UNIVERSIDAD POLITÉCNICA DE VALENCIA

Escuela Técnica Superior de Ingeniería Informática

Ingeniería Técnica en Informática de Gestión

PROYECTO DE FIN DE CARRERA

Diseño e implementación de un sistema de planificación distribuido

Autor: Alejandro Torreño Lerma

Dirigido por:
Dr. Óscar Sapena Vercher

Grupo de Tecnología Informática - Inteligencia Artificial
Group of Reasoning on Planning and Scheduling (GRPS-AI)
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera, s/n
46022 Valencia, Spain

12 de Febrero de 2012

A todos los integrantes pasados y presentes del laboratorio 2L08 del DSIC.

A $C_6H_2(NO_2)_3CH_3$.

Índice

Agradecimientos

This work has been partly supported by the Spanish MICINN under projects TIN2011-27652-C03-01 and Consolider Ingenio 2010 CSD2007-00022, and the Valencian Prometeo project 2008/051.

Capítulo 1

Introducción

El término planificación se refiere al arte de construir algoritmos de control para la síntesis de cursos de acción que permitan obtener un conjunto deseado de metas a partir de una situación inicial. 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].

La Planificación Multi-Agente (en adelante PMA) generaliza el problema de planificación en dominios donde diversos agentes planifican y actúan conjuntamente. Cuando los agentes de planificación son completamente cooperativos, el ámbito de estudio se centra en cómo extender la planificación a un entorno distribuido. Tradicionalmente, la investigación en PMA se ha centrado en el diseño de arquitecturas de planificación distribuidas, mecanismos para la coordinación de planes y soluciones para combinar los planes locales de los diferentes agentes dando lugar a un plan global [Dur99, CDB05, dWtMW05]. A diferencia de estos modelos, que enfatizan el problema de controlar y coordinar soluciones locales de agentes independientes a posteriori, el presente proyecto propone un sistema de planificación distribuido que permite a los agentes participantes desarrollar un plan global conjuntamente. Nuestra propuesta se sustenta en el paradigma de Planificación de Orden Parcial (POP) [BW94].

Consideramos que PMA se refiere a la construcción de un curso de acción o plan entre un conjunto de agentes heterogéneos con diferentes capacidades y visiones del mundo. El sistema de planificación desarrollado esta orientado a la formación de un plan global a través de la composición de los planes individuales propuestos por los agentes participantes. Para ello, hemos adoptado un enfoque de planificación basada en refinamientos, por el cual los agentes proponen una serie de refinamientos sobre un plan base inicialmente vacío, hasta que se alcanza un plan solución. Este objetivo se consigue manteniendo

a su vez una visión distribuida del dominio de planificación, de modo que los agentes protegen su información privada.

De este modo, el presente proyecto introduce un nuevo modelo de PMA, así como la implementación y desarrollo de un sistema de planificación distribuido basado en dicho modelo. Por tanto, esta memoria proporciona una descripción de los componentes principales del sistema junto con detalles de implementación y resultados experimentales.

El presente documento se organiza como sigue: el capítulo 2 proporciona una descripción del estado del arte en los temas principales relacionados con este trabajo; el capítulo 3 resume el modelo teórico en el que se basa el sistema de planificación implementado; el capítulo 4 detalla el diseño del sistema de planificación distribuido, incluyendo una descripción de sus principales componentes y funcionalidades; el capítulo 5 documenta el desarrollo del sistema de planificación y muestra los resultados experimentales obtenidos; finalmente el capítulo 6 concluye y resume el trabajo futuro a desarrollar.

Capítulo 2

Antecedentes

El presente capítulo introduce el estado del arte de los principales tópicos relacionados con este trabajo: la sección 2.1 introduce el tópico de planificación clásica; la sección 2.2 resume el estado del arte de Planificación Multi-Agente (MAP); la sección 2.3 describe el paradigma de Planificación de Orden Parcial (POP), y la sección 2.4 detalla las características y la evolución de los lenguajes de descripción de dominios de planificación más relevantes.

2.1 Planificación clásica centralizada

El problema de planificación clásica se define como sigue [Wel99]: dada una descripción del estado inicial del mundo $\mathcal{I}$ (en algún lenguaje formal), una descripción de la meta del agente $\mathcal{G}$ (es decir, qué comportamiento es deseable), y una descripción de las posibles acciones (atómicas) que pueden llevarse a cabo $\mathcal{A}$, modelada a través de funciones de transformación de estados, se persigue 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. De este modo, 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 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 centrado en diferentes paradigmas de resolución de problemas. Los diferentes paradigmas de planificación pueden clasificarse de acuerdo a los siguientes conceptos [Sap05]:

- **Representación de los nodos:** Los planificadores pueden llevar a

cabo una búsqueda basada en estados, donde un nodo en el árbol de búsqueda representa una situación concreta, o una búsqueda basada en planes, donde los nodos representan planes.

- **Tipo de encadenamiento:** Un planificador hacia delante parte del estado inicial para resolver los objetivos del problema, mientras que los planificadores regresivos llevan a cabo la búsqueda en la dirección opuesta, lo que reduce la ramificación del árbol de búsqueda, dado que el proceso está dirigido por las metas.

- **Representación de los planes:** Los planificadores de orden total construyen soluciones parciales como secuencias totalmente ordenadas de acciones, mientras que los planificadores de orden parcial establecen restricciones de orden parcial entre las acciones de cada solución parcial.

Los primeros planificadores llevaban a cabo una búsqueda basada en estados explorando el espacio de los estados del mundo, de modo que las transiciones entre estados son provocadas por las acciones. Más tarde, la necesidad de manipular planes parciales 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. Otros enfoques aplican otro tipo de refinamientos, como Hierarchical Task Network (HTN) [EHN94], que reemplaza acciones abstractas por fragmentos de plan de bajo nivel.

La investigación en planificación a introducido otros enfoques que incrementan significativamente la eficiencia de los sistemas de planificación. Entre estos modelos, destacan GRAPHPLAN [BF97], SATPLAN [KS96] y planificación heurística [BG01]. GRAPHPLAN y SATPLAN trabajan construyendo distintas estructuras y buscando soluciones en ellas. En GRAPHPLAN, la estructura es un grafo, mientras que en SATPLAN es un conjunto de proposiciones. En planificación heurística, se emplea una función de utilidad o heurística para guiar la búsqueda en el espacio de estados. Este enfoque ha resultado muy exitoso, como ha demostrado el planificador FF [HN01].

En conclusión, la comunidad investigadora en planificación sigue trabajando activamente en la búsqueda de nuevas estrategias de búsqueda eficientes para planificación clásica centralizada, y particularmente en el diseño de técnicas independientes del dominio para mejorar la eficiencia de los sistemas automatizados de planificación. Aunque el modelo de planificación clásica centralizada ha guiado la mayor parte de la investigación en planificación, otros campos van ganando relevancia progresivamente, como la planificación en dominios estocásticos, dinámicos y parcialmente observables, que

no siguen las asunciones básicas de la planificación clásica. El objetivo es poder resolver problemas de planificación del mundo real, lo que hace necesario resolver problemas como el manejo de incertidumbre en el dominio, la monitorización y ejecución de los planes, la planificación distribuida y en entornos de tiempo real, etc.

2.2 Planificación Multi-Agente

El término Planificación Multi-Agente (PMA) designa el problema de planificación en entornos donde diversas entidades independientes (agentes) planifican y actúan conjuntamente. PMA puede referirse a la planificación *por* múltiples agentes (planificación distribuida) o a la planificación *para* múltiples agentes (ejecución multi-agente). PMA puede involucrar agentes que planifican para un objetivo común, un agente que coordina los planes de otros (fusión de planes), o agentes definiendo sus propios planes mientras negocian con otros acerca de tareas o recursos [Cle05]. Los agentes de planificación pueden colaborar para alcanzar objetivos comunes o actuar de forma egoísta para satisfacer sus propias metas privadas.

PMA puede verse como el problema de coordinar agentes en un entorno compartido donde la información está distribuida [dDOW99]. Esta definición subraya dos aspectos fundamentales de PMA que no están presentes en la planificación clásica: la coordinación de las actividades de planificación y la distribución de la información entre agentes.

En general, la resolución de una tarea de PMA conlleva las siguientes fases [Dur99]: 1) refinamiento de la meta global, 2) asignación de tareas, 3) coordinación antes de la planificación, 4) planificación individual, 5) coordinación después de la planificación, y 6) ejecución del plan. Algunas de estas fases pueden ignorarse o combinarse. Por ejemplo, algunos trabajos no distribuyen las metas explícitamente (evitando la fase 2) [BRR10, BN09], mientras que otros aplican únicamente coordinación después de la planificación (evitando de este modo la fase 3) [VDKDW05, CDB05].

Algunas aproximaciones de PMA se centran en resolver problemas en los que los agentes presentan escasas interacciones entre sí. Los agentes en estos modelos diseñan los planes independientemente, de modo que el objetivo es coordinar a posteriori estos planes individuales. Muchas de estas propuestas están enfocadas a la fusión de las soluciones locales en un plan global conjunto [Dur99, dWtMW05, TBdWW02, KPT02]. El trabajo en [VDKDW05] coordina los planes locales diseñados por agentes egoístas usando técnicas de reparación de planes. Otros trabajos tienen en cuenta las necesidades de comunicación que surgen en tiempo de ejecución de los planes [TNP10], o

coordinan las soluciones locales resolviendo un problema de satisfacción distribuida de restricciones [NBD10]. Por último, algunos trabajos combinan varias fases del proceso PMA en lugar de centrarse en la fusión de planes a posteriori. El trabajo en [ER96] combina planificación, coordinación y ejecución para tratar con agentes insinceros.

Otros sistemas de PMA están orientados a tratar con problemas que presentan fuertes interacciones entre los agentes de planificación. Para ello, resuelven el problema de PMA a través de un enfoque centralizado. Estos trabajos asumen que los agentes de planificación son completamente cooperativos, de modo que construyen bases de conocimiento completas y fuerzan a los agentes a comunicarse toda la información de la que disponen. Estos enfoques se centran en la construcción cooperativa e incremental de un plan conjunto, de forma que los agentes realizan su actividad de planificación sobre una representación de plan centralizad, combinando las actividades de planificación y coordinación. Algunas propuestas aplican el enfoque de planificación continua [dDOW99], que combina planificación y ejecución, coordinando los agentes en tiempo de ejecución [BN09]. La propuesta en [JR11] lleva a cabo un proceso iterativo de planificación basada en refinamientos, utilizando tecnología de planificación centralizada. [BRR10] propone un protocolo de diálogos iterados para coordinar los agentes. El popular algoritmo Partial Global Planning (PGP) [DL91] y su extensión, Generalized Partial Global Planning (GPGP) [DL92], combinan planificación y coordinación permitiendo a los agentes comunicar sus planes locales al resto de agentes, que fusionan esta información con sus planes globales parciales para mejorarlos.

Nuestro enfoque de PMA está relacionado con este último grupo de propuestas. Nuestro modelo permite resolver problemas de PMA en los que los agentes son heterogéneos y tienen diferente información y capacidades, así como metas privadas. Nuestro trabajo hace hincapié en la importancia de la privacidad en PMA, de modo que los agentes conservan su información privada y comparten únicamente la información clave que afecta también a otros agentes.

2.3 Planificación de Orden Parcial

En los planificadores de orden total, las acciones del plan se obtienen en el mismo orden en que se ejecutan. De este modo, si el planificador escoge una acción incorrecta, deberá introducir otra acción que deshaga los efectos de la primera acción. Al contrario que estos modelos, el paradigma de Planificación de Orden Parcial (*POP*) [BW94] introduce un enfoque más flexible, estableciendo relaciones de orden parcial entre las acciones en vez de forzar

un orden concreto entre ellas. Los planificadores basados en POP trabajan sobre todas las metas a la vez, sin comprometer un orden concreto entre las acciones hasta que lo determina la propia estructura del plan. Esta estrategia basada en retrasar la toma de decisiones durante el proceso de búsqueda se denomina *estrategia de menor compromiso* [Wel94].

En vez de llevara a cabo una búsqueda en el espacio de estados, los modelos basados en POP adoptan una búsqueda en el espacio de planes. De este modo, POP se basa en la construcción de un árbol de búsqueda en el que cada nodo representa un plan de orden parcial. POP se encuadra también en el conjunto de procesos de búsqueda regresivos, dado que el proceso comienza por satisfacer las metas del problema, y construye el plan hacia atrás. Por tanto, POP es un proceso de búsqueda regresivo y basado en planes.

Los planes parciales contienen un conjunto de *pasos* [Wel94], que constituyen las representaciones de las acciones de planificación dentro del plan parcial. Cada paso se compone de un conjunto de precondiciones y efectos, excepto los pasos ficticios inicial y final, que representan, respectivamente, el estado inicial y las metas del problema. Los pasos están parcialmente ordenados a través de un conjunto de *restricciones de orden*. Una restricción de orden parcial indica únicamente una relación de precedencia entre dos pasos, por lo que es posible que otros pasos queden ordenados entre los dos pasos a los que hace referencia la restricción de orden.

Una *meta abierta* es una precondición que no está soportada todavía por un *enlace causal*. Un enlace causal indica que un paso soporta una precondición de otro paso, dado que tiene un efecto que coincide con dicha precondición. La introducción de enlaces causales en el plan conlleva la aparición de *amenazas*. Una amenaza se produce cuando hay un paso que puede ordenarse entre los dos pasos del enlace causal, y que tiene un efecto complementario a la precondición soportada en el enlace causal. El proceso de búsqueda POP está orientado a la progresiva resolución de las amenazas y metas abiertas del plan a través de la adición de enlaces causales y restricciones de orden.

El algoritmo 1 resume el proceso POP. El proceso se inicia con un plan vacío inicial, que sólo incluye los pasos ficticios $\mathcal{I}$ y $\mathcal{F}$. Este plan se almacena en la lista de $Nodos\_abiertos$ que almacena los nodos hoja del árbol de búsqueda.

En cada iteración, el algoritmo de búsqueda extrae un plan de la lista de$Nodos_abiertos$, y resuelve una de las inconsistencias del plan (metas abiertas y amenazas). La resolución de la inconsistencia conlleva la generación de un conjunto de planes sucesores del plan actual, tantos como formas haya de resolver la inconsistencia. Los planes sucesores son almacenados en la lista de $Nodos\_abiertos$.

Una vez se extrae un plan sin inconsistencias, es devuelto como solución, y

Algoritmo 1 Algoritmo POP

```
Nodos_abiertos ← {Plan vacío}
repetir
  Seleccionar Π ∈ Nodos_abiertos
  Inconsistencias_pendientes ← metas_abiertas(Π) ∪ amenazas(Π)
  si Inconsistencias_pendientes = ∅ entonces
    devolver Π
  fin si
  Seleccionar y extraer Φ ∈ Inconsistencias_pendientes
  Sucesores ← {Π_r}, ∀ Π_r que resuelve Φ
  si Sucesores ≠ ∅ entonces
    Nodos_abiertos ← Nodos_abiertos ∪ Sucesores
  fin si
hasta que Nodos_abiertos = ∅
devolver fallo
```

el proceso POP termina con éxito. En caso de que la lista de *Nodos_abiertos* quede vacía antes de encontrar una solución, habremos explorado completamente el espacio de búsqueda sin encontrar solución, por lo que el proceso terminará sin éxito.

La investigación en planificación se centró en POP durante la década de los '90. Sin embargo, en los últimos años, POP ha sido abandonado progresivamente en favor de otros paradigmas más eficientes y escalables. La dificultad de diseñar heurísticas eficientes para POP condiciona el rendimiento de los planificadores basados en este paradigma. Ni siquiera los trabajos más recientes orientados a mejorar el rendimiento del paradigma POP [NK01] resultan suficientemente eficientes para competir con los enfoques de planificación basada en estados. Sin embargo, en los últimos años POP ha ganado relevancia, dado que su flexibilidad lo convierte en una buena alternativa para el desarrollo de planificadores multi-agente.

2.4 Lenguajes de planificación

Uno de los principales problemas a los que se ha enfrentado la comunidad de planificación es el problema de la representación. El uso de un buen lenguaje de planificación es básico para el desarrollo de herramientas de planificación. Desde los años '70, la mayoría de propuestas de planificación han sido influenciadas por el lenguaje *STRIPS* [FN71], que resuelve de forma efectiva el problema marco [MH69], y da soporte a estrategias de tipo divide y vencerás

[Gef00]. Esta sección describe brevemente las características de este lenguaje y sus extensiones más relevantes: *ADL* [Ped89] y *PDDL* [McD00].

2.4.1 STRIPS

El lenguaje *STRIPS* (STanford Research Institute Problem Solver [FN71]) fue desarrollado a principios de los '70, como parte del sistema de planificación para el robot *Shakey*. *STRIPS* propone un modelo simple y compacto para la especificación de dominios de planificación.

La representación propuesta por *STRIPS* incluye diversas limitaciones que complican la descripción de problemas reales [RN03], por lo que a lo largo de los años se han introducido un conjunto de extensiones que enriquecen la expresividad del lenguaje y simplifican la definición de dominios de planificación.

Las limitaciones del lenguaje incluyen la posibilidad de utilizar únicamente literales positivos para describir las precondiciones de las acciones y el estado inicial, de modo que la información no mencionada se considera falsa (negación por fallo). Asimismo, las precondiciones, efectos y metas sólo se pueden describir mediante conjunción de literales, por lo que no pueden utilizarse disyunciones. Por último, destaca la ausencia de soporte de tipos y restricciones de igualdad de tipo ($a = b$).

Además de corregir estas limitaciones, las extensiones de *STRIPS* mejoran el lenguaje introduciendo nuevas características, como gestión del tiempo y expresiones numéricas.

2.4.2 ADL

Una de las extensiones más populares de *STRIPS* es *ADL* (*Action Description Language*. *ADL* usa un modelo algebraico para describir los estados del mundo, lo que lo hace más expresivo que *STRIPS*.

Las mejoras que introduce *ADL* sobre *STRIPS* incluyen la incorporación de tipos en los objetos del problema, la introducción de metas y precondiciones negadas, precondiciones disyuntivas y restricciones de igualdad. Del mismo modo, se introducen los efectos condicionales, que sólo son efectivos si se cumple una determinada condición en el estado en el que se aplica la acción.

Las extensiones introducidas por *ADL* mejoran notablemente la expresividad de *STRIPS*. Estas nuevas características permiten también mejorar la eficiencia de los sistemas de planificación [KNHD97].

2.4.3 PDDL

Además de *ADL* se han desarrollado otras muchas extensiones de *STRIPS*, como *FStrips* (*Functional STRIPS* [Gef00]. Sin embargo, la extensión de *STRIPS* de mayor repercusión es *PDDL* (Planning Domain Definition Language) [GHK+98]) *PDDL* fue desarrollado para la Competición Internacional de Planning de 1998 [McD00], con el objetivo de proporcionar una notación común para el modelado de problemas de planificación. Desde su introducción, *PDDL* se ha convertido en el lenguaje de referencia para la mayoría de sistemas de planificación.

Además de *STRIPS* y *ADL*, *PDDL* ha recibido la influencia de otros muchos formalismos: *SIPE-2* [Wil88], *Prodigy 4.0* [BEG+92], *UCMP* [EHN94], *Unpop* [McD96] y *UCPOP* [BCF+95]. Las características más relevantes de *PDDL* incluyen la definición de efectos condicionales, cuantificación universal, acciones jerárquicas, axiomas de dominio, y restricciones de seguridad.

La mayor parte de planificadores manejan únicamente un subconjunto de las características de *PDDL*. Por simplicidad, las funcionalidades de *PDDL* se agrupan en conjuntos de requerimientos, de modo que los planificadores pueden comprobar fácilmente si soportan las características de un determinado dominio de planificación.

2.4.4 Extensiones de PDDL

La Competición Internacional de Planning (*IPC*) [DKS+00] se ha convertido en una importante referencia para la investigación en planificación. Uno de los resultados más importantes de su primera edición fue la adopción de *PDDL* como el lenguaje común de definición de dominios de planificación [MGH+98]. Las siguientes ediciones de la competición sirvieron para introducir nuevas extensiones del lenguaje. Esta sección presenta las características principales de dichas extensiones.

La primera revisión de *PDDL* se introdujo en la *IPC* de 2002 (*IPC-3*) [FL03]. Esta extensión añade capacidades numéricas y de manejo de tiempo a *PDDL*.

La cuarta *IPC* (*IPC-4*), celebrada en 2004, introdujo una nueva revisión de *PDDL*, *PDDL2.2* [Ede03]. Esta extensión introduce cambios relativamente moderados, entre los que destacan los axiomas derivados (reglas de la forma $sif(x)entoncesP(x)$) y literales que pasan a ser verdaderos o falsos en instantes de tiempo establecidos.

PDDL3.0 [GL05] se desarrolló para la *IPC* de 2006 (*IPC-6*). *PDDL3.0* enfatiza la importancia de la calidad del plan, a diferencia de las extensiones anteriores. Para ello, se introducen características como las restricciones de

trayectoria de estado, las restricciones débiles y las preferencias.

Finalmente, *PDDL3.1* [Kov11], la extensión más reciente de *PDDL* se introdujo en la *IPC* de 2008. El objetivo de esta extensión era 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 que toman 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 los costes de las acciones adquieren importancia para determinar la calidad del plan.

Capítulo 3

Modelo de Planificación Multi-Agente

Esta sección presenta el modelo de Planificación Multi-Agente (PMA) en el que se basa el sistema de planificación distribuido implementado. Del mismo modo, se describe el procedimiento seguido por los agentes para construir e intercambiar planes.

3.1 Especificación de una tarea de PMA

Definición 1. *(**Tarea de PMA**) Una **tarea de PMA** 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 PMA* $\mathcal{T}$*.* $\mathcal{G}$ *es el conjunto de metas de la tarea de PMA 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, nuestro modelo permite la representación explícita de la información verdadera y falsa. Por tanto, nuestro 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. Esto se refiere a la información relativa al estado inicial $\mathcal{I}$ y a las metas $\mathcal{G}$.

Definición 2. *(**Variable instanciada**) Una **variable instanciada** del problema es una tupla de la forma$\langle v, d\rangle$, donde $v \in \mathcal{V}$ y $d \in \mathcal{D}_v$. 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.*

Los agentes en nuestro modelo son heterogéneos, dado que pueden tener diferentes conocimientos y habilidades de planificación. Además, pueden tener información incompleta acerca de la tarea de PMA, dado que ésta está distribuida entre los agentes. En dicho caso, los agentes deben cooperar para resolver la tarea de PMA. 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 $i \in \mathcal{AG}$ usaremos la notación de superíndice x^i para cada aspecto x.

Del conjunto de variables $\mathcal{V}$ de la tarea de PMA, $\mathcal{V}^i$ es el conjunto de variables gestionadas por el agente i, lo que incluye las variables privada 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 PMA, $\mathcal{I}$,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 PMA 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 PMA 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 una 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 del plan final, 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 $\langle v, d\rangle$ from 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, y en consecuencia, el resto de los valores de $\mathcal{D}_v$, a excepción de d, son valores desconocidos.

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, $(\exists\langle v, d\rangle \in S) \Rightarrow (\forall d' \in \mathcal{D}_v, d' \neq d, \exists\langle v, \neg d'\rangle \in S)$.

Cada agente i posee una función de utilidad $\mathcal{F}^i$ para evaluar la calidad de los planes propuestos. $\mathcal{F}^i$ asigna un coste $coste^i(\alpha) \in \mathbb{R}_0^+$ a cada acción α de un plan de acuerdo con la visión de la tarea de PMA que tiene el agente i. Por último, las metas privadas de un agente i, $\mathcal{PG}^i$, son variables instanciadas que el agente i está interesado en conseguir. Las metas privadas se codifican como restricciones débiles [GL06], dado que no es obligatorio que los agentes las consigan.

3.2 Planificación basada en refinamientos

Nuestro modelo de PMA es un enfoque de planificación basada en refinamientos, un método consistente en el refinamiento del conjunto de posibles planes [Kam97]. Un agente propone un plan Π que típicamente resuelve un conjunto de metas abiertas; a continuación, el resto de agentes cooperan para refinar Π, resolviendo algunas de sus metas abiertas. De este modo, los agentes resuelven cooperativamente la tarea de PMA mediante consecutivos refinamientos de un plan inicialmente vacío.

En este contexto, la Planificación de Orden Parcial (POP) [BW94] aparece como un enfoque adecuado para plantear la planificación basada en refinamientos, dado que este paradigma está orientado a resolver las metas abiertas de forma progresiva. De este modo, los agentes en nuestro modelo planifican mediante la adopción del paradigma POP. A continuación, se proporcionan las definiciones básicas de POP y su adaptación al contexto de PMA.

3.2.1 Planificación de Orden Parcial centralizada

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 PMA $\mathcal{T}$ contendrá siempre las dos acciones ficticias, de modo 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 PMA $\mathcal{T}$, 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. El proceso de búsqueda POP se centra en introducir enlaces causales para soportar estas precondiciones, también llamadas *metas abiertas*.

Definición 5. *(**Meta abierta**) Una **meta abierta** en un plan parcial* $\Pi = \langle \Delta, \mathcal{OR}, \mathcal{CL} \rangle$ *es una variable instanciada og de la forma* $\langle v, d \rangle$ *o bien* $\langle v, \neg d \rangle$*, tal que* $v \in \mathcal{V}$*,* $d \in \mathcal{D}_v$*,* $og \in PRE(\beta)$*,* $\beta \in \Delta$*, y* $\nexists \alpha \in \Delta / \alpha \overset{og}{\rightarrow} \beta \in \mathcal{CL}$*.* $metasAbiertas(\Pi)$ *denota el conjunto de metas abiertas en* Π*. Un plan es **incompleto** si tiene metas abiertas. En caso contrario, se trata de un plan **completo**.*

A medida que el proceso de búsqueda POP progresa, los enlaces causales en un plan de orden parcial pueden quedar desprotegidos como resultado de la introducción de una nueva acción que no está ordenada con respecto al enlace causal. Estos conflictos son conocidos como *amenazas*.

Definición 6. *(**Amenazas**) Una **amenaza** en un plan parcial* $\Pi = \langle \Delta, \mathcal{OR}, \mathcal{CL} \rangle$ *representa un conflicto entre una acción del plan y un enlace causal. Una acción* γ *causa una amenaza sobre un enlace causal* $\alpha \overset{\langle v,d \rangle}{\rightarrow} \beta$ *si* $((v = d') \in EFF(\gamma) \vee (v \neq d) \in EFF(\gamma))$*, donde* $v \in \mathcal{V}$*,* $d \in \mathcal{D}_v$*,* $d' \in \mathcal{D}_v$ *y* $d \neq d'$*, y no existe una restricción de orden* $\gamma \prec \alpha$ *o* $\beta \prec \gamma$*. La acción* γ *causará una amenaza sobre un enlace causal de la forma* $\alpha \overset{\langle v,\neg d \rangle}{\rightarrow} \beta$ *si* $(v = d) \in EFF(\gamma)$*, donde* $v \in \mathcal{V}$*,* $d \in \mathcal{D}_v$*, y no existe una restricción de orden* $\gamma \prec \alpha$ *o* $\beta \prec \gamma$*.* $Threats(\Pi)$ *denota el conjunto de amenazas en* Π*.*

Una amenaza $t \in Threats(\Pi)$ se puede resolver *promocionando* o *democionando* la acción amenazante γ con respecto al enlace causal amenazado $\alpha \overset{\langle v,d \rangle}{\rightarrow} \beta$ o $\alpha \overset{\langle v,\neg d \rangle}{\rightarrow} \beta$, esto es, introduciendo una restricción de orden $\gamma \prec \alpha$ o $\beta \prec \gamma$.

3.2.2 Planificación de Orden Parcial distribuida

Los agentes en nuestro modelo de PMA cooperan para refinar un plan base Π inicialmente vacío proponiendo una serie de *pasos de refinamiento* que resuelven algunas de las metas abiertas en Π.

Definición 7. *(**Paso de refinamiento**) Un paso de refinamiento* Π^i *desarrollado por un agente i sobre un plan base* Π_g*, donde* $g \in metasAbiertas(\Pi_g)$*, es una tripla* $\Pi^i = \langle \Delta^i, OR^i, CL^i \rangle$*, donde* $\Delta^i \subseteq \mathcal{A}$ *es un conjunto de acciones*

y OR^i y CL^i son conjuntos de restricciones de orden y enlaces causales sobre Δ^i, respectivamente. Π^i es un plan parcial libre de amenazas que resuelve g, así como todas las nuevas metas abiertas de la forma $\langle v, d\rangle$ o bien $\langle v, \neg d\rangle$ que surgen de esta resolución y sólo pueden ser resueltas por el agente i, donde $(v \in \mathcal{V}^i) \wedge (v \notin \mathcal{V}^j, \forall j \neq i)$. Es decir, además de resolver una meta abierta del plan base, el agente resolverá también las nuevas metas abiertas relacionadas con las variables instanciadas privadas del agente, dejando las metas abiertas públicas sin resolver. Dicho de otro modo, el proceso de refinamiento itera únicamente sobre variables instanciadas públicas. Sea $g \in metasAbiertas(\Pi_g)$ una variable instanciada de la forma $\langle v, d\rangle$ o bien $\langle v, \neg d\rangle$; un agente i propondrá un paso de refinamiento sobre Π_g si y sólo si $v \in \mathcal{V}^i$.

En nuestro modelo de PMA los planes parciales son planes multi-agente concurrentes, dado que dos o más acciones pueden ser ejecutadas por distintos agentes concurrentemente. Algunos modelos de PMA adoptan una forma simple de concurrencia: dos acciones concurrentes son mutuamente consistentes si ninguna de ellas cambia el valor de una variable de estado que figura en las precondiciones o efectos de la otra acción [BN09]. Nosotros imponemos una restricción de concurrencia adicional: las precondiciones de dos acciones deben ser consistentes [BB01] para que estas dos acciones sean mutuamente consistentes. Esta definición de concurrencia puede extenderse directamente a una acción conjunta $\alpha = \langle \alpha_1, \ldots, \alpha_n\rangle$.

Definición 8. *(**Acciones mutuamente consistentes**) Dos acciones concurrentes $\alpha \in \mathcal{A}^i$ y $\beta \in \mathcal{A}^j$ son **mutuamente consistentes** si no se cumple ninguna de las siguientes condiciones:*

- $\exists(v = d) \in EFF(\alpha)$ *y* $\exists(\langle v, d'\rangle \in PRE(\beta) \vee \langle v, \neg d\rangle \in PRE(\beta))$, *donde* $v \in \mathcal{V}^i \cap \mathcal{V}^j$, $d \in \mathcal{D}^i_v \cap \mathcal{D}^j_v$, $d' \in \mathcal{D}^j_v$ *y* $d \neq d'$, *o viceversa.*

- $\exists(v = d) \in EFF(\alpha)$ *y* $\exists((v = d') \in EFF(\beta) \vee (v \neq d) \in EFF(\beta))$, *donde* $v \in \mathcal{V}^i \cap \mathcal{V}^j$, $d \in \mathcal{D}^i_v \cap \mathcal{D}^j_v$, $d' \in \mathcal{D}^j_v$ *y* $d \neq d'$, *o viceversa.*

- $\exists\langle v, d\rangle \in PRE(\alpha)$ *y* $\exists(\langle v, d'\rangle \in PRE(\beta) \vee \langle v, \neg d\rangle \in PRE(\beta))$, *donde* $v \in \mathcal{V}^i \cap \mathcal{V}^j$, $d \in \mathcal{D}^i_v \cap \mathcal{D}^j_v$, $d' \in \mathcal{D}^j_v$ *y* $d \neq d'$, *o viceversa.*

Dado que los agentes corrigen las inconsistencias de concurrencia a través de la detección de amenazas sobre los enlaces causales de sus planes, la concurrencia está asegurada entre las acciones privadas, dado que los planes refinamiento están siempre libres de amenazas. Sin embargo, los problemas de concurrencia entre dos acciones públicas pueden no surgir hasta que

todas sus precondiciones están completamente soportadas mediante enlaces causales. De este modo, no es posible asegurar que dos acciones concurrentes son mutuamente consistentes hasta que sus precondiciones estén completamente soportadas. Por tanto, nuestra noción de plan multi-agente concurrente distingue entre acciones privadas y públicas al tratar la concurrencia.

Definición 9. *(**Plan concurrente multi-agente**) Un plan de orden parcial* $\Pi = \langle \Delta, \mathcal{OR}, \mathcal{CL} \rangle$ *es un* ***plan concurrente multi-agente*** *si para cada par de acciones públicas concurrentes* α *y* β*,* $\alpha \neq \beta$*,* $\forall p_\alpha \in PRE(\alpha), p_\alpha \notin metasAbiertas(\Pi)$*,* $\forall p_\beta \in PRE(\beta), p_\beta \notin metasAbiertas(\Pi)$*,* α *y* β *son mutuamente consistentes.*

Definición 10. *(**Plan refinamiento**) Un plan refinamiento* Π *desarrollado por un agente* i *sobre un plan base* Π_g *es un plan concurrente multi-agente que resulta de la composición de* Π_g *y un paso de refinamiento* Π^i *propuesto por el agente* i*.* Π *se define como* $\Pi = \Pi_g \circ \Pi^i$*, donde* $\circ$ *representa la operación de composición.*

Por tanto, un agente i puede construir un plan refinamiento Π sobre un plan base Π_g componiendo Π_g con un paso de refinamiento Π^i que resuelve al menos $g \in metasAbiertas(\Pi_g)$, es decir, $\Pi = \Pi_g \circ \Pi^i$. Como se ha indicado anteriormente, los pasos de refinamiento están siempre libres de amenazas y sus acciones son mutuamente consistentes. Por tanto, su un paso de refinamiento provoca alguna inconsistencia de concurrencia o amenaza en el plan refinamiento resultante, el agente que ha realizado la propuesta es el encargado de corregir la inconsistencia. Si un agente no es capaz de dar con un plan refinamiento consistente, no realizará ninguna propuesta. En caso de que no se encuentren refinamientos para un plan base, este se considera un plan sin salida.

Definición 11. *(**Plan sin salida**) Un plan* Π *recibe el nombre de* ***plan sin salida*** *si* $\exists g \in metasAbiertas(\Pi)$ *y no hay ningún paso de refinamiento* Π^i *tal que* $g \notin metasAbiertas(\Pi \circ \Pi^i)$*; esto es, no hay ningún paso de refinamiento que resuelva la meta abierta* g*.*

Definición 12. *(**Plan solución**) Un plan concurrente multi-agente* Π *es un* ***plan solución*** *para una tarea de PMA* $\mathcal{T}$ *if* $metasAbiertas(\Pi) = \emptyset$ *(*Π *es un plan completo),* $Threats(\Pi) = \emptyset$*, y cada par de acciones* $\alpha, \beta \in \Pi$ *son mutuamente consistentes.*

Por tanto, un plan solución es un plan concurrente multi-agente completo. Nótese que requerimos que Π sea un plan completo, de modo que no puede tener metas abiertas pendientes. En consecuencia, las precondiciones de la acción ficticia final α_f estarán también resueltas, lo que garantiza que Π resuelve la tarea de PMA $\mathcal{T}$.

Capítulo 4

Diseño del sistema de planificación distribuido

El modelo teórico mostrado en la sección anterior se ha empleado como base para el diseño del planificador distribuido. El presente capítulo lleva a cabo una descripción detallada del diseño del sistema, analizando sus principales componentes, que pueden resumirse como sigue (ver figura 4.1):

- **Ficheros de entrada:** Los agentes de planificación reciben un conjunto de ficheros de descripción, que modelan la información de la tarea de PMA. Dado que PMA introduce una serie de requerimientos que no están presentes en la planificación clásica centralizada, se ha diseñado un lenguaje de PMA para este fin.

- **Sistema multi-agente:** Este componente permite a los agentes comunicarse e interactuar con el resto, facilitando el intercambio de planes y la toma de decisiones respecto a la adopción del siguiente plan base.

- **Planificador de Orden Parcial (POP):** El POP es el elemento clave de nuestro sistema de planificación. Cada uno de los agentes tiene un POP embebido, lo que les permite generar un conjunto de planes refinamiento una vez reciben un plan base.

El presente capítulo se estructura como sigue: la sección 4.1 presenta el lenguaje diseñado para describir dominios de PMA; la sección 4.2 describe el algoritmo de PMA que los agentes llevan a cabo para resolver las tareas de PMA, y la sección 4.3 presenta las extensiones realizadas sobre el algoritmo clásico POP para adaptarlo a un contexto de PMA.

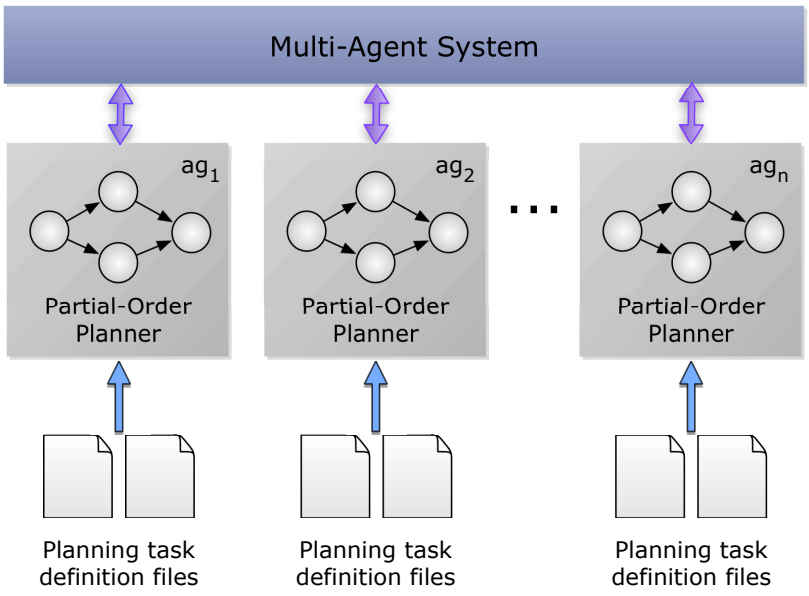

Figura 4.1: Estructura del sistema de PMA diseñado

4.1 Lenguaje de planificación

Nuestro planificador distribuido configura la tarea de PMA de cada agente mediante un conjunto de ficheros de entrada. Estos ficheros codifican tanto las variables $\mathcal{V}^i$, los objetos asociados a dichas variables $\mathcal{O}^i$, las acciones de planificación $\mathcal{A}^i$, y el estado inicial del agente $\mathcal{I}^i$. Toda esta información se codifica mediante un lenguaje de planificación.

Nuestro lenguaje de planificación se basa en *PDDL3.1* [Kov11], la versión más reciente de *PDDL* [GHK+98]. *PDDL3.1* permite modelar el dominio de planificación mediante variables de estado, introduciendo el concepto de variable objeto (ver sección 2.4). Hemos extendido el lenguaje *PDDL3.1* para soportar los requisitos adicionales de PMA que no están soportados en el lenguaje original, como la definición de información compartida y de metas globales y locales.

En planificación centralizada, la tarea de planificación se describe a través de dos ficheros, el *dominio* de planificación (que describe los tipos de objetos, variables de estado y acciones de la tarea) y el *problema* concreto a resolver (que incluye los objetos concretos de la tarea y la descripción del estado inicial y las metas). En nuestro caso, cada agente participante contará con sus propios ficheros de dominio y problema.

Para soportar los requisitos adicionales de las tareas de PMA, se han introducido las siguientes estructuras: `shared-data` indica qué variables in-

stanciadas puede compartir el agente y con qué agentes. `global-goal` y `private-goal` modelan las metas globales de la tarea y los objetivos privados del agente. Finalmente, se ha incluido una estructura `multi-functions` para simplificar la especificación de variables instanciadas en el estado inicial del agente. A continuación, analizamos la sintaxis y funcionalidad de las nuevas estructuras añadidas al lenguaje de planificación.

4.1.1 Información compartida

La sección `shared-data`, situada en el fichero de problema de cada agente, determina qué información puede compartirse con otros agentes. La información a compartir se modela mediante variables instanciadas o predicados. La estructura definida sigue la siguiente sintaxis BNF:

```
<shared-data-def>  ::= (:shared-data <share-def>+)
<share-def>        ::= (<atom-formula-def>+ [- <agent>])
<agent>            ::= <name>
<atom-formula-def> ::= (<predicate> <typed-list(element)>)
<atom-formula-def> ::= (= <object-fluent> <object>)
<predicate>        ::= <name>
<object-fluent>    ::= (<name> <object>*)
<object>           ::= <name>
<element>          ::= <variable> | <constant>
<variable>         ::= ?<name>
<constant>         ::= <name>
<typed-list(x)>    ::= x*
```

Como se observa en la sintaxis BNF, es posible definir conjuntos de predicados o variables instanciadas y asociarlos a uno o más agentes del sistema (si no se especifica un `agent`, la información se comparte con todos los agentes participantes).

4.1.2 Metas privadas y globales

Una particularidad de las tareas de PMA reside en el hecho de que cada agente puede tener metas privadas, además de las metas globales de la tarea. Esta información se refleja a través de las estructuras `private-goal` y `global-goal`, ubicadas en los ficheros de problema. De modo similar a la sección `:goal` en *PDDL3.1*, las metas pueden modelarse mediante variables instanciadas o predicados. Las estructuras siguen la siguiente sintaxis BNF:

```
<private-goal-def>  ::= (:private-goal <predicates-def>)
<global-goal-def>   ::= (:global-goal <predicates-def>)
<predicates-def>    ::= <atom-formula-def>
<predicates-def>    ::= (and <atom-form-def> <atom-form-def>+)
```

```
<predicates-def>     ::= (or <atom-form-def> <atom-form-def>+)
<atom-form-def>      ::= (<predicate> <typed-list(element)>)
<atom-form-def>      ::= (= <object-fluent-def> <object>)
<predicate>          ::= <name>
<object-fluent-def>  ::= (<name> <object>*)
<object>             ::= <name>
<element>            ::= <variable> | <constant>
<variable>           ::= ?<name>
<constant>           ::= <name>
<typed-list(x)>      ::= x*
```

Como muestra la sintaxis BNF, ambos conjuntos de metas globales y privadas pueden describirse mediante conjunción o disyunción de predicados o variables instanciadas.

4.1.3 Multi-funciones

Como se ha visto en la sección 3, nuestro modelo permite la representación explícita de información postiva y negativa, lo que dificulta el proceso de codificación, dao que se incrementa el volumen de datos a codificar. Para simplificar este proceso, se ha añadido una estructura `multi-functions` que permite codificar parte de la información del problema mediante una notación más compacta y simplificada.

Las multi-funciones se declaran en el fichero de dominio, dentro de la sección `:multi-functions`. Dicha sección sigue la sintaxis BNF que se muestra a continuación:

```
<multi-func-sec-def> ::= (:multi-functions <multi-func-def>)
<multi-func-def>     ::= (<name> <typed-list(typed-v)>) <type>
<typed-v>            ::= <variable> - <type>
<type>               ::= <name>
<type>               ::= (either <name> <name>+)
<variable>           ::= ?<name>
<typed-list(x)>      ::= x*
```

En la sección `:init` del fichero de problema se emplean las multi-funciones para definir información concreta del estado inicial. La especificación de multi-funciones en la sección `:init` utiliza la siguiente sintaxis BNF:

```
<multi-func-def> ::= (= <multi-func> <value-def>)
<multi-func>     ::= (<name> <typed-list(inst-v)>*) <inst-v>+
<inst-v>         ::= <name>
<variable>       ::= ?<name>
<typed-list(x)>  ::= x*
```

4.2 Algoritmo de planificación distribuido

Esta sección detalla el algoritmo de planificación cooperativa basada en refinamientos que se ha diseñado. Los agentes siguen un protocolo que combina planificación y coordinación. Uno de los agentes lidera el proceso de selección del siguiente plan base a partir de los planes refinamiento generados por el grupo de agentes. El algoritmo de planificación se divide en las siguientes fases:

- **Intercambio inicial de información:** En esta fase inicial, los agentes intercambian información de planificación para generar estructuras de datos que serán de utilidad en las siguientes fases del proceso.
- **Proceso de planificación basada en refinamientos:** Este proceso combinan dos fases, una fase de planificación individual en la cual los agentes refinan un plan base centralizado, y un proceso de coordinación en el que los agentes intercambian sus propuestas y escogen el siguiente plan base:
  - **Proceso de refinamiento individual:** Cada agente tiene un planificador POP embebido con el que refinan de forma individual el plan base actual. El algoritmo POP clásico se ha adaptado al contexto de PMA, tal como se describe en la sección 3.2.2.
  - **Proceso de coordinación:** Los agentes intercambian los nuevos planes refinamiento desarrollados sobre el plan base actual y seleccionan el refinamiento más prometedor como el nuevo plan base.

4.2.1 Intercambio inicial de información

Antes de comenzar el proceso de planificación, los agentes llevan a cabo una fase preliminar para intercambiar la información pública de planificación. Esta fase inicial se centra en la construcción de un grafo de planificación relajado distribuido (dis-RPG) basado en el trabajo de [ZNK07]. El dis-RPG proporciona a los agentes información de planificación valiosa para el resto del proceso:

- Los agentes intercambian la información definida como compartible en la sección `:shared-data` de los ficheros de definición. Cada variable instanciada se etiqueta con la lista de agentes que pueden conseguirla, lo que proporciona a cada agente una visión de las posibles interacciones que pueden surgir con el resto de agentes en tiempo de planificación.

- Se calcula una estimación del mejor coste para conseguir cada variable instanciada. Esta información es útil para definir heurísticas que guíen el proceso de planificación.

Algoritmo 2 Construcción del dis-RPG para un agente i

Construir RPG^i inicial
repetir
  $\forall j \neq i$, i envía a j las nuevas variables instanciadas $SF^{i \to j} \in RPG^i$ de la forma $\langle v, d \rangle$ o bien $\langle v, \neg d \rangle$, donde $v \in \mathcal{V}^i \cap \mathcal{V}^j$ y $d \in \mathcal{D}_v^i \cap \mathcal{D}_v^j$
  $\forall j \neq i$, i recibe de j las nuevas variables instanciadas $SF^{j \to i} \in RPG^j$ de la forma $\langle v, d \rangle$ o bien $\langle v, \neg d \rangle$, donde $v \in \mathcal{V}^i \cap \mathcal{V}^j$ y $d \in \mathcal{D}_v^i \cap \mathcal{D}_v^j$
  $RF^i \leftarrow \emptyset$
  $\forall j \neq i, RF^i \leftarrow RF^i \cup SF^{j \to i}$
  para todo variable instanciada recibida $f \in RF^i$ **hacer**
    si $f \notin RPG^i$ **entonces**
      Insertar f en RPG^i
      $coste_{RPG^i}(f) \leftarrow coste(f)$
    fin si
    si $(f \in RPG^i) \wedge (coste_{RPG^i}(f) > coste(f))$ **entonces**
      $coste_{RPG^i}(f) \leftarrow coste(f)$
    fin si
  fin para
  Expandir RPG^i
hasta que $RF^i = \emptyset$

Nótese que ninguno de los agentes maneja una representación completa del dis-RPG. Por el contrario, cada agente mantiene un grafo de planificación distinto internamente, de modo que su información privada no se revela al resto de agentes.

El algoritmo 2 resume el proceso de construcción del dis-RPG. En primer lugar, cada agente construye un grafo de planificación inicial teniendo en cuenta únicamente sus propias acciones y variables instanciadas. Para la construcción de este grafo inicial se ha seguido el algoritmo descrito en [HN01]. El grafo de planificación contiene un conjunto de niveles de acciones y variables instanciadas intercalados. El primer nivel de variables instanciadas contiene las variables instanciadas que forman parte del estado inicial, y el primer nivel de acciones contiene todas aquellas acciones aplicables en el estado inicial (acciones cuyas precondiciones figuran en el estado inicial). Los efectos de estas acciones se sitúan en el segundo nivel de variables instan-

ciadas, y de este modo el grafo se expande hasta que no aparecen nuevas variables instanciadas.

Una vez todos los agentes han construido el grafo inicial, el proceso de composición del dis-RPG se inicia. Este proceso constructivo consta de las siguientes etapas:

- **Intercambio de variables instanciadas**. Los agentes intercambian las variables instanciadas incluidas en sus grafos de planificación de acuerdo a lo especificado en la sección `shared-data` de sus ficheros de definición. De este modo, dos agentes i y j intercambiarán solo variables instanciadas de la forma $\langle v, d\rangle$ o bien $\langle v, \neg d\rangle$, donde $v \in \mathcal{V}^i \cap \mathcal{V}^j$ y $d \in \mathcal{D}_v^i \cap \mathcal{D}_v^j$.
- **Expansión del grafo de planificación**. Cada agente i actualiza su grafo RPG^i con las nuevas variables instanciadas recibidas. Si una variable instanciada f no está aún en RPG^i, se almacena de acuerdo a $coste(f)$. Si f está ya en RPG^i, su coste se actualiza si $coste_{RPG^i}(f) > coste(f)$. De este modo, los agentes sólo almacenan el mejor coste estimado para conseguir cada variable instanciada. Una vez actualizado RPG^i, el agente i lo expande comprobando si las nuevas variables instanciadas añadidas provocan la aparición de nuevas acciones en RPG^i. Las nuevas variables instanciadas producidas como efectos de estas nuevas acciones serán intercambiadas en la siguiente etapa de intercambio de variables.

4.2.2 Proceso de planificación distribuido

Tras el intercambio inicial de información, los agentes inicial el proceso de planificación distribuido (ver algoritmo 3). El proceso comprende dos fases intercaladas: un proceso de refinamiento individual y una fase de coordinación. En la primera fase, los agentes construyen individualmente refinamientos sobre un plan base centralizado utilizando el POP que tienen integrado. En la segunda fase, los agentes siguen un proceso de coordinación mediante el que intercambian los refinamientos obtenidos y seleccionan el más prometedor como el siguiente plan base.

4.2.2.1 Fase de refinamiento individual

Cada agente participante ejecuta un proceso POP individualmente para refinar el plan base actual Π, de modo que se obtiene un conjunto de refinamientos válidos sobre Π. De acuerdo a nuestra definición de plan refinamiento (ver sección 3.2.2), un plan refinamiento Π^i de un agente i sobre un plan

Algoritmo 3 Proceso de planificación distribuido para un agente i

$\Pi \leftarrow \Pi_0$
$R = \emptyset$
repetir
  Seleccionar meta abierta $g \in metasAbiertas(\Pi)$
  Refinar plan base Π_g individualmente
  $\forall j \neq i$, enviar $Refinamientos^i(\Pi_g)$ al agente j
  $\forall j \neq i$, recibir $Refinamientos^j(\Pi_g)$
  $Refinamientos(\Pi_g) \leftarrow Refinamientos^i(\Pi_g)$
  $\forall j \neq i$, $Refinamientos(\Pi_g) \leftarrow$
    $Refinamientos(\Pi_g) \cup Refinamientos^j(\Pi_g)$
  Evaluar $Refinamientos(\Pi_g)$
  $R \leftarrow R \cup Refinamientos(\Pi_g)$
  Seleccionar mejor plan $\Pi^i \in R$
  $\Pi \leftarrow \Pi^i$
  si $metasAbiertas(\Pi) = \emptyset$ **entonces**
    devolver Π
  fin si
hasta que $R = \emptyset$

base Π resolverá una de sus metas abiertas $g \in metasAbiertas(\Pi)$, además de todas las metas abiertas privadas g^i de la forma $\langle v, d\rangle$ o bien $\langle v, \neg d\rangle$ que surjan de dicha resolución, donde $v \in \mathcal{V}^i \wedge d \in \mathcal{D}^i_v \wedge ((\forall j \neq i, v \notin \mathcal{V}^j) \vee (\forall j \neq i, d \notin \mathcal{D}^j_v)) \wedge (g^i \notin metasAbiertas(\Pi))$.

4.2.2.2 Proceso de coordinación

El proceso de coordinación está basado en un liderazgo democrático, por el cual, un testigo, que designa al agente líder, es intercambiado entre los agentes siguiendo una estrategia round-robin. El algoritmo intercala la fase de coordinación con el refinamiento individual de planes. Cada iteración de la fase de coordinación es liderada por el agente que tiene el testigo en ese momento (agente líder). Una vez la fase de coordinación termina, el agente líder pasa el testigo al siguiente agente, que pasa a liderar la siguiente iteración.

En primer lugar, los agentes intercambian los planes refinamiento que han desarrollado para proceder a su evaluación. Dicha evaluación se realiza mediante una función heurística que puntúa la calidad del plan (ver sección 4.3). Así, el agente líder selecciona el plan refinamiento con mejor puntuación.

El plan seleccionado pasa a ser adoptado por los agentes como el nuevo

plan base Π. Si $metasAbiertas(\Pi) = \emptyset$, se devuelve el plan solución y el proceso termina. Dado que algunas de las metas abiertas pueden no ser visibles para algunos agentes, todos deben confirmar que Π es un plan solución de acuerdo con su visión de Π. Si el plan no es solución, el agente líder selecciona la siguiente meta a resolver $g \in metasAbiertas(\Pi)$, lo que da comienzo a una nueva iteración del proceso de refinamiento individual.

El algoritmo de planificación puede verse como una exploración conjunta del espacio de refinamientos llevada a cabo por los agentes. Los nodos del árbol de búsqueda representan los planes refinamiento y cada iteración del algoritmo expande un nodo diferente.

4.3 Planificador de Orden Parcial

Como se ha mencionado anteriormente, el elemento clave de nuestro sistema de planificación distribuido es el sistema POP integrado en todos los agentes. Se han introducido algunos cambios y extensiones al algoritmo clásico POP en el diseño de este componente para extenderlo a un contexto de PMA. Las siguientes secciones analizan estas extensiones, y describen las heurísticas de planificación desarrolladas para guiar el proceso de búsqueda POP.

4.3.1 Extensiones del POP

Nuestro modelo de PMA, construido a partir del método de planificación basada en refinamientos, introduce una serie de nuevos requisitos que fuerzan la introducción de una serie de cambios en el algoritmo clásico de POP. Hemos diseñado una versión modificada del algoritmo clásico de POP que satisface estos requisitos. Los cambios más relevantes introducidos en el algoritmo pueden resumirse como sigue:

1. **Plan inicial:** El algoritmo POP clásico comienza con un plan vacío. Nuestro diseño permite escoger cualquier plan como plan inicial. De este modo, los agentes pueden comenzar a buscar refinamientos tomando un plan base cualquiera como plan inicial del proceso POP.

2. **Resolución de metas abiertas**. El POP diseñado resuelve únicamente la meta abierta seleccionada por los agentes como meta actual. El resto de metas abiertas en el plan inicial son ignoradas. Una vez la meta abierta inicial es resuelta, el planificador tratará de resolver en cascada todas las metas abiertas surgidas de esa primera resolución y que sólo puedan ser resueltas por el agente que está llevando a

cabo el proceso de búsqueda (nótese que el dis-RPG proporciona esta información).

3. **Comprobación de solución**. Los sistemas POP clásicos devuelven únicamente planes solución, es decir, planes libres de amenazas y de metas abiertas. Nuestro sistema de planificación devuelve refinamientos, por lo que hemos redefinido la fase de comprobación de solución del algoritmo POP. Para que un plan sea devuelto por el POP como un plan refinamiento válido, debe cumplir las siguientes condiciones:

   - Debe estar libre de amenazas.
   - Debe incluir un enlace causal que soporte la meta abierta actual.
   - Si el plan ha añadido nuevas metas abiertas a consecuencia de la resolución de la meta actual, todas aquellas que sólo sean resolubles por el agente actual deben estar resueltas.

4. **Relanzamiento del proceso de planificación**. Habitualmente, el proceso POP termina una vez se ha encontrado un plan solución. Sin embargo, el POP diseñado permite relanzar el proceso una vez se ha hallado un plan refinamiento para obtener más refinamientos del plan base.

4.3.2 Funciones heurísticas

Nuestro POP aplica un enfoque de búsqueda informada, es decir, selecciona el siguiente plan parcial a refinar de acuerdo a una función heurística [RN03], que estima la calidad del plan. El valor heurístico para un plan parcial dado Π se expresa como $f(\Pi) = g(\Pi) + h(\Pi)$, donde $g(\Pi)$ representa el coste de alcanzar el plan actual desde el plan inicial, y $h(\Pi)$ estima el coste de alcanzar un plan solución desde Π.

La eficiencia del proceso de búsqueda está íntimamente relacionada a la calidad de la función heurística. Una de las limitaciones del paradigma POP, como se describe en la sección 2.3, estriba en la ausencia de heurísticas competitivas. Por ello, hemos tomado en consideración los intentos más recientes para definir una heurística competitiva para POP [NK01].

En concreto, se han adaptado dos heurísticas recientes (en adelante las heurísticas SUM y MAX), que estiman la calidad del plan en base a la información disponible en un grafo de planificación relajado, como es el caso de nuestro dis-RPG. Como trabajo en progreso, se está desarrollando en paralelo una tercera heurística para estimar la calidad de los planes parciales (ver capítulo 6).

La heurística SUM estima el coste de alcanzar una solución desde un plan dado calculando la suma de los costes de las metas abiertas del plan en el grafo de planificación relajado. Aunque no es una heurística admisibe (puede sobrestimar el coste de alcanzar una solución), esta heurística consigue buenos resultados, por lo que se ha utilizado en las pruebas experimentales desarrolladas (ver sección 5.3). La función heurística SUM, $f(\Pi) = g(\Pi) + h(\Pi)$, se calcula como sigue:

- $g(\Pi)$ es el número de acciones del plan refinamiento actual Π (exceptuando las acciones ficticias).
- $h(\Pi)$ se define como la suma de los costes de las precondiciones abiertas del plan actual $\mathcal{OC}(\Pi)$ en el grafo de planificación relajado.

La heurística MAX, por su parte, evalúa el plan actual considerando también los costes de las metas abiertas del plan. Sin embargo, para reducir al máximo la sobrestimación, se computa dicha estimación como el coste de la precondición más costosa en el grafo relajado. Por tanto, la función heurística MAX, $f(\Pi) = g(\Pi) + h(\Pi)$, se calcula como sigue:

- $g(\Pi)$ es el número de acciones del plan refinamiento actual Π (exceptuando las acciones ficticias).
- $h(\Pi)$ se define como el coste en el grafo de planificación relajado de la precondición p más costosa del plan, $p \in \mathcal{OC}(\Pi)$.

Capítulo 5

Implementación del sistema de planificación distribuido

Este capítulo analiza el proceso de implementación del sistema de planificación distribuido llevado a cabo y sus principales componentes, así como las tecnologías empleadas para su desarrollo. Además, se muestran los resultados experimentales obtenidos en las pruebas realizadas.

El capítulo se estructura como sigue: la sección 5.1 analiza la implementación desarrollada e introduce los principales módulos del sistema; la sección 5.2 documenta las fases de que ha constado el desarrollo del proyecto; y por último, la sección 5.3 muestra los resultados experimentales obtenidos.

5.1 Análisis de la implementación

Nuestro sistema de planificación distribuido, desarrollado en el lenguaje Java [Gos00], se estructura en cinco módulos diferentes: el agente de planificación, la interfaz gráfica de usuario, el analizador, el instanciador y el planificador.

Los diferentes módulos han sido desarrollados como bundles, siguiendo el estándar OSGi [All03], de modo que cada módulo se subdivide en dos paquetes Java: un paquete común, que proporciona una interfaz bien definida para utilizar el módulo sin entrar en detalles de implementación, y un paquete de servicio, que incluye la implementación completa del módulo. El uso de la tecnología OSGi permite que cada uno de los módulos pueda utilizarse independientemente del resto en futuros proyectos, lo que facilita la reutilización del código implementado.

La figura 5.1 muestra el diagrama de clases del paquete común correspondiente al planificador. Se puede observar en la figura que todas las clases de los paquetes comunes son en realidad interfaces Java, lo que permite al

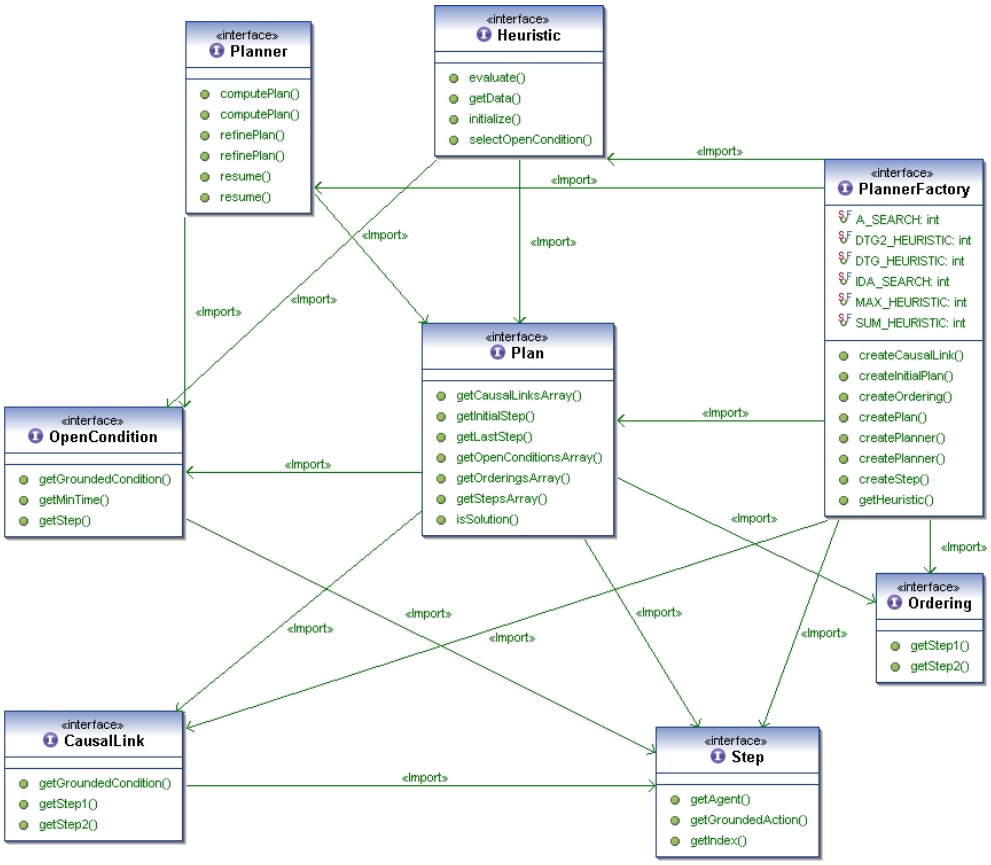

Figura 5.1: Diagrama de clases del paquete común del planificador

usuario del bundle abstraerse de la implementación concreta. El paquete de servicio asociado en el bundle llevará a cabo la implementación de las interfaces definidas en el paquete común.

La figura 5.2 muestra las dependencias entre los módulos implementados. La funcionalidad que ofrece cada uno de los módulos puede resumirse como sigue:

1. **Interfaz gráfica:** Permite al usuario comunicarse con la aplicación, lanzar tareas de planificación y consultar los resultados obtenidos, la traza del proceso de resolución y el árbol de búsqueda generado.

2. **Agente de planificación:** Este paquete configura cada uno de los agentes de planificación, que son los encargados de llevar a cabo la resolución de la tarea introducida por el usuario.

3. **Analizador:** Este módulo, se encarga de procesar los ficheros de entrada del agente y convertir la información en objetos de Java. Como

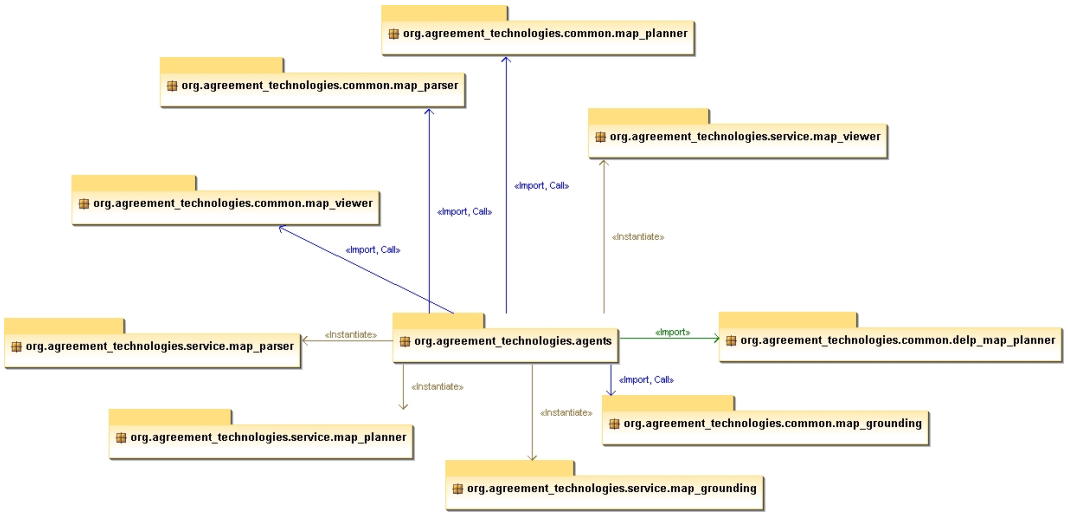

Figura 5.2: Dependencias entre los módulos implementados

se ha visto en la sección 4.1, los ficheros de entrada se codifican en un lenguaje propio de PMA basado en *PDDL3.1*.

4. **Instanciador:** El planificador implementado trabaja con dominios completamente instanciados, en los que las variables están completamente resueltas. El módulo de instanciación se encarga de resolver todas las variables a partir de la tarea procesada por el analizador. El módulo instanciador también se encarga de construir el dis-RPG descrito en la sección 4.2.

5. **Planificador:** Este módulo implementa el proceso de búsqueda POP. Aunque utiliza tecnología de planificación centralizada, se han realizado una serie de adaptaciones en el algoritmo de búsqueda para que éste sea compatible con nuestro modelo de planificación distribuida basada en refinamientos. El planificador puede trabajar también en modo centralizado.

El sistema de planificación ha sido desarrollado bajo la plataforma Magentix2 [FAS+10], desarrollada por el grupo GTI-IA del DSIC. Magentix2 proporciona los mecanismos de comunicación que permiten a los agentes interactuar. Se ha escogido Magentix2 dado que está dirigido para simplificar el desarrollo de sistemas multi-agente, además de cumplir con las especificaciones FIPA [ON98]. Magentix2 ofrece un conjunto de servicios que facilitan la implementación de sistemas multi-agente: servicio de nombres y páginas amarillas (DF), servicio de transporte de mensajes (MTS), y una completa librería de protocolos de interacción FIPA.

El módulo de agente de planificación hereda la funcionalidad de los agentes Magentix2, lo que, junto al planificador POP integrado, le proporciona capacidades de planificación distribuida. Esto implica que el agente es capaz de proporcionar servicios de planificación a otros agentes, además de trabajar conjuntamente con otros en tareas de planificación distribuida.

La figura 5.3 muestra la integración de los agentes de planificación en la plataforma Magentix2. Cuando un agente de planificación entra en el sistema, este es registrado en el DF (Directory Facilitator) y asociado a un servicio de planificación.

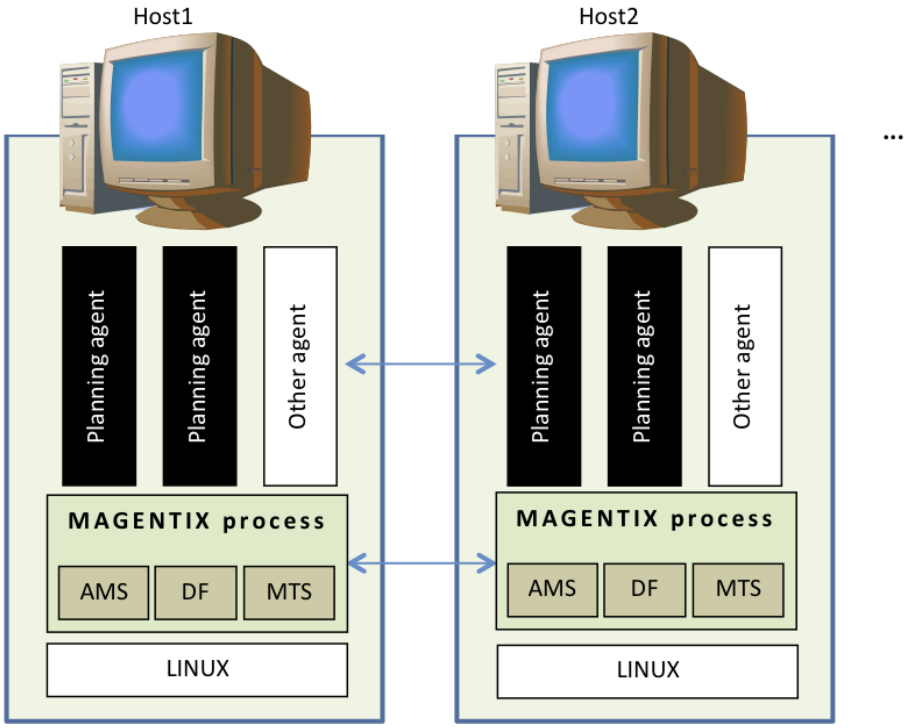

Figura 5.3: Estructura de la plataforma Magentix2

La figura 5.4 muestra el diagrama de secuencia de nuestro sistema de planificación distribuido. Como puede observarse, el usuario solicita la resolución de una tarea de planificación a la interfaz gráfica, que inicializa tantos agentes como el usuario haya especificado.

El agente de planificación es el componente central del sistema, y se encarga de solicitar servicios al resto de módulos, además de ejecutar el algoritmo de planificación basada en refinamientos descrito en el capítulo 4.2. En primer lugar, solicita al analizador que procese los ficheros de entrada, de forma que obtiene los objetos correspondientes a la tarea. A continuación,

envía al instanciador la salida del analizador, obteniendo el dominio instanciado.

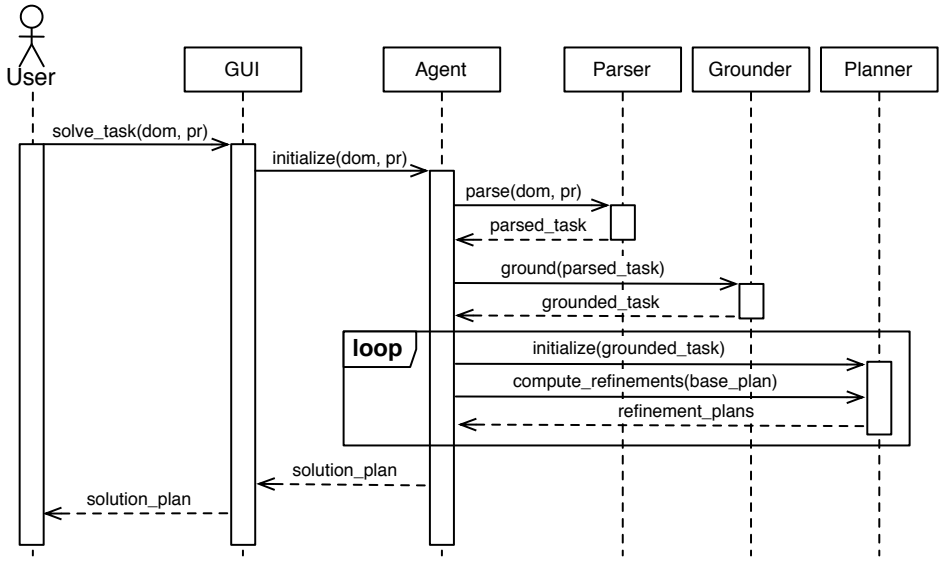

Figura 5.4: Diagrama de secuencia del sistema de planificación distribuido

Una vez dispone del dominio de planificación instanciado, el agente de planificación inicializa su planificador, configurándolo con dicho dominio. Una vez el planificador está inicializado, ambos módulos desarrollan el algoritmo presentado en el capítulo 4.2. El agente solicitará al planificador en cada iteración que procese el siguiente plan base, obteniendo los nuevos planes refinamiento.

Una vez completado el proceso, el agente de planificación devuelve el plan solución a la interfaz gráfica, que a su vez presenta al usuario la solución obtenida.

5.2 Desarrollo del proyecto

El proyecto se ha desarrollado en el último año siguiendo un ciclo de vida clásico, que comprende análisis, diseño, implementación y pruebas de la aplicación. Como se observa en la figura 5.5, que muestra el diagrama de Gantt del proyecto, las diferentes tareas se han desarrollado de modo completamente secuencial.

Las características de los módulos implementados facilitan esta secuencialización en las tareas desarrolladas. En la primera fase de la imple-

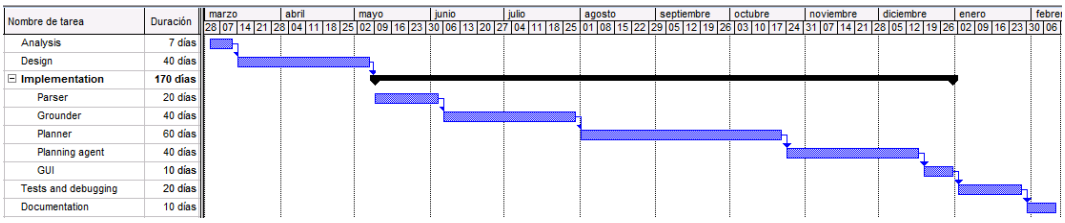

Figura 5.5: Diagrama de Gantt del proyecto

mentación se desarrolló el analizador, que no tiene ninguna dependencia con el resto de módulos. A continuación se implementó el instanciador, que depende directamente de la salida del analizador. Posteriormente se llevó a cabo la implementación del planificador, que toma como entrada la información generada por el módulo instanciador.

Los últimos módulos en implementarse han sido el agente planificador y la interfaz gráfica. El agente planificador requiere de los tres primeros módulos implementados, mientras que la interfaz gráfica interactúa únicamente con el módulo de agente planificador.

Una vez completada la fase de implementación, se ha llevado a cabo el proceso de prueba y debugging de la aplicación, corrigiendo los distintos fallos que han surgido en cada uno de los módulos. Una vez depurada la aplicación, se han realizado los experimentos para evaluar el rendimiento de la aplicación documentados en la siguiente sección. La última fase del proyecto comprendió la documentación de la aplicación y la redacción de la presente memoria.

5.3 Resultados experimentales

El sistema de planificación distribuido se ha probado mediante la ejecución de un conjunto de pruebas. Estas pruebas comparan la calidad de los planes obtenidos mediante nuestro sistema distribuido y a través de una aproximación centralizada. Hemos realizado también una serie de pruebas para determinar la escalabilidad del sistema, incrementando el número de agentes en pruebas sucesivas.

Para la realización de los tests, hemos diseñado dos dominios distintos de planificación, y generado un conjunto de tareas de prueba sobre cada uno de ellos. Las siguientes secciones presentan los dominios de PMA diseñados y los resultados de las distintas pruebas

5.3.1 Dominios de PMA

Los dominios de PMA diseñados para evaluar el sistema de planificación distribuido se han diseñado a partir de problemas de la vida real o de casos de estudio conocidos. Los siguientes apartados describen los dos dominios que se han diseñado para llevar a cabo las pruebas

5.3.1.1 Dominio de transporte

El primer dominio de PMA diseñado se muestra en la figura 5.6. Este dominio modela un escenario de transporte y almacenamiento, en el que los agentes pueden tomar el rol de una agencia de transporte o un almacén. El objetivo de la agencia de transporte es transportar una serie de paquetes a través de una red de ciudades. Por su parte, los agentes almacén se encargan de almacenar paquetes y cargar los camiones.

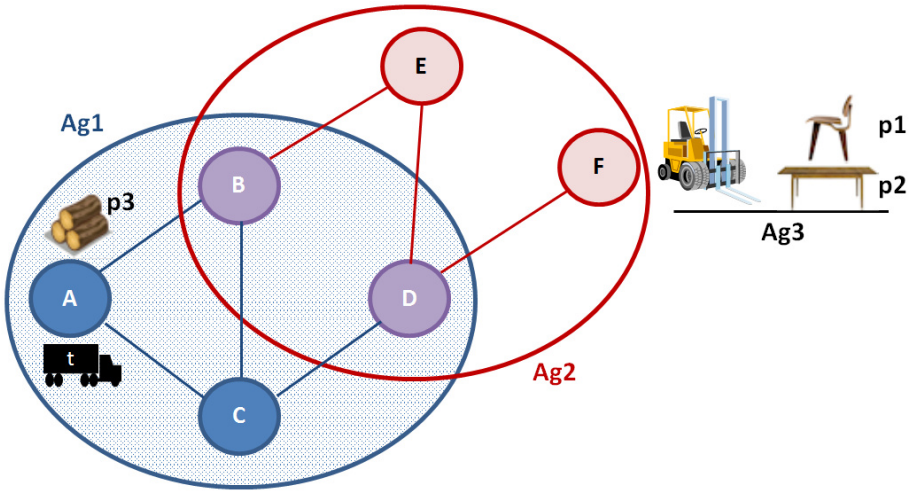

Figura 5.6: Ejemplo de tarea de transporte

Las agencias de transporte trasladan paquetes a través de una red de ciudades. El dominio describe una serie de enlaces bidireccionales sobre las ciudades, de modo que los camiones pueden usarlos para trasladarse de una ciudad a otra. Los camiones de la agencia de transporte sólo pueden desplazarse entre las ciudades incluidas en el área de trabajo de la agencia. De este modo, un agente deberá colaborar con otros para enviar paquetes fuera de su área de trabajo. Las agencias de transporte tienen las siguientes capacidades:

- `cargar`: Carga un `paquete` en un `camión`. Tanto el `paquete` como el

`camión` deben estar situados en la misma `ciudad` dentro del `área` del `agente`.

- `descargar`: Descarga un `paquete` de un `camión` situado en una `ciudad` dentro del `área` del `agente`.
- `conducir`: Conduce un `camión` de una `ciudad` a otra. Ambas ciudades deben estar localizadas dentro del `área` del `agente`.

El dominio del almacén es similar al dominio clásico del *mundo de bloques*, donde los paquetes pueden ser apilados o desapilados de una mesa o de otros paquetes. En este caso, la mesa tiene espacio para una única pila de paquetes, y hay dos tipos de paquetes, materias primas y productos terminados. El agente almacén puede entregar productos terminados en la ciudad adyacente al almacén (la ciudad de intercambio), y almacenar materias primas procedentes de los camiones. Los almacenes pueden realizar las siguientes acciones:

- `obtener`: Obtiene una `materia-prima` de la `ciudad-de-intercambio`.
- `entregar`: Entrga un `producto-terminado` en la `ciudad-de-intercambio`.
- `apilar`: Apila un `paquete` sobre otro `paquete`, o lo deja encima de la `mesa`, si esta está libre.
- `desapilar`: Desapila un `paquete` de otro `paquete`, o lo levanta de la `mesa`, si no tenía nada debajo.

5.3.1.2 Dominio del cuadro

Este dominio, adaptado del caso de estudio presentado en [PSJ98], describe una situación en la que un conjunto de trabajadores debe colgar una serie de cuadros en muros, para lo cual deben recoger una serie de herramientas distribuidas por diferentes habitaciones. Por tanto, los agentes deben moverse a través de las habitaciones para recoger las herramientas y colgar los cuadros. El dominio establece una serie de enlaces bidireccionales que indican las conexiones entre habitaciones.

La figura 5.7 muestra un ejemplo de este dominio de PMA. A diferencia del dominio de transporte, los agentes en el dominio del cuadro no están especializados, es decir, todos comparten las mismas capacidades, que pueden describirse como sigue:

- `recoger`: Obtiene una `herramienta` de una `habitación`. El `agente` y la `herramienta` deben estar situados en la misma `habitación`.

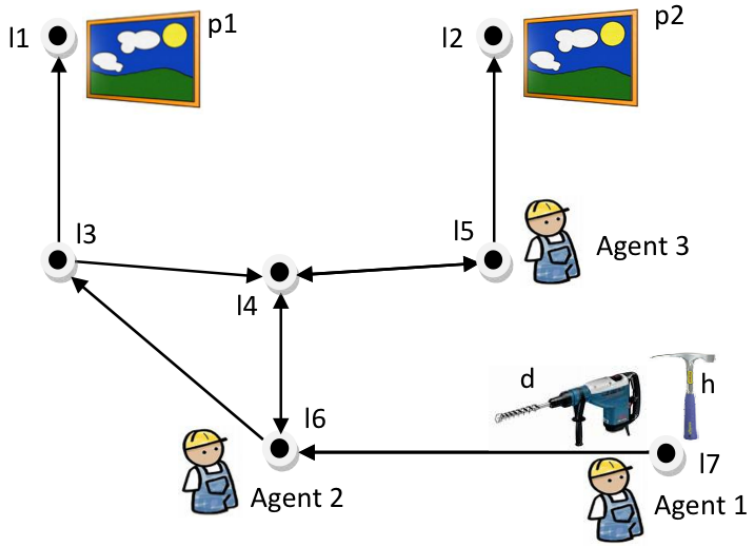

Figura 5.7: Ejemplo de tarea del cuadro

- `dejar`: Deja una herramienta `herramienta` en una `habitación`. El `agente` debe tener la `herramienta` para poder dejarla.
- `pasar`: Pasa la `herramienta` a otro `agente`. Ambos agentes deben estar en la misma `habitación`, y el primer `agente` debe llevar la `herramienta` para poder pasársela al segundo.
- `caminar`: Camina de una `habitación` a otra. Ambas habitaciones deben estar enlazadas para que el `agente` pueda caminar de una a otra.
- `colgar`: Cuelga un `cuadro` en una `habitación` con una `herramienta`. Tanto el `agente` como el `cuadro` deben estar en la misma `habitación`, y el `agente` debe llevar la herramienta `herramienta` para poder colgar el `cuadro`.

5.3.2 Pruebas y resultados

Los siguientes apartados muestran los resultados experimentales obtenidos. Hemos llevado a cabo dos pruebas diferentes[1]. La primera prueba compara la calidad de los planes solución obtenidos mediante planificación centralizada (con un sólo agente) y mediante nuestro planificador distribuido (con varios

[1]Todas las pruebas se han realizado en una única máquina con un procesador Intel Core 2 Quad a 2.83 GHz y 8 GB de memoria RAM.

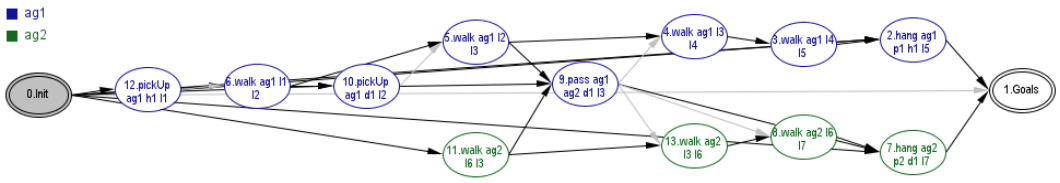

Figura 5.8: Plan solución para la tarea distribuida Cuadro2

agentes trabajando conjuntamente). Para ello, se ha definido un conjunto de tareas de PMA y una versión centralizada equivalente. Por último, hemos realizado una prueba adicional para medir la escalabilidad y la robustez de nuestro sistema de planificación distribuido. El experimento consiste en lanzar varias veces la misma tarea de PMA, añadiendo un agente más en cada ejecución, de modo que podamos estudiar el impacto en el tiempo de ejecución y los mensajes intercambiados que supone la adición de un agente a la tarea.

5.3.2.1 Planificación distribuida contra planificación centralizada

Este primer conjunto de pruebas compara la calidad de los planes solución obtenidos mediante nuestro planificador distribuido con otros generados por un sólo agente planificando de forma centralizada. El conjunto de pruebas contiene 20 tareas, 10 por cada dominio de PMA, de dificultad creciente.

Como se ha indicado en la sección 5.3.1, los agentes en las tareas de *transporte* están especializados, mientras que en el dominio del *cuadro* todos comparten las mismas habilidades. Esto provoca que los agentes de transporte se vean obligados a colaborar unos con otros para resolver las tareas, mientras que cualquier obrero en el dominio del *cuadro* puede resolver la tarea completa por sí mismo.

La tabla 5.1 muestra los resultados obtenidos. #Ag indica el número de agentes que lleva a cabo la tarea de PMA en las pruebas distribuidas. #Acciones y #PT se refieren al número de acciones y *pasos de tiempo* del plan solución, respectivamente (nótese que no contamos las acciones ficticias de los planes). Por último, Paralelismo indica el número máximo de ramas en paralelo en los planes solución.

Los pasos de tiempo son las unidades de tiempo necesarias para ejecutar el plan, es decir, hacen referencia a la duración del plan. Por ejemplo, la figura 5.8 muestra el plan solución para la tarea de PMA Cuadro2. Aunque el plan tiene 12 acciones (sin tener en cuenta las ficticias), puede ser ejecutado en

Problema	Pl. distribuida				Pl. centralizada	
	#Ag	#Acciones	#PT	Paralelismo	#Acciones	#PT
Transporte1	2	14	11	2	14	11
Transporte2	2	11	9	2	11	9
Transporte3	3	9	5	2	9	5
Transporte4	3	11	6	2	11	6
Transporte5	4	13	6	3	13	6
Transporte6	4	11	5	3	11	5
Transporte7	5	10	8	2	10	8
Transporte8	5	15	9	3	15	9
Transporte9	6	11	5	3	11	5
Transporte10	6	17	10	3	17	10
Cuadro1	2	11	6	2	14	14
Cuadro2	2	12	8	2	11	11
Cuadro3	3	6	2	3	8	8
Cuadro4	3	11	7	2	11	11
Cuadro5	4	8	2	4	11	11
Cuadro6	4	10	6	2	10	10
Cuadro7	5	8	5	2	8	8
Cuadro8	5	10	2	5	14	14
Cuadro9	6	9	5	2	9	9
Cuadro10	6	12	2	6	17	17

Tabla 5.1: Comparación entre planificación centralizada y distribuida

sólo ocho pasos de tiempo, dado que la mayoría de sus acciones se ejecutan en dos ramas paralelas. Por tanto, la duración del plan de la figura 5.8 es de 8 unidades de tiempo.

Como puede observarse en la tabla 5.1 la aproximación distribuida obtiene los mismos resultados en términos del número de acciones y pasos de tiempo en las pruebas de *transporte*. Aunque los agentes participantes en las pruebas distribuidas tienen una visión parcial del dominio y diferentes capacidades, el sistema de planificación distribuido es capaz de obtener planes de la misma calidad que un sistema centralizado. Por tanto, la calidad de los planes obtenidos por nuestro planificador distribuido no se ve afectada por la visión limitada del dominio que tiene cada agente ni por la existencia de información privada.

Los resultados del dominio del *cuadro* presentan más diferencias entre las dos aproximaciones. El planificador centralizado genera soluciones completamente lineales, dado que el único agente planificador es también el único agente ejecutor del plan. Sin embargo, el planificador distribuido toma ventaja del hecho de disponer de varios agentes de planificación y ejecución cooperando, por lo que se consiguen planes en los que los agentes trabajan colectivamente, alcanzando distintos objetivos en paralelo o bien colaborando para resolver un mismo objetivo. Por ejemplo, la figura 5.8 muestra a un agente recogiendo una herramienta y pasándola a otro agente, de modo que cada uno de ellos puede colgar un cuadro en paralelo. Este paralelismo, como se aprecia en la tabla 5.1, permite reducir el número de acciones del plan y la duración del mismo.

En conclusión, aunque el planificador distribuido es un enfoque más costoso en cuanto a tiempo de ejecución (véase la siguiente sección para un análisis de escalabilidad), consigue planes solución de igual o mejor calidad en términos de acciones y duración. El enfoque distribuido promueve la colaboración entre agentes y es efectivo paralelizando acciones, lo que reduce la duración de los planes.

5.3.2.2 Análisis de escalabilidad

Esta sección muestra las pruebas realizadas para evaluar la escalabilidad de nuestro sistema de planificación distribuido, esto es, cómo afecta el incremento del número de agentes a la eficiencia del sistema. Para ello, se han generado ocho pruebas para los dominios de *transporte* y del *cuadro*. Cada prueba incremente el número de agentes en uno, manteniendo el resto de parámetros de la tarea sin modificar.

Todas las pruebas de *transporte* incluyen diez ciudades, un camión, una mesa vacía en el almacén y un paquete de materia prima. Todos los problemas

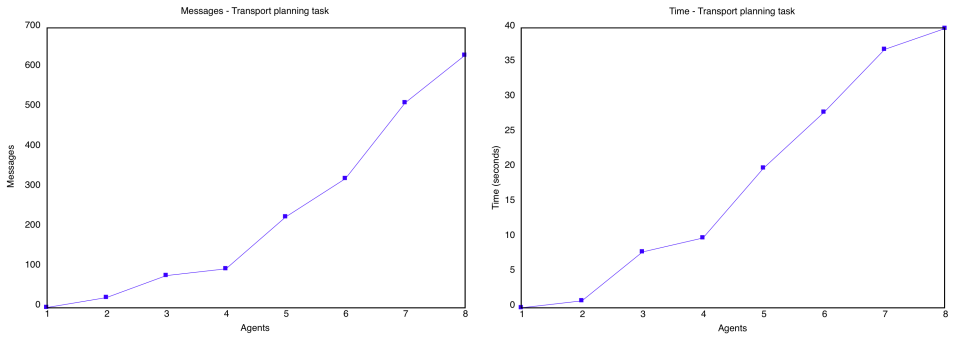

Figura 5.9: Resultados de escalabilidad para el dominio de transporte

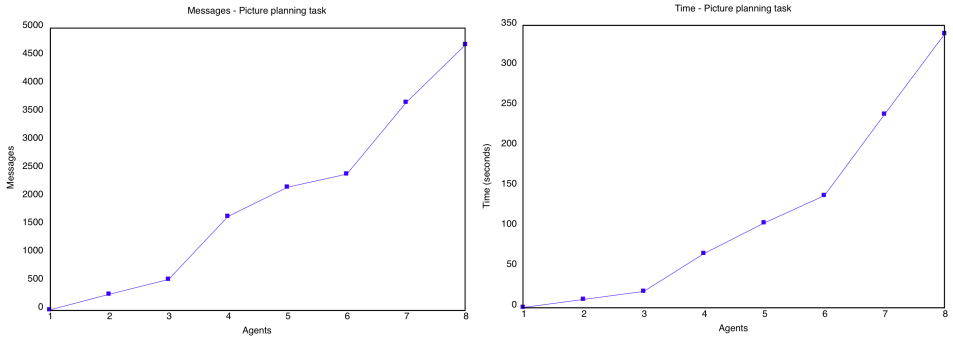

Figura 5.10: Resultados de escalabilidad para el dominio del cuadro

incluyen un único agente almacén, y cada problema incrementa el número de agencias de transporte en una. La meta para todas las pruebas es entregar el paquete en el almacén, que debe depositarlo sobre la mesa vacía. La solución óptima para todas las pruebas incluye diez acciones, y obliga a participar en ella al menos a una agencia de transporte y al almacén.

Respecto al dominio del *cuadro*, todos los problemas incluyen dos herramientas y doce habitaciones distintas. El objetivo de todas las tareas es colgar dos cuadros distintos. La solución óptima tiene ocho acciones e involucra a dos agentes distintos, cada uno de los cuales coge una herramienta y cuelga uno de los cuadros.

Las figuras 5.9 y 5.10 muestran los resultados para cada dominio. Como se puede observar, el número de mensajes intercambiados experimenta un notable incremento con cada nuevo agente introducido en la tarea. Lo mismo ocurre con el tiempo de ejecución, que está condicionado por el número de mensajes intercambiados por los agentes.

Estos resultados se deben al creciente número de planes refinamiento

propuestos por los agentes. Estos planes son comunicados a todos los agentes, por lo que la adición de un agente sobrecarga el número de mensajes intercambiados. Además, cada nuevo agente puede proponer nuevos refinamientos, y estos refinamientos pueden a su vez ser escogidos como plan base, lo que aumenta la complejidad del árbol de búsqueda.

Nótese que el número de mensajes es mucho mayor en las tareas del *cuadro*, pese a que ambas tareas tienen un tamaño y complejidad similares. Esto se debe a que los agentes del *cuadro* comparten las mismas habilidades, y por tanto cualquier agente puede proponer un refinamiento para cualquier meta. En el dominio de *transporte*, sin embargo, los agentes son especializados, lo que provoca que no todos los agentes puedan resolver cualquier meta, reduciéndose de este modo los planes propuestos, y en consecuencia los mensajes intercambiados.

En conclusión, el número de agentes en el planificador distribuido que hemos desarrollado tiene una influencia importante en su eficiencia, dado que el número de mensajes intercambiados constituye uno de los cuellos de botella del sistema. Un mayor número de agentes conlleva un mayor número de propuestas de plan, lo que redunda en una mayor complejidad del árbol de búsqueda, que es el otro factor que afecta al rendimiento del sistema de planificación distribuido implementado.

Capítulo 6

Conclusiones y trabajo futuro

Este último capítulo resume las contribuciones principales presentadas en este trabajo, y resume las futuras líneas de trabajo a desarrollar y lista las publicaciones relacionadas con el presente trabajo.

6.1 Resumen de contribuciones

En el presente trabajo se ha descrito el diseño e implementación de un sistema de planificación distribuido basado en un modelo que combina planificación basada en refinamientos y Planificación de Orden parcial (POP).

El sistema de planificación distribuido se compone de un grupo de agentes con capacidades de planificación, que generan planes individualmente y se coordinan a través de un protocolo en el cual los agentes desempeñan por turnos el rol de líder del proceso.

El algoritmo incluye un intercambio inicial de información por el que los agentes construyen un grafo de planificación relajado distribuido. La fase de resolución de problemas combina la planificación individual de refinamientos con un proceso de coordinación mediante el cual se selecciona el refinamiento más prometedor como el siguiente plan base.

El algoritmo POP se ha modificado para tratar con los requisitos adicionales que surgen en un contexto de PMA. Del mismo modo, se ha diseñado un lenguaje de planificación que incluye un conjunto de estructuras para soportar estos requisitos.

El sistema de planificación implementado está basado en el lenguaje Java. El desarrollo ha involucrado diferentes tecnologías, como la plataforma Magentix2 [FAS+10] un sistema multi-agente que implementa los protocolos de comunicación FIPA [ON98], y que constituye el componente multi-agente del sistema, que permite a los agentes comunicarse e intercambiar mensajes. El

componente POP desarrollado presenta una implementación modular, flexible y extendible que permite al usuario integrar nuevas heurísticas y métodos de búsqueda. Todos los componentes del sistema se han construido siguiendo los estándares OSGi [All03], lo que facilita su integración en una plataforma orientada a servicios.

Por último, los resultados experimentales muestran que el enfoque distribuido del sistema implementado permite obtener planes de igual o superior calidad que un enfoque de planificación centralizado, tanto en número de acciones como en duración del plan.

6.2 Trabajo futuro

El trabajo futuro a realizar sigue dos direcciones distintas: por un lado, proseguiremos con el desarrollo de una nueva heurística para POP que mejore el rendimiento del planificador; y por otro lado, probaremos métodos de coordinación alternativos para mejorar el proceso de selección de plan base.

La heurística que está en desarrollo se base en el trabajo de [Hel04], que introduce el formalismo de *Grafos de Transición de Dominio* (DTGs). Estos grafos dirigidos representan las formas en que una variable de estado puede cambiar su valor de acuerdo a las acciones de planificación.

Nuestra heurística en desarrollo utiliza los DTGs para generar una estructura de datos intermedia que aproxima la estructura de un plan solución para la tarea de planificación, el Grafo de Transición de Plan (PTG).

El trabajo en esta heurística se enfoca por tanto en mejorar la construcción del PTG, de modo que esté lo más cerca posible de los planes solución, lo que redundará en una mayor calidad de la heurística.

El segundo punto a desarrollar como trabajo futuro se centra en la implementación de métodos de coordinación alternativos para mejorar la evaluación y selección de planes refinamiento. En este sentido, se ha desarrollado un modelo de argumentación que permite afrontar la evaluación y selección de planes desde un punto de vista social.

Nuestro modelo de argumentación adapta la instanciación de un esquema argumentativo y las cuestiones críticas asociadas siguiendo la representación computacional de argumentación práctica presentada en [ABCM06, ABC07]. Este enfoque es particularmente adecuado para representar planes multiagente concurrentes.

Además del modelo de argumentación, el trabajo futuro es probar otros métodos de coordinación entre agentes para reemplazar la actual función heurística ad-hoc, de modo que mejoremos la eficiencia del sistema de planificación distribuido implementado.

6.3 Publicaciones relacionadas

Los siguientes artículos de investigación están directamente relacionados con el presente trabajo, y han sido publicados durante su desarrollo:

- O. Sapena, A. Torreño, E. Onaindia. *On the Construction of Joint Plans through Argumentation Schemes.* 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011).
- O. Sapena, E. Onaindia, A. Torreño. *On the use of Argumentation in Multi-Agent Planning.* 19th European Conference on Artificial Intelligence (ECAI 2010).
- E. Onaindia, O. Sapena, A. Torreño. *Argumentation-based Planning in multi-agent systems.* Negotiation and argumentation in Multiagent Systems. Bentham eBooks. In press (2011).
- E. Onaindia, O. Sapena, A. Torreño. *Cooperative Distributed Planning through Argumentation.* International Journal of Artificial Intelligence. ISSN: 0974-0635. In Press (2010).
- A. Torreño, E. Onaindia, O. Sapena. *Reaching a common agreement discourse universe on Multi-Agent Planning.* 5th International Conference on Hybrid Artificial Intelligence Systems (HAIS 2010).
- S. Pajares, E. Onaindia, A. Torreño. *An architecture for Defeasible-Reasoning-based Cooperative Distributed Planning.* 9th International Conference on Cooperative Information Systems (CoopIS 2011).

Bibliografía

[ABC07] K. Atkinson and T. Bench-Capon. Practical reasoning as presumptive argumentation using action based alternating transition systems. *Artificial Intelligence*, 171:855–874, 2007.

[ABCM06] Katie Atkinson, Trevor J. M. Bench-Capon, and Peter McBurney. Computational representation of practical argument. *Synthese*, 152(2):157–206, 2006.

[All03] O.S.G. Alliance. *OSGi service platform, release 3.* IOS Press, Inc., 2003.

[BB01] C. Boutilier and R. I. Brafman. Partial-order planning with concurrent interacting actions. *Journal of Artificial Intelligence Research*, 14:105–136, 2001.

[BCF+95] A. Barrett, D. Christianson, M. Friedman, K. Golden, C. Kwok, J.S. Penberthy, Y. Sun, and D. Weld. UCPOP user's manual (version 4.0). *Technical Report 93-09-06d, University of Washington*, 1995.

[BEG+92] J. Blythe, O. Etzioni, Y. Gil, R. Joseph, D. Kahn, C. Knoblock, S. Minton, A. PÈrez, S. Reilly, M. Veloso, and X. Wang. Prodigy 4.0: The manual and tutorial. *Technical report CMU-CS-92-150. Carnegie Mellon University*, 1992.

[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.

[BN09] M. Brenner and B. Nebel. Continual planning and acting in dynamic multiagent environments. *Journal of Autonomous Agents and Multiagent Systems*, 19(3):297–331, 2009.

[BRR10] A. Belesiotis, M. Rovatsos, and I. Rahwan. Agreeing on plans through iterated disputes. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 765–772, 2010.

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

[CDB05] J.S. Cox, E.H. Durfee, and T. Bartold. A distributed framework for solving the multiagent plan coordination problem. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 821–827. ACM, 2005.

[Cle05] B. J. Clement. *Proceedings of the Workshop on Multi-agent Planning and Scheduling.* International Conference on Automated Planning and Scheduling ICAPS-05, 2005.

[dDOW99] M.E. desJardins, E.H. Durfee, C.L. Ortiz, and M.J. Wolverton. A survey of research in distributed continual planning. *AI Magazine*, 20(4):13–22, 1999.

[DKS+00] L. Derek, H. Kautz, B. Selman, B. Bonet, H. Geffner, J. Koehler, M. Brenner, F. Hoffmann, J.and Rittinger, C. Anderson, D. Weld, D. Smith, M. Fox, and D. Long. The aips-98 planning competition. *AI Magazine*, 21:13–33, 2000.

[DL91] E. H. Durfee and V. Lesser. Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics, Special Issue on Distributed Sensor Networks*, 21(5):1167–1183, 1991.

[DL92] Keith Decker and Victor R. Lesser. Generalizing the partial global planning algorithm. *International Journal of Cooperative Information Systems*, 2(2):319–346, 1992.

[Dur99] E. H. Durfee. *Distributed problem solving and planning*, volume In Gerhard Weiss editor, pages 118–149. The MIT Press, San Francisco, CA, 1999.

[dWtMW05] M. de Weerdt, A. ter Mors, and C. Witteveen. Multi-agent planning. an introduction to planning and coordination. In *Handouts of the European Agent Systems Summer School (EASSS-05)*, pages 1–32, 2005.

[Ede03] S. Edelkamp. Taming numbers and durations in the model checking integrated planning system. *Journal of Artificial Intelligence Research (JAIR)*, 20:195–238, 2003.

[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.

[ER96] E. Ephrati and J. S. Rosenschein. Deriving consensus in multiagent systems. *Artificial Intelligence*, 87(1-2):21–74, 1996.

[FAS+10] R.L. Fogués, J.M. Alberola, J.M. Such, A. Espinosa, and A. Garcia-Fornes. Towards dynamic agent interaction support in open multiagent systems. In *Proceedings of the 2010 conference on Artificial Intelligence Research and Development: Proceedings of the 13th International Conference of the Catalan Association for Artificial Intelligence*, pages 89–98. IOS Press, 2010.

[FL03] M. Fox and D. Long. PDDL2.1: an extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.

[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. In *Logic-based artificial intelligence*, pages 187–209. Kluwer Academic Publishers, 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.

[GL05] A. Gerevini and D. Long. Plan constraints and preferences in PDDL3. *Technical Report, Department of Electronics for Automation, University of Brescia, Italy*, 2005.

[GL06] A. Gerevini and D. Long. Preferences and soft constraints in PDDL3. In *ICAPS Workshop on Planning with Preferences and Soft Constraints*, volume 6, pages 46–53. Citeseer, 2006.

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

[Gos00] J. Gosling. *The Java language specification*. Prentice Hall, 2000.

[Hel04] M. Helmert. A planning heuristic based on causal graph analysis. *Proceedings of ICAPS*, pages 161–170, 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.

[JR11] Anders Jonsson and Michael Rovatsos. Scaling up multiagent planning: A best-response approach. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS)*, pages 114–121. AAAI, 2011.

[Kam97] S. Kambhampati. Refinement planning as a unifying framework for plan synthesis. *AI Magazine*, 18(2):67–97, 1997.

[KNHD97] J. Koehler, B. Nebel, J. Hoffmann, and Y. Dimopoulos. Extending planning graphs to an adl subset. *Proceedings of the Fourth European Conference in Planning*, pages 273–285, 1997.

[Kov11] D. L. Kovacs. Complete BNF description of PDDL3.1. Technical report, 2011.

[KPT02] Gal A. Kaminka, David V. Pynadath, and Milind Tambe. Monitoring teams by overhearing: A multi-agent plan-recognition approach. *Journal of Artificial Intelligence Research (JAIR)*, 17:83–135, 2002.

[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.

[McD96] D. McDermott. A heuristic estimator for means-ends analysis in planning. *Proceedings of the International Conference on Artificial Intelligence Planning Systems*, 3:142–149, 1996.

[McD00] D. McDermott. The 1998 AI planning systems competition. *AI Magazine*, 21(2):35–55, 2000.

[MGH+98] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL: The planning domain definition language. 1998.

[MH69] J. McCarthy and P.J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

[NBD10] R. Nissim, R.I. Brafman, and C. Domshlak. A general, fully distributed multi-agent planning algorithm. In *Proceedings of the 9th International Conference on Autonomous Agents*, pages 1323–1330, 2010.

[NK01] X.L. Nguyen and S. Kambhampati. Reviving partial order planning. In *Proceedings of the 17th International Joint Conference on Artificial intelligence-Volume 1*, pages 459–464. Morgan Kaufmann Publishers, 2001.

[ON98] P.D. O'Brien and R.C. Nicol. Fipa - towards a standard for software agents. *BT Technology Journal*, 16(3):51–59, 1998.

[Ped89] E. Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In *Proc. 1st Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'89)*, pages 324–332, 1989.

[PJP97] M.E. Pollack, D. Joslin, and M. Paolucci. Flaw selection strategies for partial-order planning. *Arxiv preprint cs/9706101*, 1997.

[PSJ98] S. Parsons, C. Sierra, and N. Jennings. Agents that reason and negotiate by arguing. *Journal of Logic and computation*, 8(3):261, 1998.

[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.

[RN03] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

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

[TBdWW02] Hans Tonino, André Bos, Mathijs de Weerdt, and Cees Witteveen. Plan coordination by revision in collective agent based systems. *Artificial Intelligence*, 142(2):121–145, 2002.

[TNP10] Y. Tang, T. Norman, and S. Parsons. A model for integrating dialogue and the execution of joint plans. *Argumentation in Multi-Agent Systems*, pages 60–78, 2010.

[VDKDW05] R. Van Der Krogt and M. De Weerdt. Plan repair as an extension of planning. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling*, pages 161–170, 2005.

[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.

[Wil88] D.E. Wilkins. *Practical Planning: Extending the Classical AI Planning Paradigm*. Morgan Kaufmann, 1988.

[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.