instrument0) true)	[s0,s1](= (have-image star1 infrared0) true)
[s0](= (power-on instrument0) false)			[s0,s1](= (have-image phenomenon7 spectrograph2) true)
[s0](= (calibration-target instrument0) star1)			
[s0](= (on-board satellite0) instrument0)			
[s0](= (supports instrument0) infrared0)			
[s0,s1](= (pointing satellite0) star4)			
[s0,s1](= (pointing satellite1) star1)			
[s0,s1](= (have-image star1 infrared0) false)			
[s0,s1](= (have-image phenomenon7 spectrograph2) false)			

Cuadro 4.3: LL^0 del `satellite0`

- **Validación precondición `(= (power-on instrument0) true)`**: A continuación, se valida la siguiente precondición, `(= (power-on instrument0)`

Conjunto de aristas E^0
(= (calibrated instrument0) true) $\leq_n$ (= (have-image star1 infrared0) true)

true), comprobándose que es un landmark que sólo consigue el agente satellite0. Por tanto, la información referente a este landmark se añade a las estructuras LG^0, LL^0 y E^0. El estado de estas estructuras es el siguiente:

0	1	2	3
[s0](= (power-avail satellite0) true)	[s0](= (power-on instrument0) true)	[s0](= (calibrated instrument0) true)	[s0,s1](= (have-image star1 infrared0) true)
[s0](= (power-on instrument0) false)			[s0,s1](= (have-image phenomenon7 spectrograph2) true)
[s0](= (calibration-target instrument0) star1)			
[s0](= (on-board satellite0) instrument0)			
[s0](= (supports instrument0) infrared0)			
[s0,s1](= (pointing satellite0) star4)			
[s0,s1](= (pointing satellite1) star1)			
[s0,s1](= (have-image star1 infrared0) false)			
[s0,s1](= (have-image phenomenon7 spectrograph2) false)			

Cuadro 4.4: LL^0 del satellite0

Conjunto de aristas E^0
(= (calibrated instrument0) true) $\leq_n$ (= (have-image star1 infrared0) true)
(= (power-on instrument0) true) $\leq_n$ (= (have-image star1 infrared0) true)

- **Validación precondición (= (pointing satellite0) star1)** : Este caso es similiar al anterior, ya que la precondición (= (pointing satellite0) star1) es un landmark que sólo consigue el agente satellite0. Por tanto, la información del landmark sólo se añade a las estructuras LG^0, LL^0 y E^0. Así quedaría la estructura LL^0:

  Y la estructura E^0:

0	1	2	3
[s0](= (power-avail satellite0) true)	[s0](= (power-on instrument0) true)	[s0](= (calibrated instrument0) true)	[s0,s1](= (have-image star1 infrared0) true)
[s0](= (power-on instrument0) false)	[s0](= (pointing satellite0) star1)		[s0,s1](= (have-image phenomenon7 spectrograph2) true)
[s0](= (calibration-target instrument0) star1)			
[s0](= (on-board satellite0) instrument0)			
[s0](= (supports instrument0) infrared0)			
[s0,s1](= (pointing satellite0) star4)			
[s0,s1](= (pointing satellite1) star1)			
[s0,s1](= (have-image star1 infrared0) false)			
[s0,s1](= (have-image phenomenon7 spectrograph2) false)			

Cuadro 4.5: LL^0 del satellite0

Conjunto de aristas E^0
(= (calibrated instrument0) true) $\leq_n$ (= (have-image star1 infrared0) true)
(= (power-on instrument0) true) $\leq_n$ (= (have-image star1 infrared0) true)
(= (pointing satellite0) star1) $\leq_n$ (= (have-image star1 infrared0) true)

En este punto, la traza volvería a la fase de **selección de un landmark**, donde el agente satellite0 mantendrá el rol de coordinador. Este seguirá seleccionando landmarks hasta explorar completamente el nivel actual de la estructura LL^0. Una vez el agente satellite0 termine de explorar el último nivel de LL^0, el agente satellite1 asumirá el rol de *coordinador* para explorar el último nivel de LL^1. Cuando el agente satellite1 termine de explorar el último nivel de LL^1, el agente satellite0 recuperará el rol de *coordinador*, pasando a explorar el nivel immediatamente anterior de la estructura LL^0.

El proceso iterativo continuará hasta que no quedan landmarks por analizar en las estructuras LL^0 y LL^1.

Tras finalizar el algoritmo, éste devuelve los siguientes grafos de landmarks:

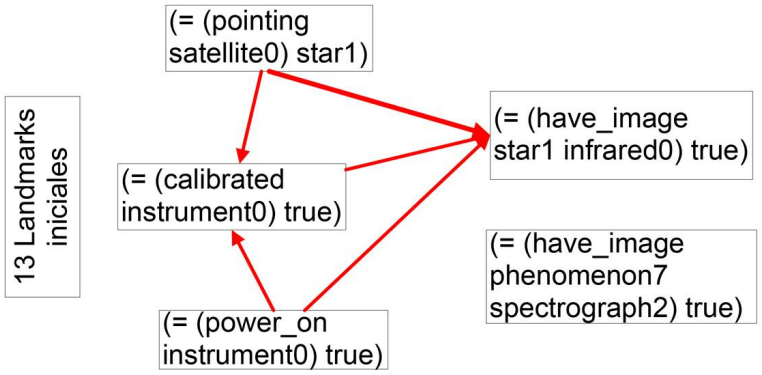

Figura 4.6: Grafo de landmarks del agente `satellite0`

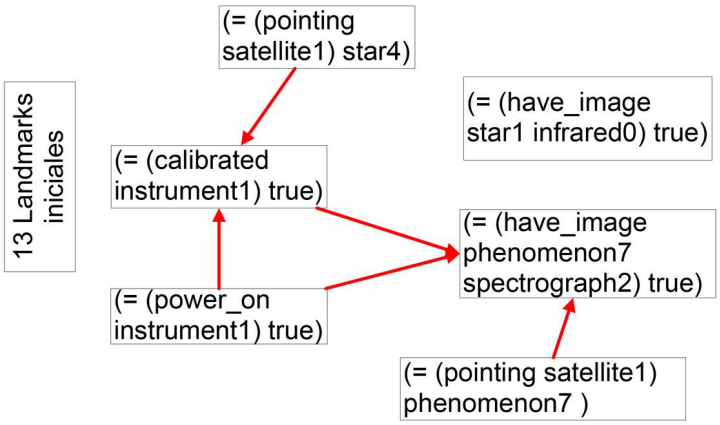

Figura 4.7: Grafo de landmarks del agente `satellite1`

4.4. Heurística FMAP-land

La finalidad básica del cálculo de landmarks y órdenes necesarios reside en su uso como base para mejorar la eficiencia de la búsqueda heurística en sistemas de planificación [RW10].

El objetivo del algoritmo desarrollado en este Proyecto Final de Carrera se centra en la aplicación del árbol de landmarks multiagente para el desarrollo de una función heurística que, combinada con la heurística de FMAP, h_{fmap}, busca mejorar la calidad de las estimaciones de dicho planificador.

Concretamente, cada plan Π en FMAP-land se evalúa mediante una función $f(\Pi) = g + 2 * h_{fmap}(\Pi) + h_{land}(\Pi)$, donde h_{land} se define como el número de landmarks por conseguir en el plan Π. Por tanto, h_{land} = /*N*/-*landmarksAlcanzados(*Π*)*. El valor *landmarksAlcanzados*(Π) se calcula como se describe a continuación:

- Se linealiza el plan Π, obteniéndose una secuencia de acciones.
- Se analizan por orden las acciones, verificándose los landmarks que aparecen como efectos de cada acción.
- Se comprueba que los landmarks aparezcan en el orden definido en el árbol de landmarks. Por cada landmark encontrado en el orden correcto, se incrementa el valor de *landmarksAlcanzados*(Π) en una unidad.

Capítulo 5

Resultados Experimentales

Este capítulo analiza los resultados experimentales obtenidos al aplicar el algoritmo de extracción de landmarks en problemas de dos dominios de la Competición Internacional de Planning [1]: *Rovers* y *Satellite*. Estos dominios presentan las siguientes características:

- **Rovers:** Los agentes en este dominio son un conjunto de rovers, vehículos de exploración espacial diseñados para desplazarse sobre la superficie de otros planetas. Los rovers tienen como objetivo la recolección de tres tipos de muestras: fotografías, muestras de tierra y rocas. Cada rover está equipado con una serie de instrumentos que le capacitan para la recolección de muestras de cada tipo. Una vez han recogido las muestras, los rovers transmiten la información obtenida a una estación espacial.

- **Satellite:** Este dominio se ha utilizado en los ejemplos del capítulo 4. El dominio consta de un conjunto de satélites de comunicaciones que tienen como objetivo la obtención de fotografías de determinados fenómenos espaciales. Cada satélite dispone de una serie de instrumentos fotográficos a bordo que le permiten obtener fotografías mediante distintos modos de operación.

Este Proyecto de Final de Carrera se centra en el cálculo de *landmarks* simples, obviando los *landmarks disyuntivos*, que serán implementados como parte del trabajo futuro a desarrollar. Un *landmark disyuntivo* es una disyunción de literales que actúa como un nodo del *grafo de landmarks*. Pese a que no tienen utilidad por sí mismos, los *landmarks disyuntivos* facilitan la extracción de nuevos *landmarks* simples que no podrían hallarse de otro

[1] http://ipc.icaps-conference.org/

modo.

En muchos de los problemas de la Competición Internacional de Planning, la eliminación del cálculo de *landmarks disyuntivos* condiciona notablemente el número de *landmarks* extraídos por el algoritmo. Por ello, los problemas seleccionados para la experimentación han sido objeto de diversas modificaciones con el fin de maximizar el número de *landmarks* obtenidos.

Para la experimentación se han seleccionado tres problemas de cada dominio y se han modificado para reducir en lo posible el impacto de los *landmarks disyuntivos*. Estos cambios permiten la obtención de un número significativo de *landmarks* para cada problema.

5.1. Dominio *Rovers*

El Cuadro 5.1 muestra los resultados experimentales obtenidos para el dominio *Rovers*:

# Landmarks extraídos del dominio Rovers				
Problema	Agentes	Estado inicial	Metas	Otros Landmarks
Pfile1	rover0	45	2	2
	rover1	45	2	2
Pfile2	rover0	53	2	0
	rover1	53	2	4
Pfile3	rover0	59	6	2
	rover1	59	6	5
	rover2	59	6	4

Cuadro 5.1: Resultados de los problemas del dominio *Rovers*

El Cuadro 5.1 indica, para cada problema y agente participante, el número de landmarks obtenidos. Los literales pertenecientes al estado inicial y las metas del problema son siempre landmarks por definición. La columna *Otros landmarks*, por su parte, muestra los landmarks que no pertenecen a ninguno de esos dos conjuntos.

Para ilustrar con más detalle los resultados obtenidos, las Figuras 5.1, 5.2 y 5.3 muestran los grafos de landmarks obtenidos en el problema *Pfile3* por los agentes `rover0`, `rover1` y `rover2`, respectivamente. Este problema

se ha modelado de modo que cada agente puede recoger un tipo de muestra diferente: el `rover0` puede hacer fotografías, mientras que el `rover1` está equipado con instrumental para recoger rocas y el `rover2` está capacitado para recoger y analizar muestras de tierra. El problema incluye seis metas, consistentes en la obtención de una fotografía, tres rocas y dos muestras de tierra.

La figura 5.1 muestra un orden necesario entre el landmark `(= (have-image rover0 objective0 high-res) true)` y la meta `(= (communicated-image-data objective0 high-res) true)`. Esto indica que es necesario conseguir este landmark antes que la meta en todo plan solución. Como se puede comprobar intuitivamente, antes de transmitir una imagen, es necesario haberla obtenido. Del mismo modo, se puede observar un orden necesario entre los literales `(= (calibrated camera0 rover0) true)` y `(= (have-image rover0 objective0 high-res) true)`, dado que es necesario calibrar el instrumental fotográfico antes de obtener la fotografía.

La Figura 5.2 ilustra el grafo de landmarks obtenido por el agente `rover1`. Dado que este agente está especializado en la recolección de rocas, se aprecian tres componentes conexas en torno a las tres metas relacionadas con las rocas.

Inicialmente, el `rover1` está situado en el `waypoint3`, por lo que puede recoger directamente una roca en dicha localización: `(= (have-rock-analysis rover1 waypoint3) true)`. Por tanto, el agente no necesita desplazarse, sólo debe llevar a cabo el análisis de la muestra y a continuación comunicar los datos, lo que se refleja en el orden necesario `(= (have-rock-analysis rover1 waypoint3) true)` $\leq_n$ `(= (communicated-rock-data waypoint3) true)`.

Para alcanzar las otras dos metas mostradas en la figura 5.2, el agente `rover1` deberá desplazarse necesariamente hasta las localizaciones correspondientes (`waypoint2` y `waypoint4`) antes de analizar las muestras y comunicarlas. Como se observa en el grafo, para cada una de estas metas hay dos órdenes necesarios que reflejan esta secuencia de landmarks.

La Figura 5.3 ilustra el grafo de landmarks obtenido por el agente `rover2`. Ya que este agente está especializado en la recolección de tierra, se aprecian dos componentes conexas en torno a las dos metas relacionadas con la comunicación de muestras de tierra.

Inicialmente, el `rover2` está situado en el `waypoint4`, por lo que puede recoger directamente una muestra de tierra en esa localización: `(= (have-soil-`

`analysis rover2 waypoint4) true)`. A continuación, el agente analizará la muestra y comunicará sus resultados, lo cual se refleja en el orden necesario `(=(have-soil-analysis rover2 waypoint4) true)` $\leq_n$ `(=(communicated -soil-data waypoint4) true)`.

Para alcanzar la siguiente meta, `(= (communicated-soil-data waypoint1) true)`, el agente debe desplazarse a la posición `waypoint1`. Para llegar al `waypoint1`, al agente debe atravesar necesariamente el `waypoint0`, lo que se muestra en el orden necesario `(= (at rover2) waypoint0)` $\leq_n$ `(= (at rover2) waypoint1)`.

Cuando el agente se encuentre en el `waypoint1` puede recoger la muestra de tierra, lo que se refleja en el orden necesario `(= (at rover2) waypoint1)` $\leq_n$ `(= (have-soil-analysis rover2 waypoint1) true)`. Por último, el agente debe analizar la muestra y comunicar sus resultados, dando lugar al orden necesario`(= (have-soil-analysis rover2 waypoint1) true)` $\leq_n$ `(= (communicated-soil-data waypoint1) true)`.

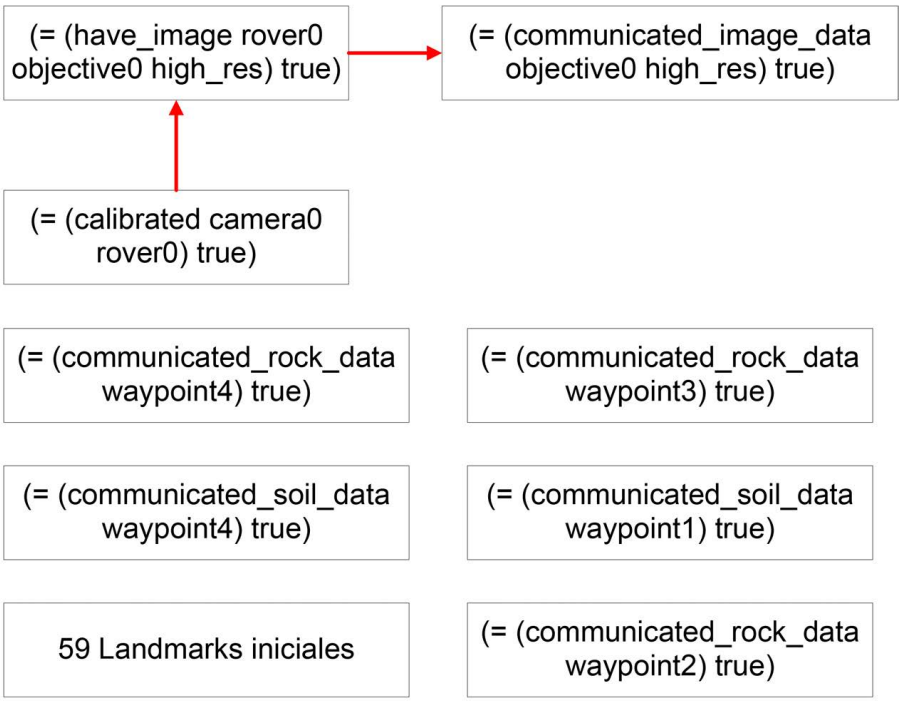

Figura 5.1: Estructura LG^0 del agente `rover0`

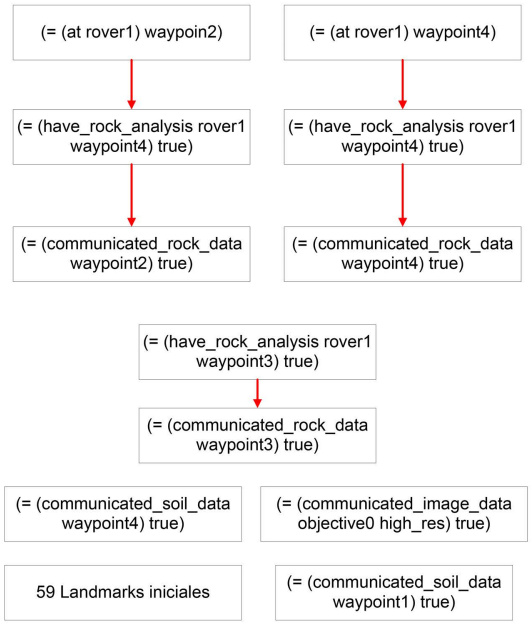

Figura 5.2: Estructura LG^1 del agente `rover1`

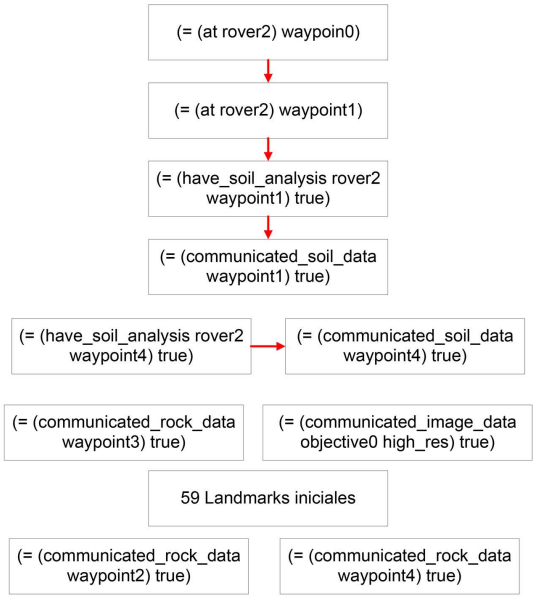

Figura 5.3: Estructura LG^2 del agente `rover2`

5.2. Dominio *Satellite*

El Cuadro 5.2 muestra los resultados experimentales obtenidos para el dominio *Satellite*:

# Landmarks extraídos del dominio Satellite				
Problema	Agentes	Estado inicial	Metas	Otros Landmarks
Pfile1	satellite0	27	5	8
	satellite1	27	5	0
Pfile2	satellite0	25	8	5
	satellite1	25	8	9
Pfile3	satellite0	57	7	5
	satellite1	57	7	6
	satellite2	57	7	4

Cuadro 5.2: Resultados de los problemas del dominio *Satellite*

Para ilustrar con más detalle los resultados obtenidos, las Figuras 5.4, 5.5 y 5.6 muestran los grafos de landmarks obtenidos en el problema *Pfile3* para los agentes `satellite0`, `satellite1` y `satellite2`, respectivamente. Este problema se ha modelado de forma que cada agente tiene un único instrumento a bordo con un solo modo de disparo: el `satellite0` puede hacer fotografías con el modo `thermograph0`, mientras que el `satellite1` puede hacerlas con el modo `image2` y el `satellite2` está capacitado para hacer fotografías con el modo `spectrograph0`. El problema incluye siete metas, seis de ellas consistentes en la obtención de fotografías: dos fotografías con el modo `thermograph0`, tres fotografías con el modo `image2` y una fotografía con el modo `spectrograph0`. Adicionalmente, se requiere que en el estado final el `satellite0` esté orientado hacia la dirección `phenomenon5`.

La Figura 5.4 muestra que el `satellite0` es capaz de conseguir las dos metas relacionadas con la toma de fotografías en modo `thermograph0`. Para obtener dichas imágenes, `(= (have-image star3 thermograph0) true)` y `(= (have-image star7 thermograph0) true)`, el satélite debe encender el instrumento `instrument0` y calibrarlo. Esto implica la inclusión de los siguientes órdenes necesarios:

- `(= (calibrated instrument0) true)` $\leq_n$ `(= (have-image star3 thermograph0) true)`
- `(= (power-on instrument0) true)` $\leq_n$ `(= (have-image star3 thermograph0) true)`

- `(= (calibrated instrument0) true)` $\leq_n$ `(= (have-image star7 thermograph0) true)`
- `(= (power-on instrument0) true)` $\leq_n$ `(= (have-image star7 thermograph0) true)`

Además, antes de realizar cualquiera de las dos fotografías, el `satellite0` debe orientarse hacia el fenómeno espacial a fotografiar, lo que queda reflejado en los siguientes órdenes necesarios:

- `(= (pointing instrument0) star3)` $\leq_n$ `(= (have-image star3 thermograph0) true)`
- `(= (pointing instrument0) star7)` $\leq_n$ `(= (have-image star7 thermograph0) true)`

Por último, para calibrar el instrumento `instrument0`, el `satellite0` debe encenderlo y orientarse hacia la dirección de calibración del mismo, lo que motiva la inclusión de los siguientes órdenes necesarios en el grafo de landmarks:

- `(= (power-on instrument0) true)` $\leq_n$ `(= (calibrated instrument0) true)`
- `(= (pointing instrument0) groundstation2)` $\leq_n$ `(= (calibrated instrument0) true)`

Respecto a la meta `(= (pointing instrument0) phenomenon5)`, el `satellite0` la consigue orientándose hacia fenómeno `phenomenon5`, lo cual no genera ningún orden necesario.

La Figura 5.5 muestra el grafo de landmarks del `satellite1`. Este agente puede resolver las tres metas relacionadas con la toma de fotografías en modo `image2`. Como se aprecia en la Figura 5.5, este agente obtiene órdenes necesarios similares a los del resto de satélites. Antes de realizar las fotografias, el `satellite1` debe encender su instrumento `instrument1` y calibrarlo, lo que genera los siguientes órdenes necesarios:

- `(= (calibrated instrument1) true)` $\leq_n$ `(= (have-image phenomenon5 image2) true)`
- `(= (power-on instrument1) true)` $\leq_n$ `(= (have-image phenomenon5 image2) true)`

- `(= (calibrated instrument1) true)` $\leq_n$ `(= (have-image phenomenon6 image2) true)`
- `(= (power-on instrument1) true)` $\leq_n$ `(= (have-image phenomenon6 image2) true)`
- `(= (calibrated instrument1) true)` $\leq_n$ `(= (have-image phenomenon8 image2) true)`
- `(= (power-on instrument1) true)` $\leq_n$ `(= (have-image phenomenon8 image2) true)`

Además, antes de realizar cualquiera de las tres fotografías, el satélite `satellite1` debe orientarse hacia al fenómeno espacial a fotografiar, lo que queda reflejado en los siguientes órdenes necesarios:

- `(= (pointing instrument1) phenomenon5)` $\leq_n$ `(= (have-image phenomenon5 image2) true)`
- `(= (pointing instrument1) phenomenon6)` $\leq_n$ `(= (have-image phenomenon6 image2) true)`
- `(= (pointing instrument1) phenomenon8)` $\leq_n$ `(= (have-image phenomenon8 image2) true)`

Antes de calibrar su instrumento, el satélite `satellite1` debe encenderlo y orientarse hacia la dirección de calibrado `groundstation1`. Esto implica la inclusión de los siguientes órdenes necesarios:

- `(= (power-on instrument1) true)` $\leq_n$ `(= (calibrated instrument1) true)`
- `(= (pointing instrument1) groundstation1)` $\leq_n$ `(= (calibrated instrument1) true)`

En la Figura 5.6 se observa que el `satellite2` puede resolver la meta relacionada con la toma de fotografías en modo `spectrograph0`. Como los demás satélites, antes de realizar una fotografía debe encender y calibrar su instrumental fotográfico (`instrument2`), dando lugar a los siguientes órdenes necesarios:

- `(= (calibrated instrument2) true)` $\leq_n$ `(= (have-image planet9 spectrograph0) true)`

- `(= (power-on instrument2) true)` $\leq_n$ `(= (have-image planet9 spectrograph0) true)`

A fin de realizar la fotografía, el `satellite2` debe orientarse en la dirección del fenómeno a fotografiar. En este caso, el `satellite2` apuntará a la dirección `planet9`, lo que implica la inclusión del siguiente orden necesario:

- `(= (pointing instrument2) planet9)` $\leq_n$ `(= (have-image planet9 spectrograph0) true)`

Por último, antes de calibrar su instrumento, el `satellite2` debe encender el instrumento `instrument2` y orientarse hacia la dirección de calibrado `groundstation0`, lo que queda reflejado en los siguientes órdenes necesarios:

- `(= (power-on instrument2) true)` $\leq_n$ `(= (calibrated instrument2) true)`
- `(= (pointing instrument2) groundstation0)` $\leq_n$ `(= (calibrated instrument2) true)`

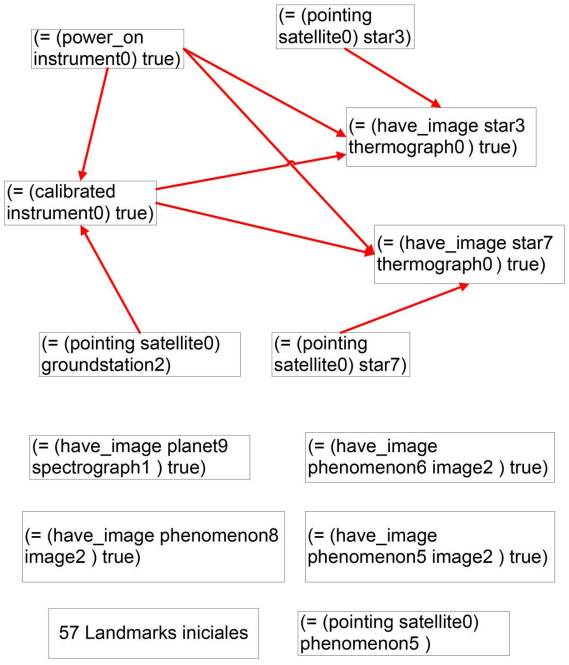

Figura 5.4: Estructura LG^0 del agente `satellite0`

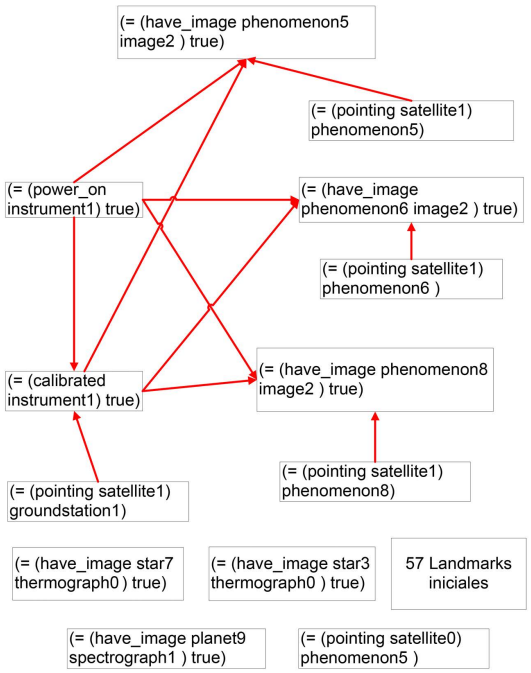

Figura 5.5: Estructura LG^1 del agente `satellite1`

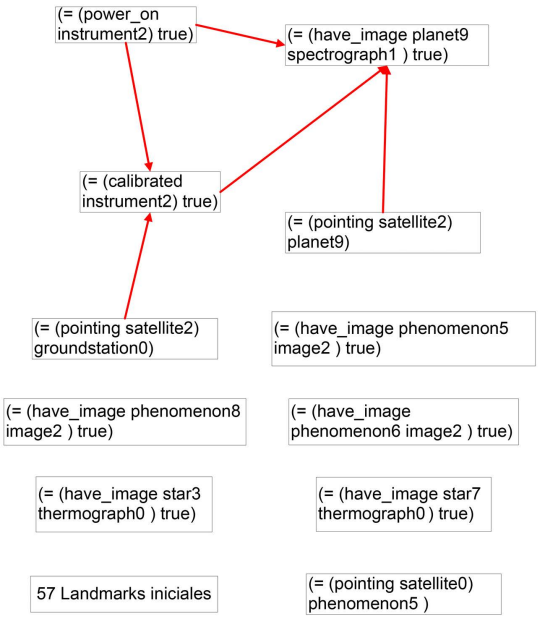

Figura 5.6: Estructura LG^2 del agente `satellite2`

Capítulo 6

Conclusiones y Trabajo Futuro

Este último capítulo resume el trabajo realizado en este Proyecto Final de Carrera y las futuras líneas de investigación a desarollar.

6.1. Resumen del trabajo realizado

El presente proyecto ha desarrollado el diseño e implementación multiagente del algoritmo de extracción de landmarks presentado en el artículo *Ordered Landmarks in Planning* [HPS04]. A partir del grafo de landmarks obtenido, se ha diseñado una función heurística con el objeto de mejorar el rendimiento del planificador FMAP, dando lugar a una nueva versión, FMAP-Land.

El algoritmo de extración de landmarks implementado resuelve las dificultades derivadas de la privacidad y coordinación entre agentes, permitiendo el cálculo de landmarks en entornos distribuidos.

Para facilitar la coordinación entre agentes, la implementación del algoritmo se ha integrado en el framework PlanInteraction[1]. Los recursos que ofrece esta plataforma han permitido gestionar adecuadamente las comunicaciones entre agentes necesarias para llevar a cabo el algoritmo planteado.

Los resultados experimentales muestran la eficacia del algoritmo presentado, que extrae, para cada problema, un número significativo de landmarks y órdenes necesarios, lo que proporciona información valiosa acerca de los planes solución.

[1] http://servergrps.dsic.upv.es/planinteraction/

6.2. Trabajo futuro

Hay dos líneas de trabajo futuro abiertas tras el presente Proyecto Final de Carrera:

- **Implementación de landmarks disyuntivos:** En la implementación actual se obvian los landmarks disyuntivos. Por ello, una de las líneas de trabajo futuro se centra en añadirlos a la implementación de FMAP-Land y así extraer todos los landmarks posibles de un problema.
- **Comparación entre FMAP y FMAP-Land:** Tras implementar el cálculo de landmarks disyuntivos, se llevará a cabo la implementación de la heurística de FMAP-Land y se desarrollará una amplia batería de pruebas. Estas pruebas se centrarán tanto en dominios de la Competición Internacional de Planning como en otros dominios modelados a partir de casos reales, con el objetivo de validar la nueva función heurística y comparar su rendimiento con la heurística original de FMAP.

Bibliografía

[BF97] A. Blum and M. L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1-2):281–300, 1997.

[BG01] B. Bonet and H. Geffner. Planning as heuristic search. *Artificial Intelligence*, 129:5–33, 2001.

[BN95] C. Bäckström and B. Nebel. Complexity results for SAS+ planning. *Computational Intelligence*, 11(4):625–655, 1995.

[BW94] A. Barrett and D. S. Weld. Partial-order planning: Evaluating possible efficiency gains. *Artificial Intelligence*, 67(1):71–112, 1994.

[EHN94] K. Erol, J. Hendler, and D. Nau. UCMP: A sound and complete procedure for hierarchical task-network planning. In *Proceedings of the International Conference on Artificial Intelligence Planning Systems*, pages 249–254, 1994.

[FN71] R. Fikes and N.J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3):189–208, 1971.

[Gef00] H. Geffner. Functional strips: a more flexible language for planning and problem solving. pages 187–209, 2000.

[GHK+98] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL - the planning domain definition language. *AIPS-98 Planning Committee*, 1998.

[GNT04] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning. Theory and Practice*. Morgan Kaufmann, 2004.

[HN01] J. Hoffmann and B. Nebel. The FF planning system: Fast planning generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[HPS04] Jörg Hoffmann, Julie Porteous, and Laura Sebastia. Ordered landmarks in planning. *J. Artif. Intell. Res. (JAIR)*, 22:215–278, 2004.

[KS96] H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic and stochastic search. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 1194–1201, 1996.

[PW92] J.S. Penberthy and D.S. Weld. UCPOP: A sound, complete, partial order planner for ADL. *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, pages 103–114, 1992.

[RW10] S. Richter and M. Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39(1):127–177, 2010.

[Sap05] O. Sapena. *Planificación Independiente del Dominio en Entornos Dinámicos de Tiempo Restringido*. PhD thesis, Universidad Politécnica de Valencia, 2005.

[SO03] Laura Sebastià and Eva Onaindía. Descomposición y resolución concurrente de problemas de planificación independiente del dominio. 2003.

[Wel94] D.S. Weld. An introduction to least commitment planning. *AI magazine*, 15(4):27, 1994.

[Wel99] D.S. Weld. Recent advances in AI planning. *AI Magazine*, 20(2):93–123, 1999.

[ZNK07] J.F. Zhang, X.T. Nguyen, and R. Kowalczyk. Graph-based multi-agent replanning algorithm. In *Proceedings of the 6th Conference on Autonomous Agents and Multiagent Systems*, 2007.