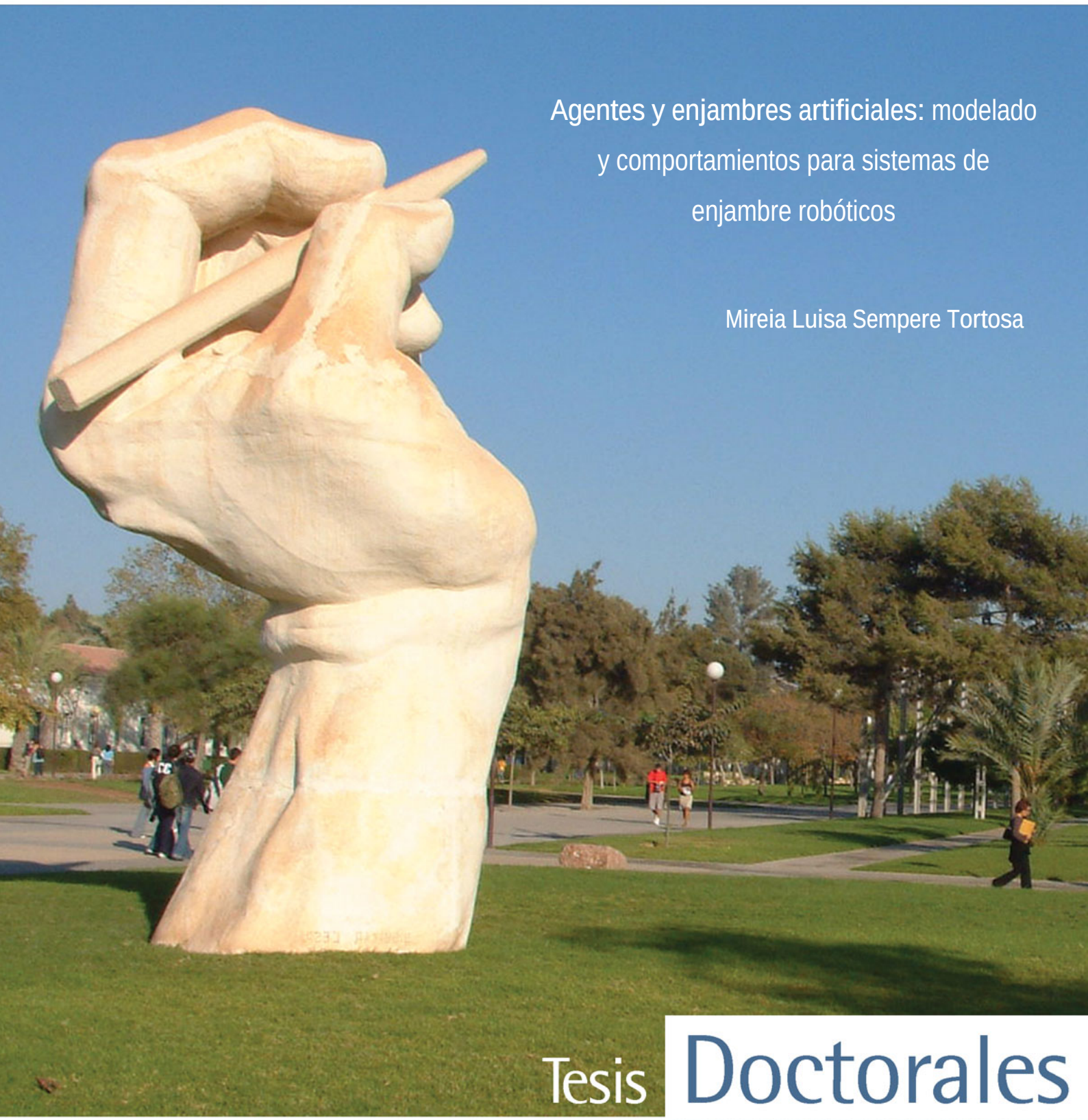
Universitat d'Alacant
Universidad de Alicante

Agentes y enjambres artificiales: modelado y comportamientos para sistemas de enjambre robóticos

Mireia Luisa Sempere Tortosa



Tesis Doctorales

www.eltallerdigital.com

UNIVERSIDAD de ALICANTE

Agentes y enjambres artificiales: modelado y comportamientos para sistemas de enjambre robóticos

TESIS DOCTORAL

Autora
Mireia Luisa Sempere Tortosa

Directores
Fidel Aznar Gregori
Mar Pujol López

Universidad de Alicante
Dpto. de Ciencia de la Computación e Inteligencia Artificial

Diciembre de 2013

Agradecimientos

En primer lugar me gustaría expresar mi más sincero agradecimiento a mis directores de tesis, Mar y Fidel, por su paciencia, su apoyo y su comprensión durante todo este tiempo.

Me gustaría agradecer de forma especial a Ramón, por haberme dado la oportunidad de ingresar en el mundo de la investigación y por toda su ayuda y sus consejos.

Me gustaría también agradecer a toda la gente que ha contribuido en cierta manera en esta tesis. A Javi, por ayudarme con la experimentación de los comportamientos. Y a Pilar por escucharme y sugerir algunas ideas. En general, a los componentes del grupo de investigación Informática Industrial e Inteligencia Artificial y al Departamento de Ciencia de la Computación e Inteligencia Artificial de la Universidad de Alicante.

A Virgilio, que en esta última etapa de la tesis se ha cargado de trabajo y quitado horas de sueño para que yo pudiera terminarla.

También quiero agradecer a mis amigos. Gracias a Jordi, Alma, Cati y Jorge porque, aun a un vuelo de distancia, siempre están ahí, en los buenos y en los malos momentos.

Finalmente, quiero agradecer a mi familia: a mis padres, por estar siempre a mi lado, y por intentar entender y apoyarme en todas mis decisiones; a Mariam y Jose, por ser unos buenos hermanos; y a Pablo, porque es el niño de mis ojos. Y por último, a Mario y a Pako, por su apoyo incondicional, por sus "¿y cuándo vas a terminar la tesis?", por toda su paciencia y por entender que no les dedicara todo el tiempo que merecen.

A mis padres, por todo su apoyo.
A Pako y Mario, por hacer cada día especial.

Resumen

La robótica de enjambre es un campo de investigación dentro del área de la robótica que estudia la coordinación de un gran número de robots simples. Este campo de investigación se inspira en el comportamiento observado en los insectos sociales, los cuales son grandes ejemplos de cómo un gran número de individuos simples pueden interactuar para crear sistemas inteligentes colectivos. En estos sistemas el comportamiento colectivo emerge de forma auto-organizada a partir de las interacciones entre los individuos y entre los individuos y el entorno.

De la misma manera, en los sistemas de enjambre artificiales, la inteligencia es una propiedad emergente a partir del comportamiento global del enjambre. En estos sistemas, el enjambre es capaz de llevar a cabo tareas, de manera global, que están fuera de las capacidades de un robot individual.

Los sistemas robóticos de enjambre deben cumplir una serie de características que los diferencian de otros sistemas multi-robóticos. Algunas de estas características son compartidas con los sistemas multi-agente, por lo tanto, éstos pueden ser una alternativa para la construcción de este tipo de sistemas.

En este trabajo se presentan un conjunto de comportamientos colectivos para enjambres artificiales. Para dotar de un marco de implantación a estos comportamientos, se define un modelo de arquitectura híbrida para el control de un enjambre de robots basada en un sistema multi-agente. Una de las características básicas de este modelo es su división en capas, que permite utilizar la capa inferior, de enjambre puro, de manera aislada, para tareas donde el comportamiento colectivo del enjambre emerge únicamente a partir de las interacciones entre los agentes y el entorno.

En primer lugar, se analizan tres comportamientos básicos de robótica de enjambre: agregación, movimiento coordinado (*flocking*) y dispersión. Posteriormente, se definen dos comportamientos concretos para un enjambre de robots. El primero de éstos muestra un comportamiento para la localización de recursos en el entorno, donde los robots son capaces de localizar la

fuente de recursos más prometedora en entornos desconocidos, con ruido y con diversas fuentes de recursos. El segundo caso define un comportamiento capaz de detectar, monitorizar, cubrir y marcar el perímetro de un vertido petrolífero marítimo.

Abstract

Swarm robotics is a research field within the area of robotics that studies the coordination of a large number of simple robots. This field is inspired by the behaviour observed in social insects, which are great examples of how a large number of simple individuals can interact to create collective intelligent systems. In these systems the collective behaviour emerges in a self-organized way from the interactions among individuals and between individuals and the environment.

Similarly, in artificial swarm systems, intelligence is an emergent property from the global behaviour of the swarm. In these systems, the swarm is able to carry out tasks, globally, that are beyond the capabilities of a single robot.

Swarm robotic systems must accomplish a number of features that distinguish them from other multi-robotic systems. Some of these features are shared by multi-agent systems, therefore, they can be an alternative for the construction of swarm systems.

This thesis presents a set of collective behaviours for artificial swarms. In order to provide a framework for implementing these behaviours, a hybrid architecture for the control of a swarm of robots based on a multi-agent system is defined. One of the basic features of this architecture is that the division into two layers allows us to use the bottom layer, pure swarm, isolated, for tasks where the collective behaviour of the swarm emerges from interactions between agents and the environment.

First, three basic tasks of swarm robotics are analysed: aggregation, coordinated movement (flocking) and dispersion. Then, two specific behaviours for a swarm of robots are defined. The first of these shows a behaviour for the location of resources in the environment, where the robots are able to locate the most promising resource in unknown environments, with noise and with different resource areas. The second case defines a behaviour capable of detecting, monitoring, covering and marking the perimeter of a marine oil spill.

Resum

La robòtica d' eixam és un camp d' investigació dins de l'àrea de la robòtica que estudia la coordinació d' un gran nombre de robots simples. Aquest camp d' investigació s' inspira en el comportament observat en els insectes socials, els quals són grans exemples de com un gran nombre d' individus simples poden interactuar per a crear sistemes intel·ligents col·lectius. En aquestos sistemes el comportament col·lectiu emergix de forma auto-organitzada a partir de les interaccions entre els individus i entre els individus i l' entorn.

De la mateixa manera, en els sistemes d' eixam artificials, la intel·ligència és una propietat emergent a partir del comportament global de l' eixam. En aquestos sistemes, l' eixam és capaç de dur a terme tasques, de manera global, que estan fora de les capacitats d' un robot individual.

Els sistemes robòtics d' eixam han de complir una sèrie de característiques que els diferencien d' altres sistemes multi-robótics. Algunes d' estes característiques són compartides amb els sistemes multi-agent, per tant, aquestos poden ser una alternativa per a la construcció d' aquest tipus de sistemes.

En aquest treball es presenten un conjunt de comportaments col·lectius per a eixams artificials. Per a dotar d' un marc d' implantació a aquestos comportaments, es definix un model d' arquitectura híbrida per al control d' un eixam de robots basada en un sistema multi-agent. Una de les característiques bàsiques d' aquest model és que la divisió en dos capes permet utilitzar la capa inferior, d' eixam pur, de manera aïllada, per a tasques on el comportament col·lectiu de l' eixam emergix únicament a partir de les interaccions entre els agents i l' entorn.

En primer lloc, s' analitzen tres tasques bàsiques de robòtica d' eixam: agregació, moviment coordinat (*flocking*) i dispersió. Posteriorment, es definixen dos comportaments concrets per a un eixam de robots. El primer d' aquestos mostra un comportament per a la localització de recursos a l' entorn, on els robots són capaços de localitzar la font de recursos més prometedora en entorns desconeguts, amb soroll i amb diverses fonts de recursos. El

segon cas definix un comportament capaç de detectar, monitoritzar, cobrir i marcar el perímetre d' un abocament petrolífer marítim.

Índice general

Índice de figuras

Capítulo 1

Introducción

En las últimas décadas se ha producido un gran incremento en las investigaciones y actividades que tratan de conseguir sistemas multi-robóticos capaces de llevar a cabo un comportamiento colectivo [CFK97]. La evolución natural de este paradigma ha sido el uso de grandes grupos de robots simples que son capaces de llevar a cabo tareas que un único robot no podría conseguir [MC08]. Aunque en la bibliografía es posible encontrar diferentes términos que hacen referencia al uso de grupos de robots como robótica colectiva, robótica distribuida, colonias de robots, etc. [Sah05], actualmente estos grupos de robots se denominan, generalmente, enjambres.

En este capítulo se realizará una breve introducción a la robótica de enjambre, se definirá el término y se resaltarán sus aportaciones y limitaciones. Por último se establecerán los objetivos de esta tesis y se describirá su hilo argumental.

Cabe destacar que este trabajo se enmarca dentro de las tareas desarrolladas por el grupo de investigación I3A (Informática Industrial e Inteligencia Artificial) de la Universidad de Alicante y es el resultado de su experiencia dentro de los campos de la robótica móvil y robótica de enjambre, sistemas inteligentes, visión artificial, gráficos y realidad virtual y aumentada. Destacamos de manera especial el siguiente proyecto de investigación, marco en el cual se ha desarrollado el trabajo aquí presentado: **"Adaptación y Autoensamblado Morfológico en Sistemas de Enjambre Inteligentes" (TIN2009-10581)** concedido por el Ministerio de Ciencia e Innovación.

1.1. Robótica de enjambre

La robótica de enjambre es una nueva aproximación a la coordinación de un gran número de robots relativamente simples. De manera que estos robots pueden llevar a cabo tareas colectivas que están fuera de las capacidades de un único robot [DS04]. Este campo de investigación se inspira en el comportamiento observado en los insectos sociales, como hormigas, termitas, avispas o abejas, los cuales son grandes ejemplos de cómo un gran número de individuos simples pueden interactuar para crear sistemas inteligentes colectivos.

Estos conjuntos de insectos demuestran tres características importantes para la robótica: robustez, flexibilidad y escalabilidad. Por lo tanto, la robótica de enjambre se basa en la metáfora de las colonias de insectos sociales para enfatizar aspectos como el control descentralizado, la comunicación limitada entre agentes, el uso de información local, la aparición de un comportamiento global y la robustez [DTG+05].

En [Sah05] se propone la siguiente definición para este término: "La robótica de enjambre es el estudio de cómo un gran número de agentes relativamente simples pueden ser diseñados de manera que el comportamiento colectivo deseado emerja a partir de las interacciones locales entre los agentes y entre los agentes y el entorno".

Es importante destacar, tras esta definición, que los sistemas multi-robóticos de enjambre deben cumplir una serie de características que los diferencian del resto de sistemas multi-robóticos [Sah05]:

- Los robots que forman el enjambre deben ser robots autónomos situados en el entorno.
- El enjambre debe estar formado por un gran número de robots que podrán dividirse en grupos más pequeños de robots homogéneos.
- Los robots serán relativamente simples.
- Los robots deberán disponer de sensores locales y capacidades de comunicación limitadas.

Estas características aseguran que la coordinación entre los robots será distribuida, que el sistema tendrá una alta tolerancia a fallos ya que debido a la redundancia de robots cada uno de los agentes que forman el sistema es prescindible pudiendo ser sustituido por otro agente, y que el sistema será escalable, permitiendo añadir o eliminar más agentes según la tarea lo requiera.

Una particularidad de este tipo de sistemas es que han acumulado una serie de problemas estándar. Es posible encontrar trabajos donde se utiliza la robótica de enjambre para solucionar problemas de agregación, dispersión, auto-ensamblado, búsquedas, movimientos coordinados, transporte cooperativo, etc.

1.1.1. Robótica de enjambre y sistemas multi-agente

Aunque en la bibliografía sobre robótica de enjambre podemos encontrar muchas alusiones a la palabra *agente*, como se puede ver en la misma definición del término, éstas hacen referencia a los agentes como entidades robóticas, no como miembros de un sistema multi-agente.

Si tenemos en cuenta la definición de sistema multi-agente que se puede encontrar en [HLW09] "un sistema multi-agente se define como una red débilmente acoplada de agentes autónomos que trabajan conjuntamente para resolver problemas que están por encima de las capacidades o conocimientos de un único robot" y, además, que los agentes tienen información incompleta (un punto de vista limitado), no existe un sistema de control global, los datos están descentralizados y la computación es asíncrona [CM10] [Wei99]; es posible ver que los sistemas multi-agente comparten muchas características con la robótica de enjambre y, por lo tanto, podrían ser una buena alternativa para modelar este tipo de sistemas.

Sin embargo, los sistemas multi-agente no se pueden trasladar fácilmente a la robótica de enjambre, debido a una serie de características particulares de los enjambres robóticos: su naturaleza física, los mecanismos de comunicación distribuidos y las estructuras de control [HTM09].

No obstante, los sistemas multi-agente pueden aportar una serie de ventajas a los sistemas robóticos de enjambre como [Wei99]:

- Mayor rapidez en la resolución del problema por el paralelismo y la distribución.
- Mayor integración de sistemas, permitiendo la interconexión e interoperación de sistemas ya existentes.
- Distribución de datos y control.
- Mejora de la eficiencia computacional, fiabilidad, escalabilidad, robustez, mantenimiento, flexibilidad y reutilización.

1.1.2. ¿Qué aporta la robótica de enjambre?

La robótica de enjambre es un campo de investigación dentro del área de la robótica que enfatiza el uso de muchos robots simples en lugar de un único robot complejo. De esta manera y teniendo en cuenta esta característica, todos los individuos del grupo son relativamente sencillos y tienen capacidades limitadas. La cooperación entre ellos es la que les permitirá llevar a cabo tareas complejas de manera colectiva.

Si comparamos estos sistemas con un robot tradicional, que puede necesitar componentes complejos o una gran capacidad de procesamiento para conseguir llevar a cabo las tareas que tiene asignadas, el uso de robots simples puede conllevar una serie de ventajas:

- Los sistemas robóticos de enjambre son **tolerantes a fallos** y **robustos**, ya que pueden seguir en funcionamiento ante el fallo de alguna unidad.
- Son sistemas con una alta **escalabilidad**, donde el tamaño del enjambre puede ser aumentado o disminuido según la tarea lo requiera.
- Enfatizan el uso del **paralelismo**, donde un conjunto de robots puede llevar a cabo una tarea más rápidamente que un único robot, descomponiendo la tarea en subtareas y ejecutándolas de manera concurrente.
- Y, por regla general, los sistemas de enjambre suelen ser más **económicos**. El coste de estos sistemas permite que sea factible la reparación o sustitución de estos equipos.

La robótica de enjambre puede ser utilizada en muchos campos. Actualmente es un área de investigación activa dentro de la robótica. Los investigadores se han inspirado en los sistemas biológicos para proponer una gran variedad de aplicaciones enfocadas a diferentes ámbitos como el industrial, militar, médico, etc. En [VDS04] indican que estos sistemas son adecuados para dominios discretos, carentes de recursos, distribuidos, descentralizados y dinámicos.

Precisamente por las características de la robótica de enjambre, estos sistemas son apropiados para utilizarse en:

Tareas donde se debe cubrir una región. Son sistemas apropiados para tareas donde se debe comprobar o realizar un seguimiento del estado del entorno.

Tareas centradas en alguna entidad del entorno. Son sistemas adecuados para tareas donde se debe buscar una fuente de recursos o recuperación de objetos.

Tareas demasiado peligrosas. Son sistemas robustos y tolerantes a fallos donde el fallo o la pérdida de algunos agentes no supondría el fallo de todo el sistema.

Tareas que deben aumentar o disminuir en el tiempo. Son sistemas escalables que permiten la agregación o eliminación de nuevos individuos.

Tareas que requieren redundancia. La redundancia es una característica implícita en estos sistemas formados por un elevado número de robots homogéneos.

1.1.3. Limitaciones de la robótica de enjambre

El objetivo de la robótica de enjambre es estudiar el diseño de robots de forma que, a partir de las interacciones entre los robots y entre los robots y el entorno, emerjan patrones de comportamiento colectivo. Es decir, que a partir de comportamientos individuales relativamente sencillos, se obtenga un comportamiento complejo a nivel global.

El hecho de que el comportamiento global emerja de manera auto-organizada puede conllevar principalmente dos problemas. Por un lado, el sistema puede ser impredecible, llegando a obtener resultados no esperados y, por otro lado, el sistema puede ser incontrolable, siendo difícil ejercer el control sobre el enjambre.

La robótica de enjambre intenta desarrollar sistemas robustos, escalables, flexibles, distribuidos, sin ningún control centralizado y formados por un elevado número de robots autónomos relativamente sencillos e ineficientes con sensores locales y mecanismos de comunicación limitados.

Todas estas características de los sistemas de enjambre hacen que en muchas ocasiones sea complicado el diseño de tareas complejas. Puede ser necesario algún tipo de deliberación sobre la tarea que se está llevando a cabo, o algún mecanismo de nivel superior para poder unir la información de los diferentes robots que forman el enjambre.

Veámoslo con un ejemplo: imaginemos una tarea de mapeado. Un enjambre robótico puede realizar mediante comportamientos individuales sencillos la cobertura del área a estudiar, pero será necesario algún mecanismo para

poder fusionar la información percibida por todos los robots y combinarla en un único mapa.

1.2. Motivación y objetivos

Como se ha comentado anteriormente, la robótica de enjambre puede ser aplicada en diferentes ámbitos (militar, industrial, médico, etc.) y puede ayudar en el desarrollo de muchas tareas (exploración, monitorización, detección, etc.). Una de las ventajas derivadas de las características de estos sistemas es que pueden facilitar el trabajo y ayudar a las personas en tareas o entornos peligrosos, ya que cada uno de sus componentes es prescindible.

Aunque la robótica de enjambre es, hoy en día, un campo de investigación activo, la mayoría de trabajos realizados se centran en dos aspectos: por un lado, muchos trabajos están focalizados en el diseño y creación de los robots físicos, en conseguir equipos lo más reducidos, eficientes y económicamente viables para formar un gran grupo de robots; por otro lado, muchos trabajos se centran en el diseño y desarrollo de comportamientos y conductas para una tarea concreta. La mayoría de trabajos que encontramos sobre definición de arquitecturas de control robóticas, se centran en la arquitectura para un único robot, para dotarlo de las capacidades necesarias para realizar una tarea, sin entrar a definir la arquitectura a nivel global. Así mismo, es posible encontrar muchos trabajos donde se proponen arquitecturas robóticas y multi-robóticas distribuidas, pero éstas no son específicas para la robótica de enjambre y, por tanto, no cumplen las características establecidas en este campo.

El objetivo principal de este trabajo es definir el modelado de un enjambre de robots inteligentes, aportando un conjunto de comportamientos para el control de un enjambre robótico, de manera que éste sea capaz de llevar a cabo comportamientos de manera colectiva que están fuera del alcance de un único robot.

Con esta consideración en mente, este objetivo principal puede ser dividido en los siguientes objetivos:

- **Estudiar los problemas susceptibles de ser desarrollados mediante robótica de enjambre.** La finalidad de este estudio es conocer los diferentes campos donde es posible aplicar la robótica de enjambre, para qué tipo de problemas son adecuados este tipo de sistemas y cuáles son sus limitaciones.

- **Estudiar las arquitecturas robóticas actuales.** La realización de

un estudio de las arquitecturas robóticas y multi-robóticas existentes permitirá conocer las capacidades y limitaciones de estas arquitecturas para la robótica de enjambre y proporcionará una guía para describir los componentes del sistema, su organización y la interacción entre éstos.

- **Estudiar la aplicación de sistemas multi-agente a la robótica de enjambre.** El uso de los sistemas multi-agente para desarrollar aplicaciones de robótica de enjambre no es directo por algunos requerimientos de este campo, como el control totalmente distribuido o la tolerancia a fallos. Este estudio permitirá conocer si es viable el uso de sistemas multi-agente en robótica de enjambre.
- **Definir el modelo de una arquitectura robótica adecuada a la robótica de enjambre.** El modelo de arquitectura definido deberá cumplir por un lado, las características que exige la robótica de enjambre (control descentralizado, autonomía de los individuos, capacidades limitadas o información local) y, por otro, deberá ser capaz de recopilar la información del enjambre, planificar, deliberar y generar información de más alto nivel para el usuario.
- **Diseñar comportamientos básicos de robótica de enjambre.** Inicialmente se definirán comportamientos estándar de la robótica de enjambre como agregación, dispersión o movimiento coordinado.
- **Definir comportamientos avanzados para la robótica de enjambre.** Se definirán comportamientos avanzados de robótica de enjambre diseñados para la realización de tareas concretas: localización de recursos y marcado del perímetro de un vertido petrolífero.

1.3. Organización de la tesis

Esta tesis se estructura de la siguiente manera:

- En el capítulo 2 se realiza una revisión del estado del arte de la robótica de enjambre. Se define la robótica de enjambre, se indican las características que debe tener un sistema multi-robótico para considerarse un sistema de enjambre y se presentan problemas susceptibles de desarrollarse con este tipo de sistemas.
- En el capítulo 3 se muestran las características comunes de la robótica de enjambre y los sistemas multi-agente. Se presenta JADE como

plataforma de agentes y se indican los problemas que pueden surgir al intentar aplicar un sistema multi-agente, como JADE, a la robótica de enjambre. Se presentan los mecanismos disponibles para solventar estos problemas como la replicación del *AMS* y la persistencia del *DF*.

- En el capítulo 4 se revisan las arquitecturas robóticas y multi-robóticas existentes. Se realiza un análisis comparativo de estas arquitecturas, analizando el cumplimiento de las características que exige la robótica de enjambre. Además, en este capítulo se define un modelo de arquitectura híbrida para el control de un sistema de enjambre basada en un sistema multi-agente. Este modelo complementa la capa de enjambre puro con una capa deliberativa de más alto nivel.

- En el capítulo 5 se describen tres comportamientos básicos de la robótica de enjambre: agregación, *flocking* (movimiento coordinado) y dispersión (exploración de un área).

- En el capítulo 6 se presenta un comportamiento de localización de recursos. Se define un comportamiento microscópico y macroscópico del enjambre para una tarea donde el enjambre debe ser capaz de encontrar las fuentes de recurso del entorno.

- En el capítulo 7 se presenta un comportamiento para la detección de vertidos. Se define el modelo microscópico y macroscópico para una tarea donde el enjambre es capaz de localizar y marcar el perímetro de un vertido de petróleo.

- Por último, en el capítulo 8, se desarrollan las conclusiones del trabajo realizado y las líneas futuras sobre esta investigación.

Capítulo 2

Robótica de enjambre

La robótica de enjambre es una nueva aproximación a la coordinación de un gran número de robots relativamente simples. De manera que estos robots pueden llevar a cabo tareas colectivas que están fuera de las capacidades de un único robot [DS04]. La robótica de enjambre encuentra sus raíces teóricas en estudios realizados sobre sociedades de animales. Concretamente, este campo se inspira en el comportamiento observado en los insectos sociales como hormigas, termitas, avispas o abejas, los cuales son grandes ejemplos de cómo un gran número de individuos simples pueden interactuar para crear sistemas inteligentes colectivos [Sah05]. Los insectos sociales son conocidos por coordinar sus acciones para conseguir realizar tareas que serían imposibles para un único individuo. Por ejemplo, las termitas son capaces de construir montículos complejos de gran tamaño, las hormigas son capaces de organizar impresionantes batidas en busca de comida, además de poder transportar colectivamente grandes presas (en comparación con su tamaño), etc.

Aunque se dé el caso de ruido en el entorno, errores en el procesamiento de la información, fallos en la ejecución de algunas tareas o la inexistencia de información global, los insectos sociales son capaces de llevar a cabo con éxito algunas tareas a nivel colectivo. Basándose en la metáfora de los insectos sociales, la robótica de enjambre enfatiza aspectos como el control descentralizado, la comunicación limitada entre agentes, información local, la aparición de comportamiento global y la robustez [DTG$^+$05].

Estos insectos sociales demuestran tres características deseables de un sistema multi-robótico: robustez, flexibilidad y escalabilidad [BS07].

Robustez: se define como el grado por el cual un sistema puede seguir en funcionamiento ante la presencia de fallos parciales o condiciones

anormales. La robustez requiere que el enjambre robótico continúe en funcionamiento, aunque sea con bajo rendimiento, debido a fallos en alguno de los agentes o ante alteraciones del entorno. Esta robustez se puede deber a diferentes factores [Sah05]:

- Redundancia en el sistema: la pérdida o fallo de operación de uno de los individuos puede ser compensado por otro individuo. Esto hace que cada uno de los individuos sea prescindible.
- Coordinación descentralizada: la destrucción de alguna parte del sistema no tiene que detener su funcionamiento. La coordinación es una propiedad emergente del sistema global.
- Simplicidad de los individuos: si lo comparamos con un único sistema complejo que puede realizar la misma tarea, en un sistema robótico de enjambre los individuos son más simples, haciendo que sean menos propensos a fallos.
- Multiplicidad de sensores: la distribución de los sensores utilizando un gran número de individuos puede incrementar la tolerancia al ruido del sistema.

Flexibilidad: se define como la capacidad del sistema para adaptarse a entornos nuevos o diferentes o a cambios en los requerimientos. La flexibilidad requiere que el sistema robótico de enjambre tenga la habilidad de generar soluciones modulares a diferentes tareas. Se puede observar un claro ejemplo en las colonias de hormigas, cada uno de los individuos puede formar parte en diferentes tareas como búsqueda de comida, recuperación de alguna presa, formación de cadenas, etc. Los sistemas de robótica de enjambre deben tener también la flexibilidad de ofrecer soluciones a las tareas utilizando diferentes estrategias de coordinación en respuesta a cambios en el entorno [Sah05].

Se deben diferenciar estas dos características anteriores, robustez y flexibilidad, en el sentido en el que si el problema cambia, el sistema debe ser flexible (no robusto) para llevar a cabo un comportamiento más adecuado para resolver el nuevo problema.

Los sistemas biológicos tienen este nivel de flexibilidad y pueden cambiar fácilmente sus comportamientos cuando los problemas cambian. Por ejemplo, las hormigas son muy flexibles y pueden solucionar problemas en tareas de búsqueda de comida o formación de la cadena con sus propios organismos de auto-organización.

Escalabilidad: se define como la habilidad para expandir un mecanismo auto-organizado para dar soporte a un número mayor o menor de individuos sin un impacto considerable en el funcionamiento. La escalabilidad requiere que el sistema robótico de enjambre sea capaz de operar con tamaños de grupo diferentes. Es decir, los mecanismos de coordinación que aseguran el funcionamiento del enjambre deben verse poco influenciados por cambios en el tamaño del grupo.

2.1. Como campo de investigación

Con los avances tecnológicos actuales, la robótica de enjambre se está volviendo un campo de investigación cada vez más viable. Ha atraído a un número significativo de grupos de investigación que están actualmente contribuyendo en este campo. Además, se han realizado varios proyectos con el objetivo de desarrollar y controlar un gran número de robots relativamente simples.

En la Unión Europea, la CEC (Comisión de las Comunidades Europeas) ha financiado estudios de robótica de enjambre a través del programa FET (*Future and Emerging Technologies*). En Estados Unidos, la agencia DARPA (*Defense Advanced Research Projects Agency*) ha financiado igualmente estudios en este campo a través del programa SDR (*Software for Distributed Robotics*).

Proyecto Swarm-Bots [DTG$^+$05]

Swarm-Bots es un proyecto financiado por la Comisión Europea a través del programa *Future and Emerging Technologies*. Este proyecto finalizó en 2005 y tenía como objetivo el desarrollo de un sistema robótico llamado swarm-bot, basado en técnicas de robótica de enjambre. El trabajo llevado a cabo en este proyecto está directamente inspirado en el comportamiento colectivo de las colonias de insectos sociales y otras sociedades animales. Se centra principalmente en el estudio de mecanismos que están presentes en los procesos de auto-organización o auto-ensamblado en agentes artificiales autónomos. Para alcanzar estos objetivos, en este proyecto se ha desarrollado un nuevo robot llamado s-bot. Estos robots son capaces de llevar a cabo comportamientos tanto individuales como colectivos con la interacción entre los diferentes robots y entre los robots y el entorno.

Cada s-bot tiene la capacidad de conectarse y desconectarse de otros robots. Estos enganches físicos se utilizan para formar un swarm-bot (estructura formada por varios robots enganchados entre sí) capaz de resolver

Figura 2.1: Configuración del swarm-bot para cruzar un hueco. Imagen obtenida de http://www.swarm-bots.org (consultada en noviembre de 2013).

problemas que no pueden ser resueltos por un único s-bot. La conexión física entre robots es esencial para resolver algunas tareas colectivas estudiadas en el proyecto como engancharse en forma de cadena para mover un objeto pesado, para evitar obstáculos como agujeros y algunos pasos en terrenos desiguales, etc. Se puede observar una unión entre estos robots en la figura 2.1.

Proyecto Pheromone Robotics [PEH05]

Este proyecto, financiado por la agencia DARPA (*Defense Advanced Research Projects Agency*), tiene como principal objetivo proporcionar una aproximación escalable y robusta para la coordinación de las acciones de un gran número de robots. Se intentan alcanzar buenos resultados en tareas como vigilancia, reconocimiento, detección de peligros, seguimiento de senderos, transporte de sustancias peligrosas, etc.

Para conseguir este objetivo se centran en el desarrollo de nuevos conceptos para la coordinación e interacción de un gran número de robots. Se basan en técnicas utilizadas por ciertos insectos como hormigas o termitas para mostrar una colaboración emergente. Concretamente, este proyecto se inspira en los marcadores químicos que estos insectos utilizan para la comunicación y coordinación para crear las *feromonas virtuales*. Estas feromonas virtuales facilitan la comunicación y coordinación y requieren poco procesamiento en los robots. Para crear estas feromonas utilizan luces y sensores direccionales situados sobre los robots. En la figura 2.2 se observa el enjambre en funcionamiento.

Figura 2.2: Pherobot Swarm. Imagen obtenida de http://www.pherobot.com (consultada en noviembre de 2013).

Figura 2.3: Robot i-swarm. Imagen obtenida de http://www.i-swarm.org (consultada en noviembre de 2013).

Proyecto I-Swarm [SSB$^+$05]

El proyecto I-Swarm ha sido financiado por la Comisión Europea. Este proyecto finalizó en 2008 y tenía como principal objetivo la construcción del primer enjambre artificial a gran escala, con un tamaño aproximado de 1000 micro-robots, con una dimensión de $2 \times 2 \times 1mm^3$ (figura 2.3). Cada uno de estos robots estaría equipado con inteligencia limitada y pre-racional. Asimismo, el enjambre consistiría en un gran número de robots heterogéneos, que se diferenciarían en el tipo de sensores y en los manipuladores. Este enjambre se utilizaría para una gran variedad de aplicaciones incluyendo, micro-ensamblado y tareas de limpieza. Además, también tendría aplicaciones en el campo de la medicina o biología.

Este proyecto intenta combinar micro-robótica, sistemas distribuidos y auto-organización en enjambres biológicos.

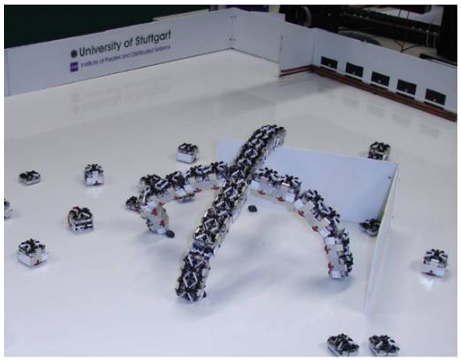


Figura 2.4: Symbrion project. Agregación de robots formando un organismo más complejo. Imagen obtenida de http://www.symbrion.eu (consultada en noviembre de 2013).

Proyectos SYMBRION y REPLICATOR

El proyecto SYMBRION *Symbiotic Evolutionary Robot Organisms* y el proyecto REPLICATOR *Robotic Evolutionary Self-Programming and Self-Assembling Organisms* ambos financiados por la Comisión Europea (*European Research Community*) surgen a raiz del proyecto anterior I-Swarm. Estos proyectos comenzaron en el año 2008 y terminarán en el 2013. El objetivo principal de estos proyectos es la investigación y desarrollo de principios de adaptación y evolución de organismos multi-robóticos inspirados en la biología y en paradigmas de computación avanzados. Estos organismos robóticos consisten en un enjambre de gran escala, que pueden ensamblarse unos con otros y compartir energía y recursos computacionales como si fueran un único individuo artificial. En la figura 2.4 puede observarse una agregación de estos robots.

Proyecto CoCoRo

Este proyecto liderado por la Universidad de Graz y financiado por la Comisión Europea se inició a mediados del año 2011 y terminará en 2014. Tiene como objetivo la creación de un enjambre autónomo de robots cognitivos e interactivos. En el proyecto se desarrollará un enjambre de vehículos autónomos submarinos capaces de interactuar entre ellos y desarrollar tareas de monitorización ecológica, búsqueda, mantenimiento, exploración y recogida de recursos en hábitats subacuáticos. La figura 2.5 muestra una imagen del prototipo.

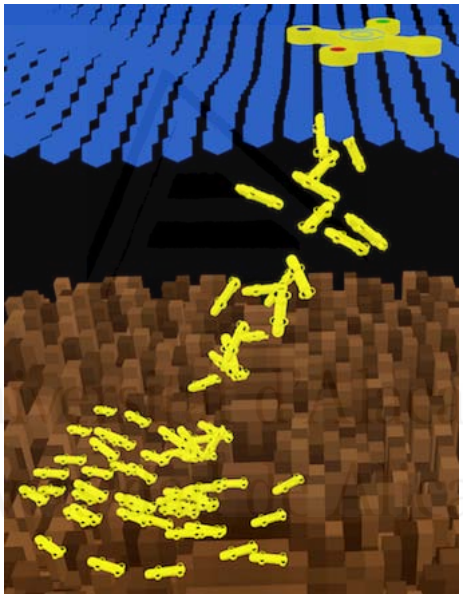


Figura 2.5: CoCoRo project. Imagen del prototipo de enjambre que puede encontrarse en http://cocoro.uni-graz.at/ (consultada en noviembre de 2013).

2.2. Evolución y definición del término

El término *enjambre* (*swarm*) se utiliza actualmente en muchos campos, podemos encontrar acepciones en biología, ingeniería, computación, etc. Referido a robótica, el término *swarm intelligence* fue acuñado por Gerardo Beni [Ben05] a finales de la década de los 80.

Anteriormente se utilizaba el término *robots celulares* para hacer referencia a grupos de robots que podían trabajar como células de un organismo para llevar a cabo tareas más complejas. Unos años más tarde se discutió que aunque el concepto era interesante, este término no era adecuado para describir esos *grupos* de robots. Además, se daba el caso de que esos grupos de robots no eran únicamente un grupo, sino que tenían unas características especiales, que de hecho se encuentran en los enjambres de insectos, como por ejemplo, control descentralizado, falta de sincronización, miembros simples y todos los miembros (casi) idénticos. Por lo tanto, *enjambre* era un término más apropiado para describir este tipo de grupos [Ben05].

Junto a este término han aparecido otros términos como control de enjambre, optimización de enjambre, etc.

El término *inteligencia de enjambre* (*swarm intelligence*) ha sido cada vez más utilizado durante los últimos 15 años. Una de las razones fundamentales es que un robot de enjambre inteligente es una máquina verdaderamente avanzada y la realización de estos sistemas es aún hoy en día una meta por alcanzar. A lo largo de estos últimos años, ha surgido el término *robótica de enjambre* para hacer referencia a la aplicación de la inteligencia de enjambre a los sistemas multi-robóticos, que se centra en la integración física y las interacciones entre los agentes y los agentes y el entorno [SGBT08].

En [Sah05] se propone la siguiente definición para este término: "La robótica de enjambre es el estudio de cómo un gran número de agentes relativamente simples pueden ser diseñados de manera que un comportamiento colectivo deseado emerja a partir de las interacciones locales entre los agentes y entre los agentes y el entorno".

Al ser un término relativamente nuevo, únicamente en lo que se refiere a multi-robótica podemos encontrar una gran variedad de términos acuñando diferentes aspectos de este campo como *robótica colectiva*, *robótica distribuida*, *colonias de robots*, etc. Algunos de estos términos son imprecisos, haciendo referencia además en algunos casos a los mismos significados. Por este motivo en [Sah05] se definen una serie de criterios que distinguen a los sistemas multi-robóticos de enjambre respecto a otros sistemas multi-robóticos:

- **Robots autónomos.** Los agentes que forman parte del enjambre deben estar situados físicamente en el entorno, deben poder interactuar físicamente con el entorno y con otros agentes y además ser autónomos. Las redes sensoriales formadas por varios sensores distribuidos pero sin capacidades de actuación físicas no son consideradas como robótica de enjambre.

- **Elevado número de robots.** Las investigaciones en este campo deben ser relevantes para la coordinación del enjambre robótico, por lo tanto, los estudios que se centran en el control de un pequeño grupo de robots sin posibilidades de escalabilidad no serían considerados robótica de enjambre. Aunque el coste, mantenimiento y experimentación con un grupo elevado de robots sea un gran obstáculo y los experimentos se lleven a cabo con un enjambre formado por 10-20 agentes, se debe tener siempre en cuenta la escalabilidad.

- **Pequeños grupos de robots homogéneos.** Los sistemas robóticos de enjambre deben estar formados por varios grupos de robots homogéneos, de manera que el número de robots en cada grupo sea lo suficientemente elevado para mantener las características del sistema respecto a robustez y flexibilidad. Aunque un sistema podría estar formado por un grupo totalmente homogéneo. Además, hay que tener en cuenta que los estudios sobre equipos de fútbol robóticos no se considerarían robótica de enjambre ya que cada uno de los jugadores tiene un rol diferente que se carga a cada robot antes de cada prueba.

- **Robots relativamente simples o ineficientes.** Los robots que forman parte del sistema robótico de enjambre deben ser relativamente simples e ineficientes por sí mismos en la tarea que deben llevar a cabo. Los robots deben tener dificultades en la tarea que deben realizar si la realizaran por sí mismos, de manera que la cooperación de un grupo de robots sea esencial. El uso de un grupo de robots debe mejorar el rendimiento y la robustez del sistema para desarrollar dicha tarea.

- **Robots con sensores locales y capacidades de comunicación limitadas.** Los robots del sistema deben tener sensores locales y habilidades de comunicación limitadas. Estas dificultades aseguran que la coordinación entre los robots sea distribuida. De hecho, el uso de canales de comunicación globales pueden tener como resultado que los mecanismos de comunicación no sean escalables.

2.3. Fuentes de inspiración y problemas estándar

Estudios en sistemas físicos y biológicos han revelado que existen gran cantidad de mecanismos de coordinación en los sistemas naturales que pueden actuar como fuentes de inspiración para la coordinación de sistemas robóticos de enjambre [SGBT08]. La robótica de enjambre se inspira en muchos campos de investigación y muchos conceptos concretos. Algunos de éstos se destacan en [GGT07] como: descentralización, estigmergia, auto-organización, emergencia, retroalimentación positiva y negativa, fluctuaciones y bifurcaciones. Dos de los principales mecanismos de coordinación son: la auto-organización y la estigmergia [SGBT08].

La estigmergia es, probablemente, la primera explicación teórica seria sobre la organización de los insectos sociales. Es un concepto introducido por el biólogo francés Pierre-Paul Grassé para designar la comunicación que hacen algunos animales a través de la interacción con el entorno. Por ejemplo, el uso de feromonas que hacen las hormigas. La estigmergia es interesante para la robótica de enjambre ya que proporciona un mecanismo de comunicación local, distribuido y escalable [SGBT08].

Es posible distinguir distintos tipos de estigmergia: estigmergia basada en marcadores (los agentes depositan marcadores especiales en el entorno), estigmergia sematectónica (los agentes basan sus acciones en el estado actual de la solución), estigmergia cuantitativa (las señales del entorno son cantidades escalares individuales, análogo a un campo de potencial) o estigmergia cualitativa (si forman un conjunto de opciones discretas).

Entre todos estos conceptos, el concepto más importante para la robótica de enjambre es el estudio de la auto-organización [GGT07]. En [VDS04] se define la propia palabra enjambrado utilizando este término, como ”la auto-organización práctica de múltiples entidades a través de interacciones locales”. Este mismo autor define la auto-organización como ”el proceso que reduce la entropía de un sistema sin intervención externa”.

Centrándonos más en robótica de enjambre, en [Sah05] se define la auto-organización como “el proceso en el que un patrón a nivel global del sistema aparece únicamente a partir de múltiples interacciones entre los componentes de nivel inferior del sistema”. En este sentido, la robótica de enjambre puede ser considerada como la ingeniería y uso de la auto-organización en enjambres móviles integrados físicamente.

En los insectos sociales podemos encontrar claros ejemplos de comportamiento de auto-organización, como en las colonias de hormigas y termitas, en las bandadas de pájaros, en los bancos de peces o incluso en los rebaños de ovejas [Liu08], como se puede observar en la figura 2.6.

Figura 2.6: Ejemplos de comportamientos de colonias de animales (hormigas, aves y peces). De izquierda a derecha imágenes obtenidas de: http://phys.org, http://animales.org.es y http://teachwild.org.au (consultadas en noviembre de 2013).

La auto-organización se basa en cuatro ingredientes básicos [Liu08]: retroalimentación positiva, retroalimentación negativa, balance entre explotación y exploración e interacciones múltiples.

Los estudios de auto-organización que se han llevado a cabo en sistemas biológicos demuestran que la retroalimentación, tanto positiva como negativa, de las interacciones entre los individuos es esencial para este fenómeno. La retroalimentación positiva, normalmente, promueve la creación de estructuras y aumenta las fluctuaciones del sistema. Por el contrario, la retroalimentación negativa funciona como un mecanismo regulador para contrarrestar la retroalimentación positiva y ayudar a estabilizar el patrón colectivo. Los sistemas auto-organizados están conducidos por sus propios componentes, los cuales interactúan basándose en información local sin ninguna referencia del sistema como un todo.

En estos sistemas, la retroalimentación positiva se genera, normalmente, a través de comportamientos auto-catalíticos (los cambios causados en el entorno del sistema de enjambre por la ejecución de un comportamiento, incrementan el desarrollo de ese comportamiento). El efecto de bola de nieve provocado por el ciclo de retroalimentación positiva es contrarrestado por un mecanismo de retroalimentación negativa, que, normalmente, es producto de una reducción o agotamiento de las fuentes físicas en el sistema o entorno.

Otros campos de investigación que estudian los principios de la auto-organización en sistemas biológicos, desarrollan modelos indicando, de manera simplificada, las interacciones en el mundo y los mecanismos de comportamiento abstractos en estos individuos. Algunas investigaciones en robótica de enjambre se han inspirado en estos modelos de la auto-organización de insectos y animales sociales.

Aunque la auto-organización se entiende como la inspiración más im-

portante de los sistemas robóticos de enjambre, existen otras fuentes de inspiración [Sah05].

- Organismos unicelulares. En algunas especies de organismos unicelulares, como bacterias, mycobacterias o amebas, se han observado interesantes ejemplos de coordinación. Estos organismos actúan de manera independiente cuando las condiciones son favorables (gran cantidad de alimento, ausencia de antibióticos, etc.) pero demuestran comportamientos coordinados cuando las condiciones son desfavorables.

- Auto-ensamblado de materiales. El auto-ensamblado se define como "la organización autónoma de componentes en patrones o estructuras sin intervención [externa]". El auto-ensamblado es interesante a diferentes escalas: molecular, nanoescala, etc. Pero de este campo es posible obtener dos ideas interesantes para la robótica de enjambre. En primer lugar el uso de plantillas puede servir como guía para estos procesos y de esta manera reducir los defectos en el auto-ensamblado. En segundo lugar es posible el uso de agentes catalíticos. Ambas ideas tienen el potencial para mejorar la formación de patrones en sistemas robóticos de enjambre.

Problemas estándar

En los últimos años la robótica de enjambre ha acumulado una serie de problemas estándar que aparecen recurrentemente en los estudios.

- Un grupo de problemas está basado en la formación de patrones:

  Agregación: es la agrupación auto-organizada de los individuos de un enjambre sin utilizar indicaciones del entorno. En sistemas robóticos de enjambre puede ser considerado como uno de los comportamientos fundamentales que actúa como precursor de otros comportamientos más complejos (*flocking*, formación de patrones o auto-ensamblado) [SGBT08][CM11]. En la naturaleza, los comportamientos de agregación, observados en diferentes organismos desde bacterias a insectos sociales y mamíferos, incrementan la posibilidad de supervivencia del enjambre en entornos hostiles [SS07]. En [CM11] se presenta un modelo para la agregación auto-organizada de robots inspirado en el comportamiento de las cucarachas, en las cuales el comportamiento de agregación emerge únicamente de las interacciones locales entre los individuos, y

la probabilidad de unirse o dejar un grupo está en función del tamaño de éste. En [SBS07] estudian dos aproximaciones para el comportamiento de agregación auto-organizada en un enjambre de robots: métodos de evolución y control probabilístico. El controlador para el método de evolución es una red neuronal con una única capa, con 12 neuronas de entrada y 3 de salida que corresponden con los actuadores del robot. Para el control probabilístico proponen un comportamiento de agregación genérico como una combinación de 4 comportamientos básicos: evitación de obstáculos, aproximación, repulsión y espera, divididos en dos capas de acuerdo a una arquitectura de subsunción. Los 3 últimos comportamientos básicos se combinan con una máquina de estados finita probabilística. En [BS09] también utilizan un autómata de estados finitos probabilísticos muy simple, con únicamente dos estados (deambular y espera). En [YT07] se inspiran en el comportamiento de agregación observable en los bancos de peces. Proponen una arquitectura motor-schema para definir este comportamiento basada en tres comportamientos: repulsivo, paralelo y de atracción.

Dispersión: puede ser considerado como lo contrario a la agregación. El desafío de este problema es la obtención de una distribución uniforme de los robots en el espacio de manera que maximice el área cubierta. La cobertura de un área es una tarea importante para varias aplicaciones como búsqueda y rescate, localización y mapeado, exploración del área, vigilancia y monitoreo. Además, algunas de éstas incluyen tareas peligrosas (campos de minas) o terrenos inaccesibles (exploración planetaria) [JaGAJ10]. En [LLC11] se presenta un algoritmo descentralizado para un enjambre de robots con el objetivo de cubrir un área para monitorizar la polución medioambiental. Este algoritmo permite a los robots localizar la fuente de polución incluso con la existencia de varias fuentes. Está inspirado en la toma de decisiones humanas y en su movimiento hacia los puntos de interés, donde las personas deciden la dirección a la que dirigirse basándose en la situación actual del entorno y tratando de acercarse al punto de interés. Los robots se guían unicamente por la intensidad de la polución y por la densidad de robots en el área. En [SH08] se describen tres algoritmos para la dispersión de un enjambre de robots. Los dos primeros algoritmos se basan en el movimiento de las moléculas

de gas, con pequeñas modificaciones. En el tercer algoritmo cada robot emite una señal, y utilizan un método de optimización de enjambres de partículas para buscar los valores mínimos de las señales emitidas. En [UTS07] se presenta un algoritmo de dispersión basado en la intensidad de la señal inalámbrica. Es similar a los métodos basados en campos de potencial, de manera que si un robot percibe una señal más fuerte que un determinado umbral tendrá un comportamiento de repulsión, si la señal es menor que un umbral, tendrá un comportamiento de atracción, lo que asegura la conectividad del enjambre. Una técnica similar que utiliza la intensidad de señal inalámbrica para determinar la distancia a otros robots para tareas de dispersión se puede encontrar en [LG06].

Auto-ensamblado: este problema se refiere a la creación organizada de estructuras a partir de las conexiones físicas entre los robots que forman el enjambre. Es interesante porque puede proporcionar capacidades y funcionalidades adicionales [GBMD06] como cruzar un hueco o subir un escalón. En [GBMD06] presentan un estudio del proyecto *swarm-bots*, donde cada robot tiene un anillo circular para permitir la conexión en dos terceras partes del robot. Esta conexión se realiza mediante una pinza rígida. Este sistema permite que no sea necesaria una alineación precisa para realizar el auto-ensamblado. En [YLC+06], sin embargo, presentan un mecanismo para el auto-ensamblado basado en un electroimán y dos abrazaderas mecánicas, donde los robots deben estar debidamente alineados. [WLTW10] define un algoritmo de control para el auto-ensamblado donde se utiliza una tabla de estado para representar la configuración de la estructura robótica que se desea formar. En [COD08] definen un lenguaje *SWARMORPH-script* de alto nivel, que permite la descripción de las reglas que gobiernan el proceso de crecimiento de la morfología conjunta. En [Kla07] se define una gramática para describir el protocolo de interacciones entre los robots que forman la topología del sistema auto-ensamblado.

Es posible encontrar otros comportamientos basados en la formación de patrones: auto-organización, mapeado del entorno, creación de gradientes, etc.

- Un segundo grupo de problemas se centra en alguna entidad del entorno:

Foraging (búsqueda de un elemento): este comportamiento se inspira en el comportamiento de las hormigas que buscan fuentes de alimento alrededor del hormiguero. El desafío de este problema radica en encontrar la estrategia de búsqueda óptima para maximizar la ratio de comida. Existen varios escenarios con un grupo de robots que deben encontrar y recolectar (o transportar) ítems a algún lugar definido [HW07]. En la bibliografía, tomando prestada la terminología de la biología, el punto de partida de los robots es llamado "nido" y los objetos o ítems a recolectar son llamados "presas" o "comida" [CD07]. Esta tarea ha sido ampliamente estudiada con una gran variedad de condiciones y arquitecturas [LG02]. En [HW07] presentan un modelo para un enjambre homogéneo de robots que utilizan como medio de comunicación feromonas virtuales. En [LW09], sin embargo, presentan un modelo para un grupo de robots heterogéneo, donde introducen dos variables que los robots son capaces de ajustar, tiempo de búsqueda y tiempo de descanso, basándose en tres indicadores: interno (éxito de la tarea), ambiental (colisiones) y social (éxito de los otros robots en la tarea). Se han realizado también estudios sobre la eficiencia de los enjambres. En [LA11] describen un modelo de conducta para mejorar la eficiencia de energía del enjambre evitando colisiones entre los robots, basándose para ello en la división de roles y del espacio de búsqueda. En [CD07] tratan de mejorar la eficiencia maximizando la energía del grupo para una tarea de *multi-foraging* donde los robots deben recolectar dos ítems diferentes. Los robots pierden energía durante la exploración y la ganan cuando un ítem es recolectado correctamente. En [LG02] definen un modelo matemático para entender cuantitativamente los efectos de las interferencias (colisiones) entre los robots que surgen al trabajar en un mismo entorno. En este estudio especifican la existencia de un número óptimo de robots que maximizan el rendimiento del grupo.

Es posible encontrar otros problemas basados en un elemento del entorno: búsqueda de una meta, *homing* (instinto de volver a casa o búsqueda del hogar), recuperación de un objeto, etc.

- Y otro tipo de problemas tratan un comportamiento más complejo del grupo:

Movimiento coordinado (*flocking*): este problema hace referen-

cia a la coordinación del movimiento de un grupo de robots, de manera que sean capaces de realizar un movimiento suave, evitando obstáculos del entorno. Hace referencia a la manera en que las poblaciones de animales como pájaros, peces o insectos se mueven conjuntamente, donde el comportamiento global emerge como una consecuencia de las interacciones locales entre los miembros vecinos [AAC10]. Muchos de los trabajos relacionados con el movimiento coordinado de un grupo de robots [MSC11], [NGMMH08], [AAC10], [TCGS08a], [SFT+11] hacen referencia al trabajo desarrollado por Reynolds en 1987 [Rey87]. Éste muestra que el movimiento coordinado se puede obtener utilizando tres comportamientos simples: separación (los individuos tratan de mantener una distancia mínima con sus vecinos), cohesión (los individuos tratan de mantenerse junto a sus vecinos) y alineación (los individuos tratan de mantener la velocidad adecuándola a la de sus vecinos). En [MSC11] presentan un algoritmo basado en las lecturas activas y pasivas de los sensores de infrarrojos para detectar objetos u otros robots. Discretizan los campos de los sensores en sectores y utilizan diferentes umbrales de distancia para cohesión y separación, surgiendo de manera emergente la alineación. Utilizan una arquitectura de subsunción simple con cuatro posibles decisiones (evitar obstáculos, separación, cohesión y alineación) para comprobar los valores devueltos por los sensores con los umbrales predefinidos. De esta manera no utilizan ningún tipo de comunicación, memoria, ni información global. En [NGMMH08] también utilizan los sensores de infrarrojos que indican la distancia y ángulo de los robots vecinos, e información de un sistema de orientación situado en la base de cada robot que permite conocer su posición y orientación global. De esta manera, cada robot reacciona a los objetos detectados por los infrarrojos y tendrá un comportamiento de cohesión o de separación dependiendo de la distancia medida. En [AAC10] utilizan una arquitectura de control basada en el comportamiento, donde definen una serie de comportamientos muy simples con unas prioridades establecidas. Estos comportamientos únicamente dependen de información local (posición) relativa a los robots vecinos. En [SFT+11] presentan una aproximación para un grupo de robots heterogéneo donde no todos los robots tienen los mismos comportamientos. Todos los robots tienen los comportamientos de separación y cohesión pero solo unos pocos tienen el compor-

tamiento de alineación.

Transporte cooperativo: el desafío de este problema es la coordinación de un grupo de robots para permitir que colaboren en el transporte de un objeto demasiado pesado para un único robot. Este problema es una evolución del problema anterior. Está inspirado en el comportamiento de las hormigas que son capaces de cooperar para recuperar grandes presas. Normalmente, una hormiga encuentra una presa, trata de moverla y si no tiene éxito recluta a otras compañeras. Cuando un grupo de hormigas intenta mover una gran presa, cambian su posición y alineación hasta que pueden moverla [KB00].

Es posible encontrar otros comportamientos: minería (recogida de restos), conducir o guiar (*shepherding*), etc.

No es una lista exhaustiva, dado que existen otras muchas tareas susceptibles de resolver con robótica de enjambre. Entre ellas, otras tareas de robótica general como la evitación de obstáculos y la navegación por cualquier terreno, que también es aplicable a enjambres robóticos.

En [VDS04] se indica que es apropiado utilizar enjambres para dominios: discretos, carentes de recursos, distribuidos, descentralizados y dinámicos.

Centrándonos en tareas concretas donde la robótica de enjambre sería aplicable, [Sah05] muestra una lista de estas tareas indicando, además, ejemplos de problemas reales:

- Tareas donde se debe cubrir una región. Los sistemas robóticos de enjambre son sistemas distribuidos y deben ser apropiados para tareas que deban comprobar el estado del entorno. Por ejemplo, el monitoreo del entorno de un lago. Además, la capacidad de los sistemas robóticos de enjambre respecto a tener los sensores distribuidos, pueden proporcionar un sistema de vigilancia para la detección inmediata de eventos peligrosos, como por ejemplo, un escape o fuga accidental de un elemento químico. Los sistemas robóticos de enjambre tienen la capacidad de situarse en la localización exacta del problema, lo que permite al sistema una mejor localización e identificación del problema. Además, por otro lado, el enjambre podría formar un parche para bloquear ese vertido.

- Tareas demasiado peligrosas. Cada uno de los agentes que forman un sistema de enjambre es prescindible haciendo que el sistema sea adecuado para dominios que conlleven tareas arriesgadas. Por ejemplo, crear un paso en un campo minado.

- Tareas que deben aumentarse o disminuir en el tiempo. Los sistemas robóticos de enjambre tienen la capacidad de aumentar o disminuir el tiempo de la tarea que están llevando a cabo según sea necesario. Por ejemplo, el escape de petróleo en un barco hundido, puede verse incrementado si el tanque del barco se rompe más. Un sistema robótico adecuado para esta tarea puede auto-ensamblarse para contener el escape inicial, pero si este escape aumenta pueden incorporarse más robots para contener la nueva cantidad de escape.

- Tareas que requieren redundancia. La robustez de un sistema robótico de enjambre proviene de la redundancia implícita en el enjambre. Esta redundancia permite que el sistema se degrade lentamente haciendo que el sistema sea menos propenso a fallos catastróficos. Por ejemplo, los sistemas robóticos de enjambre pueden crear redes de comunicación dinámicas. Estas redes son muy robustas dado que los nodos de comunicación pueden reconfigurarse ante el fallo de algún nodo.

2.4. Clasificación de los estudios

En [BS07] se presenta una amplia revisión de los estudios que se han realizado en robótica de enjambre. En este artículo realizan una clasificación que representa las diferentes direcciones de la investigación en este campo. Clasifican los estudios existentes en robótica de enjambre en diferentes ejes: modelado, diseño de comportamientos, comunicación y problemas. En el apartado anterior ya se han descrito los problemas estándar de la robótica de enjambre, en este apartado se realizará una clasificación de los estudios realizados basándose en los ejes anteriores.

2.4.1. Modelado del sistema

En robótica de enjambre los robots son normalmente simples, pero a partir de la interacción de los robots que forman el enjambre o de los robots y el entorno pueden aparecer comportamientos colectivos complejos. Para diseñar y optimizar los comportamientos de cada robot individual y por tanto, para obtener las propiedades deseadas del enjambre, uno de los objetivos es entender el efecto de los parámetros individuales en el funcionamiento global del grupo.

El modelado y análisis matemático ofrecen una alternativa a los experimentos y simulaciones con robots reales, que pueden ser costosos y conllevar mucho tiempo. “Utilizando el análisis matemático es posible estudiar los

sistemas robóticos de enjambre rápida y eficientemente y predecir su comportamiento a largo término. Adicionalmente, el análisis matemático puede ser utilizado para seleccionar parámetros que optimicen el funcionamiento del grupo y prevenir inestabilidades."[LMG05].

El modelado del sistema puede ayudar a entender mejor el sistema que se está investigando. Este punto es muy importante, sobre todo en robótica de enjambre, donde el coste de los robots físicos y los riesgos que éstos pueden sufrir en los experimentos hacen más importante el modelado y la simulación, siempre teniendo en cuenta la posible existencia de variaciones entre los resultados simulados y los reales.

En la clasificación citada anteriormente diferencian cuatro tipos de modelado: basado en sensores, microscópico, macroscópico y autómatas celulares. El modelado de autómatas celulares se ha utilizado comunmente para definir fenómenos naturales y aunque es posible encontrar algún caso aplicado a la robótica de enjambre, éste no es muy común.

Basado en sensores

Este método de modelado fue uno de los primeros y ha sido uno de los más utilizados para modelar experimentos robóticos. Es un método que utiliza el modelado de los sensores, de los actuadores de los robots y de los objetos del entorno como componentes principales del sistema. Una vez modelados estos componentes principales se modelan las interacciones de los robots con el entorno y las interacciones entre los propios robots.

Este tipo de modelado, utilizado principalmente para el desarrollo de simulaciones de sistemas robóticos, permite realizar experimentos simulados y obtener resultados acordes con aquellos que se obtendrían con sistemas reales. En estos sistemas se debe llegar a un balance entre realismo y simplicidad, ya que los modelos e interacciones deben ser lo más realistas posibles para ser útiles, pero al mismo tiempo deben ser lo más simples posibles para evitar sistemas demasiado complejos [SGBT08].

En [TCGS08b] presentan la plataforma *Kobot* como entorno de simulación para robótica de enjambre donde modelan la estructura del robot y sus sensores (infrarrojos y sistema de orientación) y actuadores. A partir de este modelado, definen un comportamiento de *flocking* auto-organizado. Esta misma plataforma (*Kobot*) la utilizan en [UTS07] modelando un sistema de señales inalámbricas para el modelado de un comportamiento de dispersión basado en estas señales. En [NGMMH08] presentan un algoritmo para el comportamiento de *flocking* utilizando reglas de control locales basadas en el modelado de los sensores de infrarrojos e información sobre la

orientación global del robot. En [LEF09] definen un modelo del robot y de un comportamiento básico para conseguir formar una formación triangular. Y en [XC08] definen un modelo del sistema de enjambre para un método de control general para la formación del enjambre.

Modelado microscópico

El modelado microscópico define los experimentos robóticos modelando cada robot y sus interacciones matemáticamente. En este método, los comportamientos de los robots se definen como estados y la transición entre estados se debe a eventos internos del robot y externos del entorno. Los estados de los robots y las transiciones entre estos estados es modelada analíticamente.

Un caso especial de modelado microscópico son los modelos microscópicos probabilísticos. Estos modelos asignan probabilidades a las transiciones entre las acciones de los robots. El comportamiento del sistema y el ruido del entorno se pueden integrar fácilmente en estos modelos. La idea principal de los modelos microscópicos probabilísticos es describir las interacciones entre los robots y entre los robots y el entorno como una serie de eventos estocásticos.

Las ventajas de los modelos microscópicos probabilísticos sobre las simulaciones basadas en sensores son dobles. En primer lugar, son necesarios únicamente unos pocos parámetros y capturar únicamente los parámetros del sistema que juegan un papel relevante. En segundo lugar, el modelo microscópico puede ser utilizado como una herramienta de desarrollo para elaborar predicciones del comportamiento [Liu08].

En [Ham10a] presentan un método para modelar sistemas de multipartículas basado en leyes físicas. Definen un modelo microscópico basándose en la ecuación de Langevin para describir la trayectoria como un proceso estocástico. En [CM07] definen un modelo microscópico para la cobertura de un grafo distribuido. Este modelo lo aplican a un problema de cobertura de un área utilizando como caso de estudio un sistema de inspección de una serie de elementos alineados en una estructura, de manera que los vértices del grafo representan los bordes de los elementos a inspeccionar. Este mismo equipo representa en [CM11] el modelo individual de cada robot utilizando un sistema dinámico de Markov. Utilizan una cadena de Markov para describir la dinámica de cada individuo en un comportamiento de agregación auto-organizado.

Modelado macroscópico

La mayor diferencia entre el modelado microscópico y el macroscópico es la granularidad de los modelos. Mientras que los modelos microscópicos modelan los experimentos modelando cada robot, los modelos macroscópicos modelan el comportamiento total del sistema directamente. En este tipo de modelado el comportamiento del sistema se define con diferentes ecuaciones y cada uno de los estados del sistema representa la media de robots en un estado concreto en un instante dado.

Por regla general, los modelos macroscópicos suelen obtener resultados de manera más rápida que los microscópicos, aunque estos últimos nos permiten obtener fluctuaciones de los experimentos. Es decir, mientras que los modelos macroscópicos permiten obtener rápidamente el comportamiento global del sistema, los modelos microscópicos permiten obtener, más lentamente, un comportamiento más preciso.

En robótica de enjambre también son utilizados los modelos macroscópicos probabilísticos para poder tratar de manera simple el ruido. Estos modelos asignan probabilidades a las transiciones entre los estados del sistema.

Es posible encontrar muchos ejemplos de definiciones de modelos macroscópicos en el campo de la robótica de enjambre. En [LL10] se propone un modelo macroscópico para estudiar la dinámica de un sistema robótico de enjambre, compuesto por ecuaciones diferenciales con retardo que describen las transiciones entre estados del sistema. En [CM11] desarrollan un modelo macroscópico probabilístico para modelar la dinámica de una proceso de agregación auto-organizado observado en artrópodos gregarios. En [SS07] proponen un modelo para predecir la distribución final de agregados en un comportamiento de agregación auto-organizada genérica. En [LW10] presentan un modelo macroscópico probabilístico para una tarea de *foraging* colectivo, con el fin de entender el efecto de los parámetros individuales en el comportamiento colectivo del enjambre. En [SAPR11] se presenta un modelo macroscópico para determinar el comportamiento del enjambre ante un problema de localización de recursos.

Tanto los modelos probabilísticos microscópicos como los macroscópicos se basan en dos suposiciones [Liu08]:

- El cumplimiento de las propiedades de Markov (o la mitad de las propiedades de Markov). Por ejemplo, el estado futuro del robot depende únicamente de su estado presente y del tiempo que está en ese estado. Esta suposición es cierta cuando los robots deciden sus acciones futuras basándose únicamente en su estado, la entrada de los sensores y la

acción que están llevando a cabo actualmente. Por lo tanto, los robots pueden ser representados como procesos de Markov y el sistema puede ser modelado como una máquina de estados finita probabilística.

- La metodología se basa en la suposición de que la cobertura del entorno por el grupo de robots es espacialmente uniforme, y las estrategias de bajo nivel de los robots no tienen un papel crítico en la métrica del sistema. De hecho, encontrar una descripción matemática apropiada para las probabilidades de transición es el principal desafío para la aplicación de ambos modelos tanto microscópico como macroscópico. Esta segunda suposición se vuelve particularmnte útil para el cálculo de las probabilidades de transición de los robots. En este caso, las probabilidades de eventos básicos, por ejemplo, detectar un objeto, depende únicamente de las consideraciones geométricas y éstas son dadas por el ratio total del área del objeto en relación con al área total del entorno.

A pesar de que algunos estudios han tenido éxito en la aplicación de estos modelos, estas dos suposiciones pueden ser una limitación para la aplicación de los modelos probabilísticos tanto microscópicos como macroscópicos.

2.4.2. Diseño de comportamientos

En la clasificación nombrada anteriormente categorizan los trabajos realizados en robótica de enjambre en tres grupos, tomando como base la adaptación conductual de los controladores del robot. De esta manera, separan los trabajos como no adaptativos, aprendizaje y evolución.

Los artículos de robótica de enjambre que trabajan con aprendizaje se basan en aprendizaje por refuerzo. A pesar de las ventajas del aprendizaje por refuerzo, éste puede tener una serie de problemas al adaptarlo a la robótica de enjambre [BS07]. Por un lado, las propiedades de convergencia teórica del aprendizaje por refuerzo requiere un gran número de pruebas de aprendizaje que son difíciles de llevar a cabo con enjambres de robots reales. Además, el espacio de búsqueda podría ser muy grande en este tipo de problemas, que requerirían muchas etapas para poder converger en resultados aceptables. Por otro lado, se debe tener en cuenta el ruido que puede surgir en la lectura de los sensores, en las acciones de los actuadores y en la interacción entre los robots que hace que el sistema sea más impredecible, lo que puede conllevar la pérdida de convergencia de algunos algoritmos de aprendizaje por refuerzo.

Los estudios dentro de este campo que imitan la evolución utilizan, mayoritariamente, algoritmos genéticos. Estos algoritmos se utilizan para evolucionar los pesos de redes neuronales para obtener el comportamiento deseado.

La gran mayoría de estudios que podemos encontrar sobre robótica de enjambre se centran en el diseño de comportamientos no adaptativos y se podrían clasificar en diferentes subcategorías según la metodología utilizada. Tres de las categorías donde podemos encontrar una gran cantidad de trabajos son: arquitectura de subsunción, autómatas de estados finitos probabilísticos y redes neuronales.

Arquitectura de subsunción La arquitectura de subsunción es una de las características más clásicas y distinguidas de la robótica basada en comportamientos. Esta arquitectura permite la coordinación eficiente de comportamientos utilizando para ello mecanismos de inhibición simples entre los comportamientos, y la construcción incremental de los controladores del robot considerando cada comportamiento como un módulo separado que puede inhibir otros comportamientos.

Varios estudios utilizan, para robótica de enjambre, una arquitectura de subsunción donde en la capa inferior se ejecuta un comportamiento de evitación de obstáculos y en las capas superiores se ejecuta el comportamiento deseado del enjambre. Por ejemplo, en [UTS07] utilizan una arquitectura de subsunción compuesta por dos capas. En la inferior definen un comportamiento de evitación de obstáculos y en la capa superior el robot ejecuta un comportamiento de dispersión. En [SBS07] definen también una arquitectura de subsunción con esta estructura, pero en la capa superior implementan un comportamiento de agregación.

En [MSC11], sin embargo, definen una arquitectura de subsunción para un comportamiento de *flocking* con cuatro comportamientos: evitación de obstáculos, separación, cohesión y alineamiento.

Autómatas de estados finitos probabilísticos Los autómatas de estados finitos probabilísticos ($AEFP$) son una manera de representar sistemas dinámicos con espacios de estados finitos. En un autómata probabilístico, las transiciones entre los estados del sistema se desencadenan con una cierta probabilidad. Generalmente, se modelan los comportamientos de un robot como estados y se definen las transiciones entre estados como efecto de alguna entrada externa o probabilidad.

El comportamiento de agregación definido en [SBS07] está implementado como una combinación de tres comportamientos básicos (aproximación, espera, repulsión) en una máquina de estados finita probabilística. En [LA11] definen una máquina de estados finita más compleja para un comportamiento de *foraging* cooperativo con nueve estados. En [LW10] utilizan también un $AEFP$ para un comportamiento de *foraging* colectivo.

Redes neuronales Las redes neuronales son mecanismos de aprendizaje muy potentes. Están inspiradas en el sistema nervioso de los humanos. En los estudios de robótica de enjambre que utilizan redes neuronales se puede encontrar el uso de algoritmos genéticos para evolucionar los pesos de la red neuronal.

En [SBS07] comparan un comportamiento de agregación definido con un autómata de estados finitos con un controlador evolutivo definido como una red neuronal con una única capa, con 12 entradas (que corresponden con los sensores) y 3 salidas (para controlar los actuadores). Utilizan un algoritmo genético para evolucionar la red neuronal. En [GD09] utilizan una red neuronal con 6 nodos ocultos, 6 entradas y 6 salidas, cuyos pesos se sintetizan con un algoritmo evolutivo para una tarea de transporte cooperativo.

2.4.3. Mecanismos de comunicación

Entre las características que enfatiza la robótica de enjambre se encuentra la capacidad de comunicación limitada de los agentes y el uso de sensores locales. Teniendo en cuenta estas características, es posible clasificar los mecanismos de comunicación utilizados en los enjambres en tres categorías:

- Interacción vía sensores. Es el tipo de comunicación más utilizado en robótica de enjambre. Es una comunicación sencilla y limitada. En este tipo de comunicación los robots reciben la información a través de sus propios sensores, detectando y diferenciando otros robots y objetos. Esta característica, muy importante en animales, se conoce como reconocimiento del parentesco (*kin recognition*). Con esta característica los animales son capaces de comportarse de manera diferente ante sus parentescos, trabajar conjuntamente para llevar a cabo varias tareas y protegerse así mismos contra sus enemigos. Muchos de los estudios de robótica de enjambre utilizan este mecanismo como medio de comunicación para agruparse, formar cadenas o arrastrar un objeto de manera cooperativa.

- Interacción vía entorno. En este tipo de comunicación los robots utilizan el entorno como un medio de comunicación. Se conoce como estigmergia. En biología existen ejemplos bien conocidos de este tipo de comunicación, por ejemplo, la comunicación vía feromonas de las hormigas. Aunque el esquema de comunicación no es trivial debido a la dificultad de crear entornos que permitan este tipo de comunicación entre los agentes, algunos estudios utilizan este mecanismo de comunicación. En estos casos utilizan únicamente una simulación del esquema de comunicación a través de mecanismos de comunicación inalámbricos como radiofrecuencia o infrarrojos. Por ejemplo, en el proyecto Pheromone Robotics [PEH05] utilizan feromonas virtuales creadas mediante luces y sensores direccionales. En [PR07] utilizan esta técnica para que un enjambre sea capaz de seguir a un líder. El líder emite feromonas químicas y el resto de agentes están dotados de sensores de detección especiales para detectar estas sustancias químicas.

- Interacción vía comunicación. Es un tipo de comunicación que puede no cumplir las propias características de la robótica de enjambre y además, puede reducir la escalabilidad y flexibilidad del sistema. Este mecanismo implica comunicación explícita con otros robots mediante mensajes broadcast o comunicación directa uno a uno. Por ejemplo, en [DDPG11] se presenta un comportamiento de cooperación autoorganizado con dos tipos de robots, robots terrestres y robots aéreos. Los robots aéreos buscan los puntos objetivos y lo transmiten a través de señal inalámbrica mediante *broadcast* a los robots terrestres. En [HSWN10], sin embargo, utilizan comunicación directa entre robots para un comportamiento de *foraging*.

2.5. Resumen

La robótica de enjambre se centra en la coordinación de un gran número de robots simples. De manera que el comportamiento colectivo emerge a partir de las interacciones entre los robots y entre los robots y el entorno. Este campo se inspira en las sociedades de insectos sociales como hormigas, termitas o avispas para enfatizar aspectos como la robustez, flexibilidad y escalabilidad.

Para que un sistema multi-robótico sea considerado un sistema de enjambre debe cumplir una serie de criterios: el sistema debe estar formado por un gran número de robots autónomos, éstos deben ser relativamente simples o ineficientes y tener capacidades de comunicación limitadas. Estos

requerimientos aseguran que la coordinación entre los robots será distribuida y que no existirá un control centralizado.

Las características que poseen los sistemas de enjambre hacen que estos sean apropiados para tareas donde se debe cubrir una región, tareas peligrosas, tareas que deben aumentar o disminuir en el tiempo y tareas que requieran redundancia.

En este capítulo se ha realizado una revisión del estado del arte de la robótica de enjambre. Se ha definido el término, se han enumerado las características que debe cumplir un sistema para ser considerado un sistema robótico de enjambre, así como los problemas susceptibles de ser desarrollados con este tipo de sistemas. Además, se han descrito algunos proyectos de investigación en el campo y se ha realizado una clasificación de los estudios existentes.

Una vez definida la robótica de enjambre, sus posibles aplicaciones y los requerimientos que debe cumplir un sistema para adecuarse a la robótica de enjambre, en el siguiente capítulo se analizarán las características comunes de los sistemas robóticos de enjambre y los sistemas multi-agente y la posibilidad de utilizar estos últimos en este campo.

Capítulo 3

Agentes, $FIPA$ y robótica de enjambre

En el capítulo anterior se han visto las características de la robótica de enjambre: control descentralizado y la autonomía de los individuos, las capacidades de comunicación limitadas y la información local, la aparición de comportamiento global y la robustez. En este capítulo se describirán las características de los sistemas multi-agente que enfatizan la autonomía de los agentes, el trabajo conjunto, la descentralización de los datos y la inexistencia de un control global. Se verá que muchas de estas características son compartidas por ambos. Además, la robótica de enjambre se inspira en el comportamiento observado en los insectos sociales, como hormigas, termitas o avispas y por otro lado, los sistemas multi-agente parecen ser una metáfora natural para entender y construir sistemas sociales artificiales donde los agentes no sólo se pueden comunicar sino también coordinar, cooperar y negociar. Por todo lo anterior, en este capítulo se verá cómo los sistemas multi-agente podrían ser una buena alternativa para modelar los sistemas robóticos de enjambre. Esta asociación no es inmediata por el control distribuido y la tolerancia a fallos en la robótica de enjambre, donde un fallo en el sistema no puede poner en peligro el funcionamiento de todo el enjambre.

Se describirá brevemente el modelo de referencia para la gestión de agentes especificado por *FIPA* y seguido por *JADE*, donde se especifica que el sistema de control de agentes, entre otros componentes, debe contener: una plataforma (AP) que proporciona la infraestructura donde los agentes se despliegan; el servicio de páginas amarillas (DF) donde los agentes publican sus servicios; y, obligatoriamente, el servicio de páginas blancas (AMS) que hace el papel de controlador llevando a cabo las tareas de gestión de la pla-

taforma. *JADE* para seguir esta especificación define el uso de contenedores de agentes. De manera que el punto de partida de una plataforma será la creación de un *contenedor principal* que mantendrá el AMS y el DF, y se encargará del registro de otros contenedores y de todos los agentes de la plataforma.

Se verá cómo el uso de este contenedor principal que contiene dos servicios clave para la plataforma, el AMS y el DF, puede ser un punto débil del sistema ya que el fallo en este contenedor puede suponer que todo el sistema quede inoperativo. Por lo tanto, podría ser necesario algún mecanismo que evite la caída del sistema ante un fallo en el contenedor principal. Se describirán las dos características que proporciona *JADE* para asegurar el funcionamiento de la plataforma en estos casos: la replicación del contenedor principal y la persistencia del DF.

Aunque el uso de estas características de *JADE* aseguran que el contenedor principal no será un punto débil del sistema y que no comprometería el funcionamiento de todo el enjambre ante un fallo, *JADE* está pensado para trabajar con una conectividad completa y continua entre contenedores. En robótica de enjambre, donde es posible que los robots se desplieguen por zonas de pérdida de cobertura o, incluso, donde no haya una infraestructura de red disponible, se debe tener en cuenta la posibilidad de pérdidas de conexión en los agentes. En este caso, se propondrá una solución al problema con el uso de redes *MANET* que proporcionan las funcionalidades de red necesarias sin la ayuda de ningún administrador central y sin utilizar infraestructuras de red fija, utilizando los módulos de transmisión inalámbrica *XBee* junto con el protocolo de enrutamiento para establecer la comunicación entre nodos *DigiMesh*.

3.1. Agentes y *FIPA*

3.1.1. Agentes inteligentes y sistemas multi-agente

Una de las áreas de conocimiento donde se ha estudiado extensamente el control de un número elevado de robots son los sistemas multi-agente. Éstos son sistemas compuestos por múltiples elementos de computación interactivos llamados agentes. Los agentes tienen dos habilidades importantes. Por un lado, son capaces de realizar acciones de manera autónoma (de decidir por sí mismos qué necesitan hacer para alcanzar sus objetivos) y por otro lado, son capaces de interactuar con otros agentes (no simplemente intercambiando datos, sino entablando una conversación análoga a las actividades sociales que realizamos cada día: cooperación, coordinación y negociación)

[Woo02]. Ya en 1995 en [WJ95] se describían las capacidades que se esperan de un agente:

- **Autonomía**: los agentes operan sin la intervención de humanos u otros y tienen algún tipo de control sobre sus acciones y estado interno.
- **Habilidades sociales**: los agentes interactúan con otros agentes a través de algún tipo de *lenguaje* para satisfacer sus objetivos.
- **Reactividad**: los agentes son capaces de percibir el entorno y responder rápidamente a los cambios que ocurren en éste para satisfacer sus objetivos.
- **Pro-actividad**: los agentes no actúan únicamente en respuesta al entorno, sino que son capaces de tomar la iniciativa llevando a cabo un comportamiento dirigido hacia el objetivo.

Cuando se trabaja con un número muy elevado de agentes, es una tarea difícil gestionar cada uno de ellos individualmente. Sería más conveniente tratarlos de manera colectiva, como una sociedad de agentes o un sistema multi-agente.

Tomamos como base la definición de sistema multi-agente que se puede encontrar en [HLW09]: "un sistema multi-agente se define como una red débilmente acoplada de agentes autónomos que trabajan conjuntamente para resolver problemas que están por encima de las capacidades o conocimientos de un único robot". Estos agentes tienen una serie de características [CM10] [Wei99]:

- Los agentes tienen información incompleta del entorno. Tienen un punto de vista limitado.
- No existe un sistema de control global.
- Los datos están descentralizados.
- La computación es asíncrona.

Por todas estas características los sistemas multi-agente parecen una metáfora natural para entender y construir un amplio rango de lo que se llaman sistemas sociales artificiales [Woo02]. Durante muchos años los estudios en inteligencia artificial han estado centrados en individuos, pero una buena parte de lo que nos hace únicos como especie son nuestras habilidades sociales, donde no sólo nos podemos comunicar, sino también coordinar, cooperar

y negociar. Estas habilidades sociales, como se ha visto en el capítulo 2, se pueden encontrar en otras especies como hormigas, abejas, termitas, etc. Por todo ello para la construcción de un sistema multi-agente se deben realizar una serie de preguntas [Woo02]:

- ¿Cómo puede emerger la cooperación en sociedades de agentes con intereses personales?
- ¿Qué lenguaje pueden utilizar los agentes para comunicarse?
- ¿Cómo pueden reconocer los agentes cuando sus creencias, metas o acciones entran en conflicto y cómo pueden alcanzar acuerdos con otros agentes?
- ¿Cómo pueden agentes autónomos coordinar sus actividades para alcanzar los objetivos de manera cooperativa?

Todas estas preguntas han de ser analizadas para llevar a cabo el diseño y desarrollo de un sistema multi-agente donde los individuos deben cooperar, coordinar y negociar. Por otro lado, aunque el desarrollo de estos sistemas sea una tarea más complicada, los sistemas multi-agente, como sistemas distribuidos, tienen la capacidad de ofrecer una serie de características deseables en un sistema [Wei99]:

- **Velocidad y eficiencia**: los agentes pueden operar de manera asíncrona y en paralelo, permitiendo un incremento de la velocidad global del sistema.
- **Robustez y fiabilidad**: el fallo de uno o varios agentes no necesariamente dejarán el sistema inservible, debido a que otros agentes disponibles en el sistema podrán sustituirlos.
- **Escalabilidad y flexibilidad**: el sistema podrá ser adaptado a un problema mayor añadiendo nuevos agentes, sin afectar al funcionamiento de otros agentes.
- **Costes**: puede ser más económico que un sistema centralizado, ya que puede estar formado por subsistemas simples de bajo coste.
- **Desarrollo y reusabilidad**: cada agente puede ser desarrollado de manera separada (desde cero o tomando como base otro agente). El sistema global puede ser probado y mantenido más fácilmente, y puede ser posible el uso y configuración de los agentes para otro propósito.

3.1.2. Robótica de enjambre y sistemas multi-agente

Como se ha visto en el capítulo 2 la robótica de enjambre es un tipo de robótica basada en la interacción de muchos robots simples, de manera que los robots pueden llevar a cabo tareas que están por encima de las capacidades de un único robot. La robótica de enjambre enfatiza aspectos como el control descentralizado, la comunicación limitada, la información local, la aparición de comportamiento global y la robustez; e indica las siguientes características:

- Los individuos que forman el enjambre deben ser autónomos.
- El grupo debe estar formado por un elevado número de robots o por pequeños grupos de robots homogéneos.
- Los individuos deben ser relativamente simples o ineficientes.
- Los individuos deben tener sensores locales y capacidades de comunicación limitadas obligando a una coordinación distribuida sin utilizar mecanismos globales de comunicación.
- Son apropiados para entornos discretos, carentes de recursos, distribuidos, descentralizados y dinámicos.

Este tipo de sistemas disfrutan de muchas ventajas, como la alta tolerancia a fallos (por la redundancia en el sistema de individuos simples o por la coordinación descentralizada), la flexibilidad (por la capacidad de adaptación a cambios en los requerimientos) y la escalabilidad (por la posibilidad de ampliar o reducir el número de robots de manera transparente). Aunque también pueden tener muchos problemas para su implantación real, como la dificultad para predecir los comportamientos macroscópicos a partir de los microscópicos o el diseño de tareas complejas.

Muchas de estas características de la robótica de enjambre son compartidas por los sistemas multi-agente. En la propia definición de éstos últimos se indica que los agentes deben ser autónomos, y que deben ser capaces de realizar tareas que están por encima de las capacidades de un único agente. Además, también se indica que los datos del sistema deben ser descentralizados y que la información del entorno de la que disponen los agentes es incompleta. Por otro lado, la robótica de enjambre se inspira en el comportamiento de los insectos sociales y los sistemas multi-agente, según Woolridge en [Woo02], parecen una metáfora natural para entender y construir un rango amplio de lo que se llaman sistemas sociales artificiales.

En [Woo02] se señala que existen una serie de factores que hacen que una aproximación basada en agentes sea apropiada:

- El entorno es abierto, o al menos, dinámico, incierto o complejo.
- Cuando la idea de los agentes sea una metáfora natural del problema a tratar.
- Existe una distribución de los datos y el control.

En base a todas estas características comunes entre los sistemas multi-agente y la robótica de enjambre, el uso de los primeros podría ser una buena alternativa para modelar este tipo de sistemas.

Formalizar un sistema de enjambre utilizando agentes puede aportar múltiples ventajas como:

- Mayor rapidez en la resolución del problema por el paralelismo y la distribución.
- Mayor integración de sistemas, permitiendo la interconexión e interoperación de sistemas ya existentes.
- Distribución de datos y control.
- Mejora de la eficiencia computacional, fiabilidad, escalabilidad, mantenimiento, flexibilidad y reutilización.

Sin embargo, nos encontramos con que los sistemas multi-agente no se pueden trasladar fácilmente a la robótica de enjambre debido a una serie de características particulares de los enjambres robóticos: su naturaleza física, los mecanismos de comunicación distribuidos y las estructuras de control [HTM09]. De hecho, si analizamos algunas arquitecturas basadas en sistemas muti-agente, nos encontramos con que efectivamente, características fundamentales en los sistemas de enjambre, como el control totalmente distribuido o la tolerancia a fallos prácticamente no se tienen en cuenta. En el punto 3.2 se analizarán más detalladamente estos problemas.

3.1.3. JADE

JADE es una de las plataformas de agentes más utilizada hoy en día [BCG07]. Es una plataforma que permite el desarrollo y ejecución de aplicaciones basadas en agentes, completamente distribuida con una infraestructura flexible y que sigue las especificaciones *FIPA*. Éstas forman el conjunto de

estándares para aplicaciones y sistemas multi-agente más aceptado y extendido [BCG07]. *JADE* proporciona funcionalidades básicas, independientes de la aplicación específica y simplifica el desarrollo de aplicaciones distribuidas basadas en agentes.

Según el modelo de referencia para la gestión de agentes especificado por *FIPA* (*FIPA Agent Management*), mostrado en la figura 3.1, y seguido por *JADE*, el sistema de control de agentes debe contener los siguientes componentes[BCG07]:

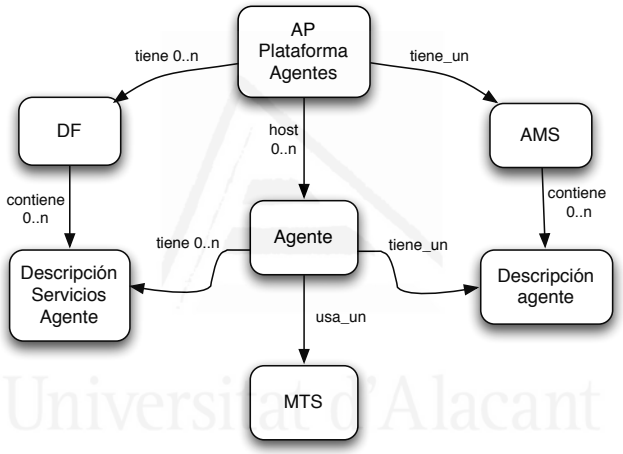


Figura 3.1: Descripción del sistema de gestión de agentes especificado por *FIPA* [Woo02].

- *AP* (*Agent Platform*): proporciona la infraestructura física donde los agentes están desplegados. Consiste en las máquinas, el sistema operativo, los agentes en sí mismos, componentes de gestión de *FIPA* y cualquier software de soporte adicional.

- Agente: es un proceso computacional dentro de la *AP* y normalmente ofrece uno o más servicios que pueden ser publicados para ser utilizados por otros agentes.

- *DF* (*Directory Facilitator*): es el servicio de páginas amarillas. Es un tipo especial de agente que permite a los agentes publicar una descripción de los servicios que ofrece para que otros agentes puedan encontrarlos fácilmente y utilizarlos. Mantiene una lista de agentes completa

y precisa y debe proporcionar la información más actual de los agentes en su directorio.

- *AMS* (*Agent Management System*): es un componente obligatorio en un *AP*. En *JADE* es un agente especial que proporciona el servicio de páginas blancas y juega el papel de controlador de la plataforma. Es el único agente que puede llevar a cabo tareas de gestión de la plataforma como crear y eliminar agentes o supervisar la migración de agentes de una plataforma a otra. Cada agente se debe registrar en el *AMS* para obtener un *AID* (*Agent Identifier*), que será conservado por el *AMS* como un directorio de todos los agentes presentes en la *AP* y su estado actual (activo, suspendido, esperando, etc.). La vida de un agente dentro de la *AP* termina con la cancelación del registro en el *AMS*. Después de esta cancelación, el *AID* de un agente puede ser borrado del directorio y puede estar disponible para otros agentes que lo soliciten. El *AMS* puede solicitar que un agente lleve a cabo una función de gestión específica, como terminar su ejecución; y tiene la autoridad para forzar esta operación si la solicitud es ignorada. Sólo puede existir un *AMS* en cada *AP*.

- *MTS* (*Message Transport Protocol*): es un servicio proporcionado por una *AP* para transportar mensajes FIPA-ACL entre agentes de una *AP* o entre agentes de diferentes *AP*.

En términos de su cobertura de estándares *FIPA*, *JADE* implementa completamente la especificación de gestión de agentes incluyendo los servicios de *AMS*, *DF* y *MTS*, para esto *JADE* hace uso de contenedores.

Una plataforma de *JADE* se compone de contenedores de agentes que pueden estar distribuidos por la red, como se muestra en la figura 3.2. En la plataforma existe un contenedor especial, llamado *Main Container*, que representa el punto de partida de la plataforma: es el primer contenedor en ser lanzado y el resto de contenedores deben unirse a él, registrándose en él. Este contenedor principal tiene las siguientes responsabilidades[BCG07]:

- Gestionar la tabla de contenedores (*CT*, *Container Table*), donde se mantiene un registro con las referencias a objetos y direcciones de todos los contenedores que componen la plataforma.

- Gestionar la tabla de descriptores de agentes global (*GADT*, *Global Agent Description Table*), donde se mantiene un registro con todos los agentes presentes en la plataforma, incluyendo su estado y ubicación.

- Mantener el *AMS* y *DF*, los dos agentes especiales que proporcionan la gestión de agentes y el servicio de páginas blancas (*AMS*) y el servicio de páginas amarillas (*DF*).

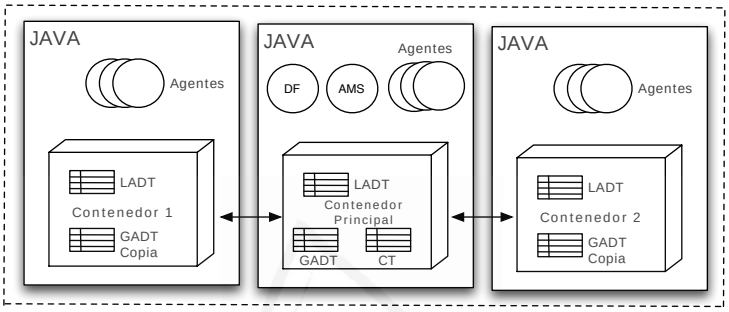


Figura 3.2: Relación entre los elementos principales de la arquitectura de JADE.

3.2. Tolerancia a fallos en *JADE*

La aplicación de la plataforma de agentes *JADE* para la construcción de un enjambre robótico puede suponer una serie de problemas. Como se ha visto en el punto 3.1.3, *JADE* se basa en el uso de contenedores, donde existe un contenedor principal responsable de gestionar toda la plataforma. Este contenedor tiene dos problemas principalmente, por un lado, se puede convertir en un cuello de botella para la plataforma; y por otro lado, al contener dos servicios clave, el agente especial *AMS* que lleva a cabo las tareas de gestión de agentes de la plataforma y el agente especial *DF* que proporciona el servicio de páginas amarillas, es un punto potencial de fallo porque ante un error en este contenedor todo el sistema quedaría inoperativo. Es necesario gestionar este contenedor de una manera efectiva para, por una parte, asegurar la tolerancia a fallos y que la plataforma permanezca operativa incluso en el caso de que se produzca un error en el contenedor principal y, por otra, para asegurar que la plataforma cumple las características que enfatiza la robótica de enjambre de control descentralizado y robustez.

Respecto al primero de los problemas en [BCG07] se indica que *JADE* evita el problema del cuello de botella en el contenedor principal porque proporciona una copia de la tabla *GADT* a cada contenedor, gestionándola

cada uno de estos contenedores localmente. Así, las operaciones de la plataforma, generalmente, no involucran al contenedor principal, sino que sólo afectan a la copia local y a los contenedores que contienen a los agentes involucrados en la operación. Cuando un contenedor debe encontrar la dirección de un agente, en primer lugar busca en su copia local $LADT$ (*Local Agent Description Table*), y sólo si la búsqueda es errónea, se contacta con el contenedor principal para obtener la referencia apropiada, que es almacenada en la copia local para usos futuros. Como el sistema es dinámico (los agentes pueden migrar, terminar su ejecución, pueden aparecer nuevos agentes, etc.) ocasionalmente el sistema puede utilizar una copia no actualizada obteniendo una dirección inválida. En este caso, el contenedor recibe una excepción y es forzado a actualizar su copia local desde el contenedor principal.

Sin embargo, aunque el contenedor principal no es un cuello de botella, sigue siendo un punto débil de la plataforma, ya que si se produce un error en éste se comprometería el funcionamiento de toda la plataforma. Es necesaria la utilización de algún mecanismo para la tolerancia a fallos que asegure la permanencia de los agentes AMS y DF en el sistema. Se han realizado diferentes estudios sobre tolerancia a fallos en sistemas multi-agente. Dos de las aproximaciones más utilizadas están basadas en redundancia (replicación) y manejo de excepciones [GBFM10][DTUV08][DLABA$^+$07].

La replicación tolera fallos en los componentes del sistema, preveniendo la ocurrencia de problemas. La idea simplificada de esta técnica es mantener múltiples copias de un componente, distribuidas por la plataforma. En caso de que el componente original falle, una de las copias tomaría el control de las tareas que estaba realizando. En [DLABA$^+$07] y [dLAABM06] proponen un sistema de replicación dinámico, donde se decide en tiempo de ejecución qué entidades son más críticas para replicar basándose en diferentes métricas, dependencias, roles y planes. En [KC00] presentan una arquitectura *Adaptive Agent Architecture* (AAA) basada en equipos de agentes intermedios llamados *brokers*. Estos agentes forman equipos jerárquicos y son los encargados de la comunicación y coordinación entre agentes. Proponen un esquema que utiliza la redundancia implícita en estos agentes para recuperar el sistema ante fallos en alguno de estos agentes.

Por otro lado, en el manejo de excepciones se envían eventos periódicos a los agentes para inspeccionar su estado. En [XD05] presentan una plataforma basada en eventos para la gestión de fallos en el sistema. Se basa en la idea de que la infraestructura de un sistema multi-agente se puede extender con un componente de gestión de fallos basado en eventos. En esta aproximación los agentes son capaces de enviar eventos como resultado de cambios en su estado interno o en la plataforma. Estos eventos son captados por el

componente gestor de eventos que los analiza y actúa en consecuencia. En [WLGW05] se define *FATMAD*, una plataforma de desarrollo multi-agente con tolerancia a fallos basada en *JADE*. Se basa en puntos de control y recuperación. Desarrollan dos protocolos interdependientes: el protocolo de creación de punto de control almacena la información en un almacenamiento estable y el protocolo de recuperación entra en acción ante un error para restaurar el punto de ejecución.

La replicación proporciona un tiempo más corto de recuperación y generalmente, es menos intrusivo. Tiene una mayor escalabilidad y es relativamente más genérico y transparente del dominio porque no se deben especificar de manera explícita las posibles anomalías que pueden surgir y la forma de tratarlas. Además, evita la sobrecarga del sistema por las pruebas periódicas que se deben realizar en el manejo de excepciones. Aunque se deben tener en cuenta los gastos adicionales de los elementos redundantes del sistema, no únicamente el coste inicial, sino también el soporte necesario para utilizar estos componentes o simplemente por tenerlos presentes en el sistema.

La gran mayoría de los trabajos que se acaban de describir se centran en conseguir la tolerancia a fallos en los agentes comunes del sistema y no en los agentes especiales. En [KSA+05], sin embargo, definen una arquitectura llamada *Virtual Agent Cluster* (VAC) que soporta AMS descentralizados en diferentes AP distribuidas. Cada AP tiene una instancia local del AMS. Esta aproximación se basa en el intercambio de mensajes para comprobar la existencia de los AMS. Esta arquitectura proporciona una mayor tolerancia a fallos al utilizar diferentes máquinas y hacer el sistema distribuido, con lo cual la caída total del sistema es más difícil pero no indica ninguna manera de recuperar aquellos elementos del sistema que se encuentran asociados al AMS caído.

Actualmente, *JADE* ya proporciona herramientas para asegurar que la plataforma permanezca operativa incluso ante un error en el contenedor principal [BCG07]. Lo consigue combinando dos características:

- Replicación del contenedor principal. Tanto el contenedor principal como el AMS son replicados, manteniendo todas las réplicas sincronizadas y asegurando que si se produce un fallo una de las réplicas tomará el control.
- Persistencia del DF. Permite que el DF sea guardado en un almacenamiento persistente. En caso de fallo del contenedor principal, un nuevo DF es creado automáticamente en el nuevo contenedor principal recuperando su catálogo del almacenamiento.

3.2.1. Replicación del AMS y persistencia del DF

Para mantener la plataforma totalmente operativa ante el fallo del contenedor principal, JADE da soporte a la replicación del contenedor principal. Ofrece el servicio $MCRS$ (*jade.core.replication.MainReplicationService*) que permite desplegar una plataforma tolerante a fallos lanzando varios contenedores principales lógicos en la plataforma [BCG07]. Sólo uno gestionará y mantendrá el sistema y el AMS mientras que el resto serán réplicas de éste. Todos estos contenedores principales constituyen un anillo lógico donde se pueden monitorizar unos a otros. Si uno falla, los otros son capaces de detectar el error y llevar a cabo las acciones de recuperación apropiadas. El resto de contenedores ordinarios de la plataforma se pueden conectar a ésta a través de cualquiera de los contenedores principales activos.

Este método cambia la topología de la plataforma *JADE* de una estrella a un anillo de estrellas como se puede observar en la figura 3.3. De esta manera, como se indica en [BCG07], es posible controlar el nivel de tolerancia a fallos de la plataforma, el nivel de escalabilidad y el nivel de distribución.

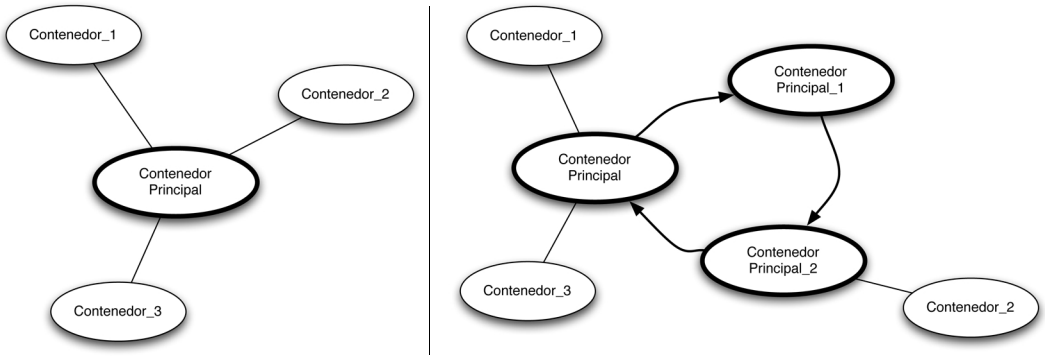


Figura 3.3: Topología de la plataforma JADE sin replicación del contenedor principal (izquierda) y con replicación (derecha). Imagen basada en [BCG07].

En este tipo de topología de anillo cada contenedor monitoriza a su vecino, de esta manera cuando un contenedor principal cae, el contenedor que le está monitorizando informa al resto de contenedores principales y se reorganizan en un nuevo anillo de menor tamaño. Si el contenedor principal que ha caído tenía otros contenedores conectados a él, éstos se vuelven huérfanos y se registran en otro contenedor principal activo. Para esto *JADE* ofrece el servicio ANS (*jade.core.replication.AddressNotificationService*) que automáticamente detecta nuevas incorporaciones o eliminaciones del anillo y

mantiene las direcciones de todos los nodos de la plataforma actualizadas.

Estos servicios aseguran que cada copia del contenedor principal está actualizada y con toda la información relevante, pero no mantiene el repositorio del servicio de páginas amarillas *DF*. Para una fiabilidad máxima JADE ofrece también métodos de persistencia del *DF*.

JADE ofrece dos posibilidades para almacenar el *DF*. La acción por defecto almacena el catálogo en memoria, lo que garantiza rapidez en el acceso pero, al mismo tiempo, se incrementa el consumo de memoria. Dependiendo de la cantidad de agentes que formen el sistema este método puede no ser la mejor opción, puesto que si el sistema está formado por un gran número de agentes puede llevar a problemas de escalabilidad. Por otro lado, también ofrece la posibilidad de configurar el *DF* para que almacene su catálogo en una base de datos relacional utilizando la interfaz de Java *JDBC* (*Java DataBase Connectivity*). Tiene la ventaja de un nivel constante de consumo de memoria. Además, asegura tolerancia a fallos del *DF*, que utilizado en combinación con el servicio *MCRS* permite el desarrollo de una plataforma totalmente tolerante a fallos. Sin embargo, en cuanto al rendimiento, el almacenamiento en una base de datos aumenta el tiempo de acceso, especialmente cuando se deben transferir un gran conjunto de datos de la base de datos al *DF*.

El uso de uno de estos dos métodos dependerá en gran medida de los requerimientos de la aplicación y, sobre todo, del número de agentes implicados.

3.2.2. Implementación y funcionamiento en *JADE*

En este punto se describirá la creación y el funcionamiento de las herramientas que ofrece *JADE* para desarrollar una plataforma tolerante a fallos. Se especificarán los pasos que sigue *JADE* para la creación e inicialización del contenedor principal y del resto de réplicas que forman el anillo de contenedores lógicos. También se indicará cómo actúa *JADE* ante el fallo del contenedor principal pasando el *AMS*, *DF* y los contenedores asociados a él a una de las réplicas.

Creación de la plataforma y contenedor principal

En primer lugar se creará el anillo de contenedores haciendo uso del servicio *MCRS* que ofrece *JADE*. A la hora de crear cada uno de los contenedores que forman el anillo se deben especificar los servicios *jade.core.replication. AddressNotificationService* y *jade.core.replication.Main ReplicationService*.

El primer paso es la creación de la plataforma y el contenedor principal. Al lanzar la orden de creación del contenedor principal se sigue un orden de ejecución e inicialización de los diferentes componentes del contenedor y de la plataforma. La figura 3.4 muestra una traza simplificada de la creación del contenedor principal.

Se empieza creando la tabla *LADT* (*Local Agent Description Table*) que como se ha comentado en el punto 3.2, se creará una tabla en cada contenedor para evitar el cuello de botella del contenedor principal.

A continuación se inicializa el temporizador y se establecen los atributos para el temporizador del hilo de ejecución.

Posteriormente se añade este nuevo nodo a la plataforma inicializando sus componentes. Crea e inicializa *IMTPManager* y obtiene el *Service Manager*. En esta inicialización se instancia el *AMS* y *DF* ya que el *AMS* se debe crear antes de la instalación de los servicios del kernel para evitar un error *NullPointerException* en el caso de que un servicio proporcione un comportamiento del *AMS*.

Una vez instanciados se procede a conectar el nodo local a la plataforma y activar los servicios. Para activar los servicios, se activan los servicios obligatorios *jade.core.management.AgentManagementService* y *jade.core.messaging.MessagingService*. Y a continuación los servicios adicionales necesarios para la tolerancia a fallos *jade.core.replication.MainReplicationService* y *jade.core.replication.AddressNotificationService*.

Una vez establecidos los servicios antes de iniciarlos se deben crear e iniciar los agentes que contendrán el *AMS* y el *DF*. Esta inicialización se debe realizar antes de arrancar los servicios ya que durante este inicio de los servicios se pueden generar mensajes dirigidos al *AMS* o *DF*. En primer lugar crea el agente *AMS* y notifica a la plataforma y posibles oyentes la creación del agente, posteriormente hace la misma operación con el agente *DF*. Cada vez que se crea un agente, en este caso el *AMS* y *DF* se envía un mensaje *Inform-Created* que llega a *MainReplicationService* para hacer un broadcast a todas las réplicas del sistema (aunque en este momento no haya ninguna).

A continuación inicia los servicios. Al iniciar los servicios se accede a las opciones de replicación en *MainReplicationService* y se almacenan los servicios que ofrece ese contenedor. Además, se genera el mensaje *New-MTP* del cual se hace un broadcast al resto de réplicas.

Por último, lanza los agentes *AMS*, *DF* y el resto de agentes del contenedor si los tuviera.

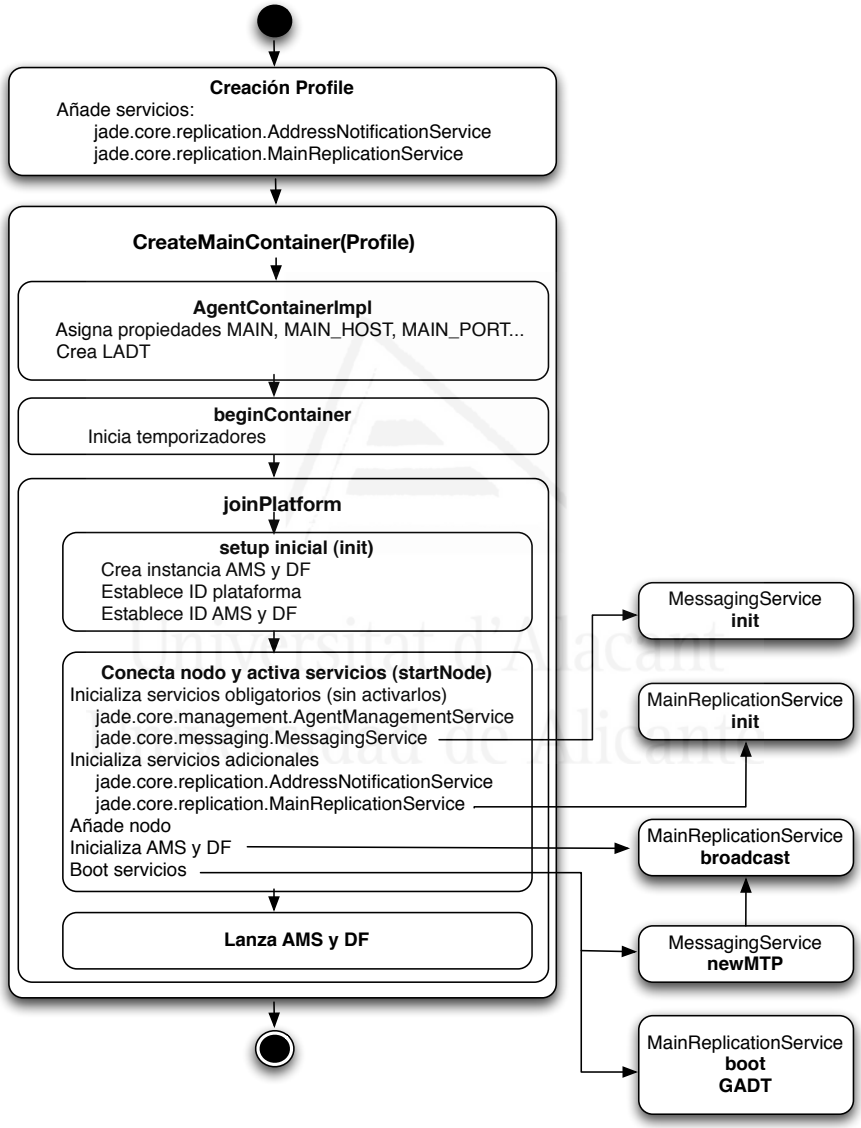


Figura 3.4: Simplificación de los pasos seguidos en la creación del contenedor principal.

Creación del anillo de contenedores lógicos

Una vez creada la plataforma y el contenedor principal se deben crear las réplicas que formarán el anillo de contenedores lógicos. Cada una de estas réplicas se crearán con la opción *LOCAL_SERVICE_MANAGER* y con los servicios adicionales *jade.core.replication.MainReplicationService* y *jade.core.replication.AddressNotificationService*. El orden de inicialización de los diferentes componentes de la plataforma es el mismo que en el contenedor principal. Se crea la tabla *LADT*, se inicia el temporizador del hilo de ejecución y se añade el nuevo nodo a la plataforma (se instancia el *AMS* y *DF*, se conecta el nodo a la plataforma y se establecen los servicios tanto los obligatorios como los adicionales). En el caso de la creación de las réplicas, como ya hay un *PlatformManager* creado se conectará a éste, creando una nueva réplica. Se crean los agentes *AMS* y *DF* que no serán activados y se avisa a todas las réplicas del sistema. En la creación de las réplicas se debe, además, asignar a la réplica el nodo al que va a monitorizar y hacer un broadcast a todos los nodos del anillo indicando la incorporación del nuevo nodo para la reconfiguración del anillo. Posterior a la inicialización de los servicios se muestra la posición en el anillo que ocupa la réplica. De esta manera, cada nodo será avisado de esta incorporación y mostrará por pantalla un mensaje similar al siguiente donde indica el número de réplica que es y a qué nodo monitoriza:

```
INFO: Main container ring re-arrenged: label = 1 monitored label = 0
```

Con estos pasos terminaría la creación e inicialización de la réplica.

La figura 3.5 muestra una traza simplificada de la creación de una de las réplicas del anillo. Al compararla con la figura 3.4 se observan las diferencias entre la creación de los dos contenedores sobre todo en lo referente a la inicialización del AMS y DF y en el lanzamiento del servicio MainReplicationService.

Cuando se añade otra réplica al anillo, las réplicas existentes capturan los comandos *ADD_REPLICA* y *ADD_NODE* con lo que añaden una nueva réplica y nodo y hacen un broadcast de esta incorporación. El servicio MessagingService de la réplica captura el comando *H_ADDROUTE* que añade la nueva dirección a los AID de los agentes locales y al *AMS* y *DF*. Por otra parte, el servicio MainReplicationService captura el comando *H_GETLABEL* con lo que obtiene la etiqueta del nodo y el comando *H_ADDREPLICA* que añade los servicios de la nueva réplica.

Si la réplica que recibe este comando fuera el líder (contenedor principal) en primer lugar debería cerrar el anillo monitorizando el nuevo servicio aña-

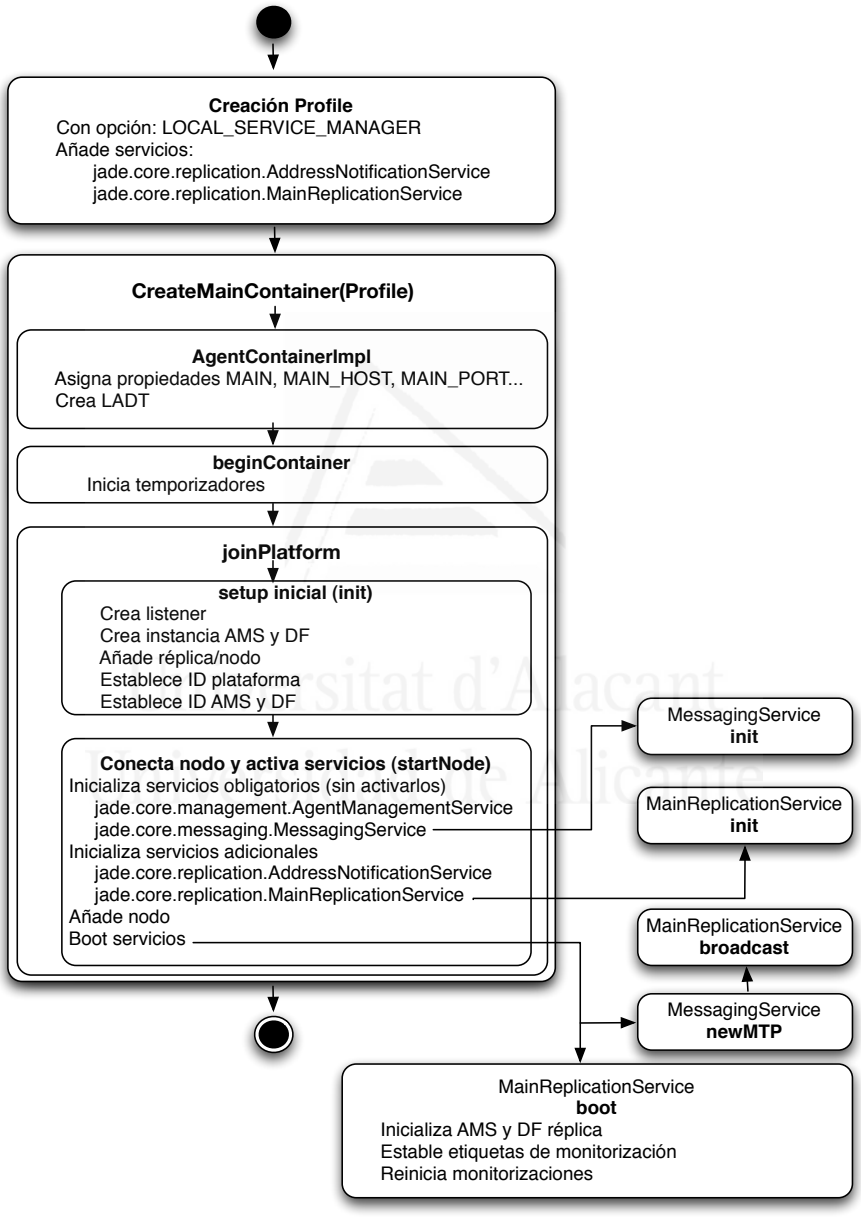


Figura 3.5: Simplificación de los pasos seguidos en la creación de una réplica.

dido y empezaría a enviar datos al nuevo servicio. Enviaría información de *GADT*, las descripciones del *AMS* y la lista de herramientas. Incorporaría dos nuevos comandos *NEW_NODE* y *NEW_SLICE* para cada servicio instalado en el nuevo nodo con el fin de permitir a los servicios locales propagar información específica de este servicio a sus servicios.

De esta manera se crearía el anillo de contenedores lógicos y el contenedor principal. La figura 3.6 muestra el contenedor principal y dos réplicas. Se observa cómo el contenedor principal tiene el *AMS* y *DF* mientras que las réplicas no contienen ningún agente.

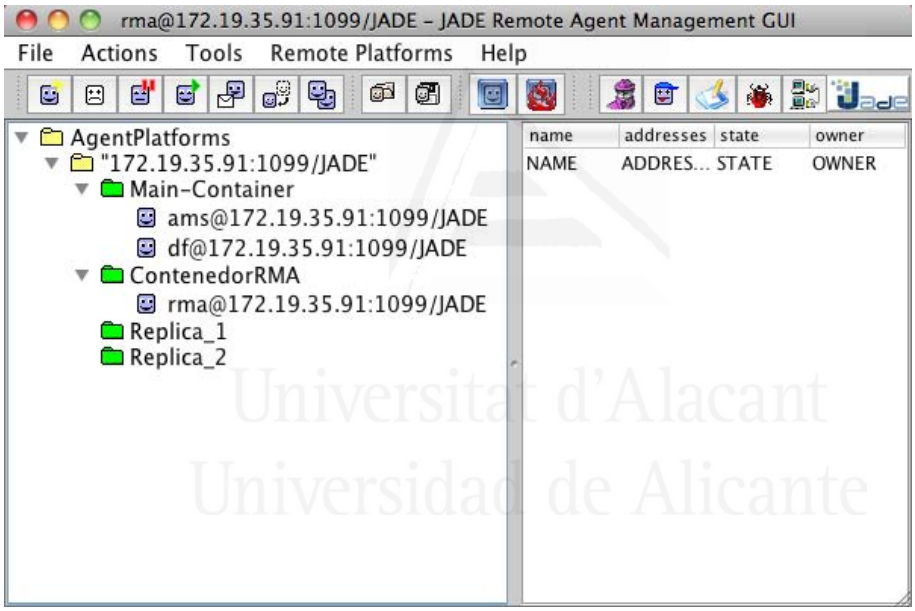


Figura 3.6: *RMA* de *JADE*. Se observa el contenedor principal con *AMS* y *DF* y dos réplicas del contenedor principal que no contienen *AMS* ni *DF*.

Reconfiguración del anillo ante la caída del contenedor principal

Cuando el contenedor principal que contiene el *AMS* y *DF* falla, el servicio *MainReplicationService* de las réplicas captura este fallo del sistema. En primer lugar comprueba si el nodo que ha fallado es el que está monitorizando, si es así comprueba la conectividad.

Si el contenedor principal parece estar muerto se puede deber a diferentes causas que no podemos distinguir, por ejemplo, puede ser una muerte real o una desconexión de red de larga duración. Para poder realizar comprobacio-

nes específicas del entorno, es posible redefinir el método *checkConnectivity* para forzar alguna de las tres acciones predefinidas:

- *REMOVE_NODE*: comportamiento por defecto, elimina el nodo y si es necesario toma el liderazgo.
- *WAIT*: no hace nada en espera de recibir algún comando.
- *SUICIDE*: mata el contenedor local para evitar futuros problemas de creación si la conectividad se reestableciera.

En este caso, la conectividad devolverá *REMOVE_NODE* y el sistema deberá eliminar el nodo. Como el contenedor ha muerto, tampoco estará activo el AMS, por lo tanto, se empiezan a interceptar los eventos de la plataforma y del MTP de parte del nuevo AMS. A continuación se elimina la réplica notificando primero a los nodos que no son hijos ni forman parte del anillo, y después a los nodos hijos. Después de realizar un broadcast de notificación a todas las réplicas se debe eliminar el nodo. Esta eliminación de nodo, elimina todos sus servicios, elimina su contenedor remoto y elimina todos los MTP instalados en el contenedor muerto. Para esto se requiere que el contenedor aun esté presente en la tabla de contenedores *Container Table*. Posteriormente elimina el contenedor de la tabla, elimina todos los agentes del contenedor muerto (incluido AMS y DF) y para terminar hace un broadcast a todas las réplicas. A continuación, se deben reasignar las etiquetas del anillo y reiniciar la monitorización. Por último, si se da el caso, y es el contenedor que monitorizaba al contenedor principal, se debe convertir en el nuevo líder, para ello debe inicializar y lanzar los agentes AMS y DF.

La figura 3.7 muestra un resumen simplificado de los pasos seguidos por el servicio *MainReplicationService* cuando el contenedor principal muere y una de las réplicas se vuelve el nuevo líder.

La figura 3.8 muestra ahora cómo la réplica_1 que monitorizaba al contenedor principal se ha vuelto el nuevo líder tras la caída de éste y ahora contiene al AMS y DF. Además el contenedor RMA que antes dependía del contenedor principal, aun con la muerte de éste, sigue en la plataforma dependiendo ahora del nuevo líder.

3.3. Mobile Ad-hoc NETworks

Como se ha visto en el capítulo 2 los sistemas robóticos de enjambre pueden desarrollar diferentes comportamientos y tareas donde se exige que

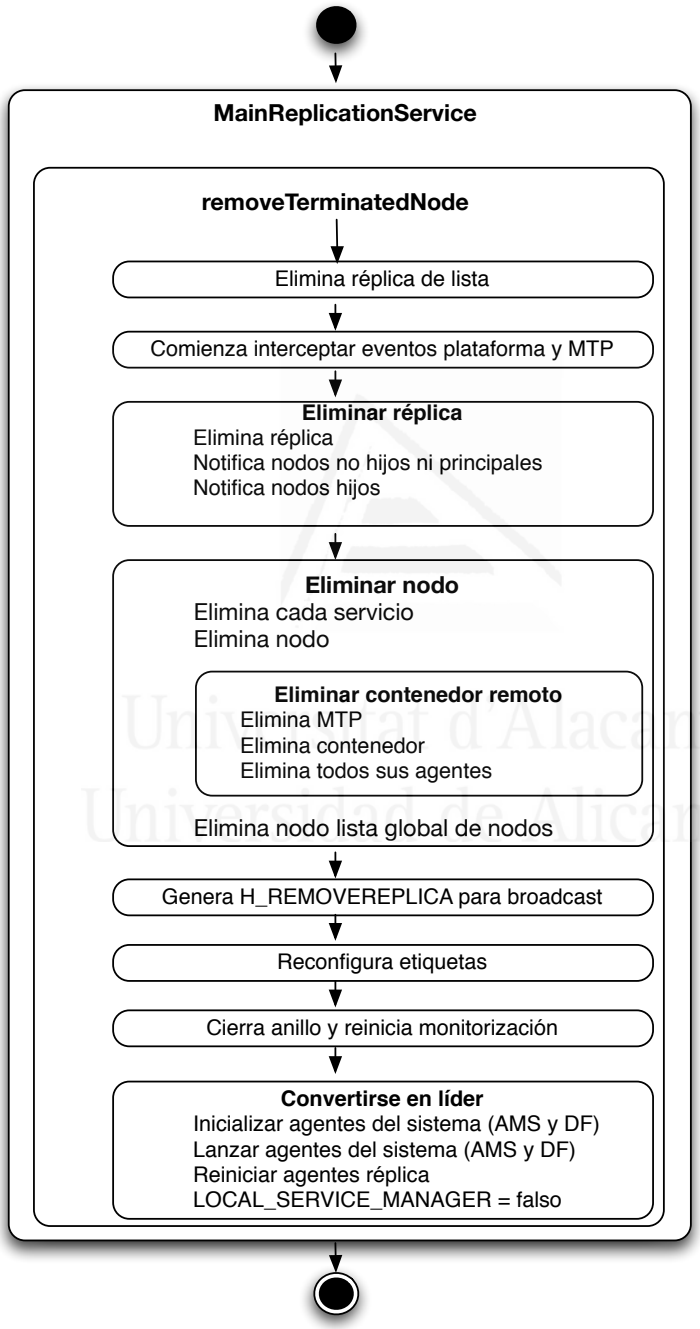


Figura 3.7: Simplificación de los pasos seguidos cuando el contenedor principal falla y la réplica que lo monitorizaba se vuelve líder.

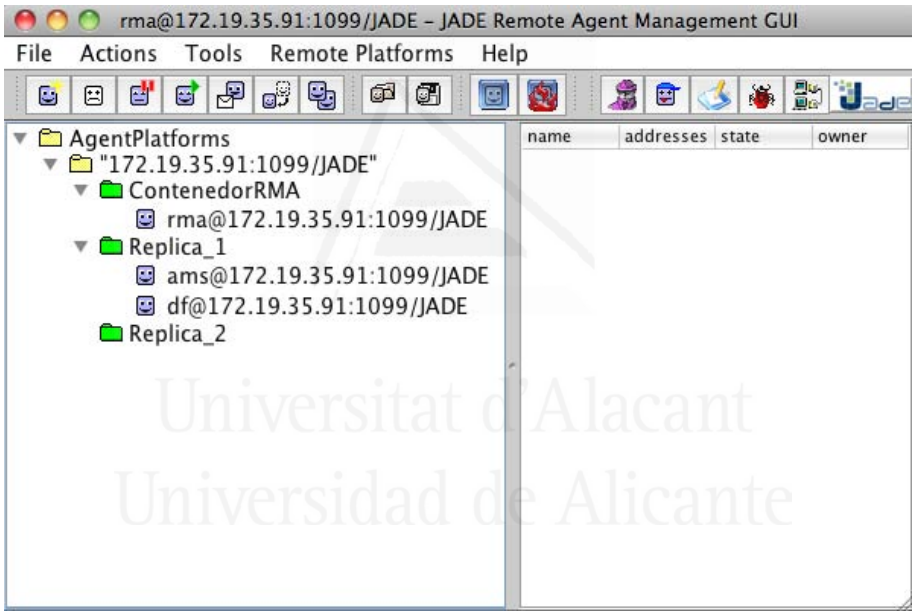


Figura 3.8: *RMA* de *JADE*. Se observa como la réplica_1 se ha vuelto el nuevo líder alojando ahora al *AMS* y *DF* ante el fallo del contenedor principal.

los robots se desplieguen por zonas donde no hay infraestructura de red y por tanto, es necesario que los robots sean capaces de formar su propia red. En este sentido, una aproximación a este problema sería el uso de redes *Mobile Ad hoc NETworks* (*MANET*).

Una red *MANET* es una colección de dispositivos con capacidades de comunicación inalámbrica capaces de comunicarse entre ellos y cooperar de manera distribuida para proporcionar las funcionalidades de red necesarias sin la ayuda de ningún administrador central y sin utilizar ninguna infraestructura de red fija existente [HMDD04][AOP10]. Estas redes son ideales en situaciones donde no hay tiempo para configurar un punto de acceso, o donde no hay infraestructura fija disponible [CB08]. Son redes que ofrecen ventajas como flexibilidad, movilidad, independencia y tolerancia [HMDD04].

Este tipo de redes tienen un comportamiento diferente a las redes inalámbricas con infraestructura, donde cada usuario se comunica directamente con un punto de acceso o estación base. Las redes *MANET* son redes auto-organizadas y auto-administradas, siendo una asociación transitoria y autónoma de nodos móviles. Los dispositivos son libres para dejar y unirse a la red, y moverse aleatoriamente lo que resulta en cambios de topología rápidos e impredecibles. Los grupos de nodos móviles formados dinámicamente deben ser capaces de coordinarse entre ellos para llevar a cabo el enrutamiento y la búsqueda de recursos. De esta manera, los nodos que se encuentran en el rango de envío de otro pueden comunicarse directamente y son responsables de descubrir dinámicamente estos nodos. Para permitir la comunicación entre nodos que no se encuentran directamente dentro del rango de envío, los nodos intermedios actúan como enrutadores para enviar los paquetes generados por otros nodos a los destinatarios. Estas rutas entre nodos deben cambiar rápidamente, requiriendo protocolos flexibles [CB08].

3.3.1. *FIPA* y *MANET*

En el año 2002 *FIPA* empezó un comité técnico (*TC ad hoc*) para desarrollar una solución para plataformas *FIPA* en entornos ad hoc. En [BWH03] se definen las propuestas que realizaron y se describen los requerimientos y posibles aproximaciones desde un punto de vista de la arquitectura. Finalmente *FIPA* decidió no desarrollar su protocolo propio de búsqueda de servicios en entornos ad hoc, sino utilizar alguna de las tecnologías ya existentes. Dos de las razones de la decisión fueron que permitir el registro y búsqueda de servicios de agentes utilizando tecnologías de descubrimiento ya existentes para conexiones punto a punto (P2P) que han sido desarrolladas específicamente para estos entornos, puede llevar a una gestión más

eficiente de las descripciones de los servicios y directorios; además que estas tecnologías también pueden ser utilizadas para redes fijas [PBW04].

A pesar de esto y que *JADE* asume una conectividad completa y continua entre contenedores [BCG07] está disponible un complemento desarrollado por un consorcio formado por compañías como Motorola, Siemenes AG o Telecom Italia llamado *LEAP* (*Lightweight Extensible Agent Platform*) que tiene en cuenta las pérdidas de conexión. El objetivo principal de este componente *JADE-LEAP* fue crear un entorno suficientemente ligero para poderse ejecutar en dispositivos móviles con recursos limitados como teléfonos móviles, aunque fue diseñado con propiedades de escalabilidad para poder ser ejecutado con dispositivos sin estas limitaciones ya que proporciona la misma *API*, ofreciendo una capa homogénea a pesar de la diversidad de dispositivos.

JADE se basa en el uso de contenedores, pero el complemento *LEAP* proporciona una manera alternativa de implementación llamada *slit execution mode*, especialmente diseñada para los requerimientos de los dispositivos móviles. De manera que no se crea un contenedor normal, sino una fina capa llamada front-end. Esta capa proporciona agentes con las mismas características que los contenedores, pero implementando únicamente un pequeño subconjunto, mientras que delega los otros a un proceso remoto llamado back-end. A la vista de los agentes que residen en él, el front-end es un contenedor normal, mientras que a la vista del resto de contenedores existentes en la plataforma, incluido el contenedor principal, el back-end es un contenedor normal. El resultado es que el front-end y el back-end forman un contenedor dividido en dos partes. Estas dos partes se comunican a través de una conexión dedicada.

Este modo de ejecución además de permitir la ejecución en dispositivos con más restricciones, el front-end y back-end incluyen un mecanismo para hacer transparente para las aplicaciones las pérdidas de conexión. Esto es, si la conexión entre ellos cae, los mensajes hacia o desde los agentes alojados en el front-end se almacenarán en un búfer, y tan pronto como la comunicación se restablezca los mensajes serán entregados a sus destinatarios.

Aunque este complemento *LEAP* tenga en cuenta estas pérdidas de conexión no es adecuado para robótica de enjambre en entornos donde no hay disponible una infraestructura de red fija ya que las pérdidas de conexión que tiene en cuenta son únicamente entre el front-end y el back-end, asumiendo que el back-end y resto de contenedores normales de la plataforma, incluido el contenedor principal, tienen una conectividad completa y continua.

3.3.2. Robótica de enjambre y *MANET*

Por las características de las redes *MANET* vistas anteriormente, éstas pueden ser una estructura de red adecuada para trabajar con enjambres de robots, donde los robots pueden dispersarse por entornos donde no esté disponible una estructura de red fija y donde los robots pueden también dispersarse por zonas con pérdidas de cobertura.

Como las redes *MANET* se caracterizan por una topología de red multipunto que puede cambiar frecuentemente debido a la movilidad, es necesario un protocolo de enrutamiento eficiente para establecer la comunicación entre nodos. *DigiMesh* es un protocolo que permite el desarrollo de este tipo de redes. En una red *DigiMesh* sólo existe un tipo de nodo, de manera que todos los nodos pueden enrutar los datos y además son intercambiables. Entre las ventajas que ofrece *DigiMesh* se encuentra la facilidad de configuración de la red, mayor flexibilidad de expansión y aumento de la fiabilidad en entornos donde puedan existir cortes de conexión [Pap08]. Es posible generar una red de este tipo utilizando los módulos de transmisión inalámbrica *XBee* que permiten utilizar el protocolo *DigiMesh*. La figura 3.9 muestra cómo quedaría la arquitectura con la incorporación de los módulos *XBee* y el protocolo *DigiMesh*.

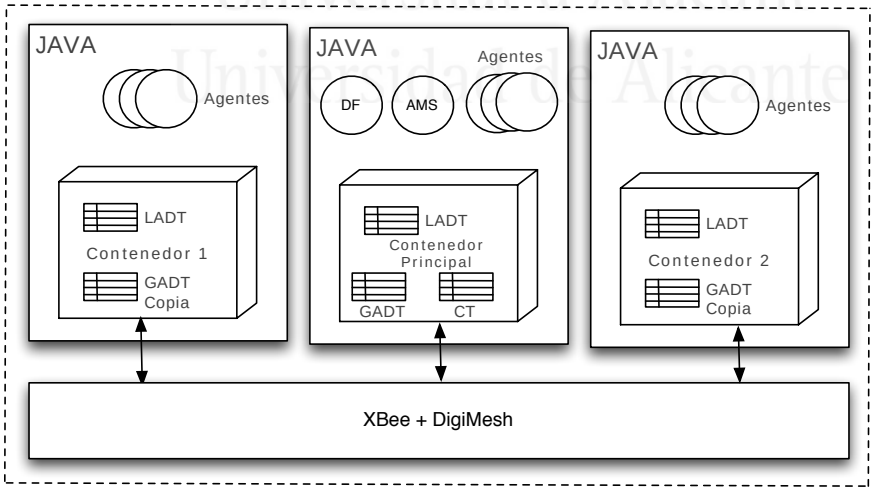


Figura 3.9: Arquitectura del sistema con el uso de los módulos de transmisión inalámbrica *XBee* y el protocolo *DigiMesh*.

Por un lado, la robótica de enjambre podría asumir la pérdida de un in-

dividuo, ya que según sus características, el sistema seguiría siendo robusto ante la pérdida o fallo de uno de los individuos porque éste puede ser compensado por otro individuo, siendo ninguno de ellos imprescindible. Pero, por otro lado, y a pesar de que esta carecterística permitiría al sistema continuar en funcionamiento ante la pérdida de uno de sus componentes, el uso de *JADE* junto con una red *MANET* nos da una solución ante problemas de desconexión y falta de cobertura.

Debido al uso de *JADE*, cuando un agente no obligatorio se desconecta de la plataforma, el resto del sistema continuaría en funcionamiento. Si la conexión volviera a restablecerse el agente continuaría formando parte del sistema. Si durante la desconexión se enviaran mensajes desde o hacia el agente caído, estos mensajes se perderían, pero el protocolo de red *MANET DigiMesh* permite el uso de un mecanismo de confirmación de recepción de mensajes que permitiría conocer si el mensaje ha sido recibido y por lo tanto actuar en consecuencia si no fuera el caso.

3.4. Resumen

En este capítulo se ha realizado una breve descripción de los agentes y sistemas multi-agente, enfatizando que éstos comparten muchas características con los sistemas robóticos de enjambre vistos en el capítulo anterior y por tanto podrían ser una buena alternativa para su implementación.

Se han descrito los componentes especificados por *FIPA* y seguidos por *JADE* para un sistema de gestión de agentes. Se han analizado los problemas que podría suponer el uso de un contenedor principal como alojamiento de los agentes *AMS* y *DF* para la implementación de un sistema robótico de enjambre: el cuello de botella que supondría el contenedor principal y el punto débil de la plataforma, ya que si el contenedor principal fallara podría poner en riesgo el funcionamiento de todo el sistema.

Respecto al primer problema, el cuello de botella del contenedor principal, se ha visto como *JADE* evita este problema proporcionando una copia de la tabla *GADT* a cada contenedor que la gestiona localmente. Por otro lado, para el problema que podría suponer la caída del contenedor principal, se han analizado las herramientas que proporciona *JADE* para evitarlo, la replicación del contenedor principal y la persistencia del *DF*.

La replicación del contenedor principal cambia la topología de estrella básica de una plataforma a una topología en anillo, donde el anillo estará formado por réplicas del contenedor principal, de manera que si el contenedor principal falla, una de las réplicas tomará el control de la plataforma,

pasando a ella el AMS, el DF y todos los componentes del sistema que se hubieran quedado huérfanos.

En el punto 3.2.2 se aportan varias trazas que muestran el funcionamiento de estas herramientas, como la creación del anillo de contenedores lógicos y la reconfiguración de éste ante la caída del contenedor principal. Debido a que no existe documentación detallada del funcionamiento de estas opciones de *JADE*, se ha debido realizar un estudio del código interno del mismo para la elaboración de estas trazas.

Para esto se ha utilizado la versión *JADE 4.0*, puesto que en la última versión disponible, *JADE 4.2*, no funciona correctamente este servicio de replicación al no estar instalado *NotificationService*, que hace que el sistema no sea notificado apropiadamente ante la caída del contenedor principal.

Por último, en robótica de enjambre se debe tener en cuenta la posibilidad de pérdidas de cobertura o incluso la posibilidad de que el enjambre se despliegue por zonas donde no haya una instraestructura de red disponible, por lo tanto, pueden existir pérdidas de conexión en los agentes. Como *JADE* suele trabajar con una conectividad completa y continua entre contenedores, se ha prorporcionado una posible solución con la incorporación de una red *MANET* que proporciona las funcionalidades de red necesarias sin la ayuda de ningún administrador central y sin utilizar infraestructuras de red fija, utilizando los módulos de transmisión inalámbrica *XBee* junto con el protocolo de enrutamiento para establecer la comunicación entre nodos *DigiMesh*.

En el capítulo siguiente se revisarán las características de las arquitecturas robóticas existentes. Teniendo en cuenta las características vistas en este capítulo, se presentará el diseño y funcionamiento de un modelo de arquitectura basado en un sistema multi-agente para el control de un enjambre robótico.

Capítulo 4

Arquitecturas robóticas y sistemas de enjambre

En el desarrollo de sistemas robóticos móviles, la arquitectura robótica tiene un papel muy importante ya que, por un lado, es la plataforma que interconecta todos los subsistemas y controla el sistema global [ME10] y, por otro, proporciona todas las estructuras necesarias para la coordinación, comunicación y control [LOC00]. Por lo tanto, la definición de una arquitectura robótica debe proporcionar una guía para la construcción de sistemas inteligentes y debe describir los componentes que debe tener el sistema, cómo se organizan y cómo interactúan para darle al sistema su funcionalidad.

Las arquitecturas existentes pueden ser divididas, de manera general, en dos grupos: las arquitecturas individuales, que se centran en las decisiones y comportamientos de un único robot; y las arquitecturas multi-robóticas, que prestan más atención a la gestión de las relaciones y flujos de información entre los diferentes componentes para poder llevar a cabo tareas que necesiten mecanismos cooperativos.

La mayor parte del esfuerzo en este campo se ha centrado durante muchos años en la definición de arquitecturas individuales, donde han sido desarrolladas muchas arquitecturas de control para robots móviles. Cada una de ellas es apropiada y se adapta a un tipo de problema particular. Posteriormente, este esfuerzo se ha dividido realizándose estudios donde se presentan arquitecturas para sistemas multi-robóticos centrándose en la orquestación de los comportamientos entre varios robots.

En esta sección se presenta un modelo de arquitectura híbrida basada en agentes para el control de un sistema robótico de enjambre. Éste modela las diferentes características de la robótica de enjambre. Por un lado, permite

que la coordinación de los robots sea distribuida, por otro lado, es fácilmente escalable permitiendo la agregación y eliminación de robots en el sistema, y por último, tiene una alta tolerancia a fallos con sistemas de control que permiten la eliminación y agregación de nuevos agentes ante el fallo de alguno de ellos. La principal aportación de este modelo es facilitar el uso de agentes de alto nivel para dirigir un enjambre de agentes simples y facilitar la monitarización de éstos. Esto posibilitará la ejecución de tareas complejas, sin perder, por un lado, las ventajas de los sistemas de enjambre, y por otro lado, aportando las ventajas de los sistemas multi-agente (reutilización de código, conectividad, estandarización,...) ya que permite la formalización de enjambres mediante sistemas multi-agente.

En este capítulo se realiza una revisión de las arquitecturas robóticas existentes, diferenciando entre arquitecturas individuales y arquitecturas multi-robóticas. A continuación, se analizarán las propiedades de estas arquitecturas y su adecuación a la robótica de enjambre. Y, por último, se presentará un modelo de arquitectura híbrida basada en agentes para el control de un enjambre de robots.

4.1. Arquitecturas robóticas

Los esfuerzos iniciales en los campos de la robótica móvil y computación basada en agentes se centraron en la especificación de arquitecturas de control inteligentes, donde se establecieron algunos paradigmas que han perdurado en el tiempo. Este conjunto de arquitecturas tiene un rango muy amplio, desde las arquitecturas puramente reactivas (o de conducta) que actúan según la relación entre estímulo-respuesta, como la arquitectura de subsunción de Brooks, a las arquitecturas deliberativas capaces de razonar sobre sus acciones, como aquellas basadas en el modelo BDI (*Belief Desire Intention*). Entre estos dos extremos se encuentran las arquitecturas híbridas o por capas, que intentan unir reacción y deliberación en un esfuerzo por adoptar la mejor acción en cada momento [BCG07].

Actualmente, se diferencian tres paradigmas en el diseño de arquitecturas robóticas: deliberativo, reactivo e híbrido, según la relación entre tres primitivas básicas: sentir, planificar y actuar. La figura 4.1 muestra la relación entre estas tres primitivas según cada paradigma. A continuación se describirán estos paradigmas y algunas de sus arquitecturas más representativas o recientes.

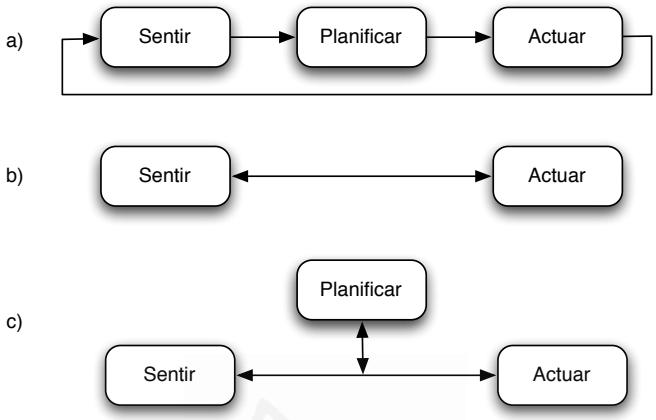


Figura 4.1: Paradigmas de arquitecturas robóticas. a)Jerárquico (deliberativo), b)Reactivo y c)Híbrido.

4.1.1. Paradigma deliberativo

El paradigma deliberativo (o jerárquico) es el más antiguo prevaleciendo desde mitad de la década de los 60 hasta principios de los 90. En este paradigma el robot siente el mundo, planea la siguiente acción y actúa. Entonces, vuelve a sentir el mundo, planificar y actuar. En cada paso, el robot planea explícitamente el siguiente movimiento [Mur00]. Es decir, en cada paso, el robot siente el entorno y crea un modelo del mundo combinando su información sensorial. A continuación, este modelo es utilizado por el módulo de planificación para generar un camino hacia la meta. Y finalmente, el robot ejecuta las acciones planificadas. El robot repetirá este proceso hasta alcanzar el objetivo.

Para realizar el proceso de planificación o razonamiento es necesario que el robot tenga una representación interna del entorno, de manera que la información sensorial se orienta en intentar obtener un modelo del mundo lo más exacto posible. Esta exigencia es en muchos entornos, hoy en día, inviable. Además, es posible que se dé el caso que en el tiempo transcurrido desde la percepción hasta la ejecución de las acciones el entorno varíe, siendo falsas las percepciones que supone el robot.

Un ejemplo de arquitectura deliberativa es la arquitectura NASREM. El nombre de esta arquitectura deriva de las siglas de *NASA/NBS Standard REference Model* [AML89]. Esta arquitectura define un modelo teórico que representa el sistema de control del robot con seis capas organizadas en tres

módulos (procesamiento sensorial, modelado del mundo y descomposición de tareas), ayudados por un sistema de comunicación y una memoria global.

La necesidad de un modelo global del mundo, es una de las principales desventajas de este modelo. Algunos de los problemas derivados de esta necesidad son:

- No completitud del modelo. La creación de esta representación del entorno puede ser en muchos casos una tarea complicada donde se puede dar el caso de que todas las características del entorno real no puedan ser representadas en el modelo. Por lo tanto, es posible que el robot realice acciones incorrectas por falta de información.
- Lentitud en el proceso. En cada ciclo el robot debe actualizar el modelo global del entorno y entonces realizar el proceso de planificación. Este proceso puede ser lento y por tanto conllevar una serie de problemas: por un lado puede resultar en un cuello de botella significativo y; por otro lado, se puede dar el caso de que en el tiempo transcurrido desde la percepción hasta la acción el entorno haya variado y por tanto, el robot puede llevar a cabo acciones incorrectas.
- Incertidumbre en el entorno. En entornos dinámicos e inciertos es complicado que el robot pueda mantener la representación interna del entorno actualizada. Lo que podría también conllevar acciones incorrectas.
- Problema del marco de actuación [Mur00]. En muchas ocasiones no es posible predecir el efecto que tendrán las acciones del robot en el entorno, lo que dificulta la planificación a largo plazo.

4.1.2. Paradigma reactivo

El paradigma reactivo fue muy utilizado entre finales de la década de los 80 y principios de los 90. Estas arquitecturas tratan de evitar las desventajas de la aproximación deliberativa en entornos dinámicos y desconocidos. En esta aproximación no es necesario construir un modelo del mundo y la información leída de los sensores está directamente acoplada con los actuadores del robot utilizando un conjunto particular de funciones de transferencia llamadas comportamientos [NTMNM11]. Este paradigma se basa en el principio de acción-reacción, rechazando la planificación y basándose en una conexión, más o menos directa, entre los sensores y los actuadores. Se compone de múltiples procesos sentir-actuar que componen los patrones de comportamiento.

Existen muchas arquitecturas que podrían clasificarse dentro del paradigma reactivo pero una de las arquitecturas más formalizadas es la arquitectura de subsunción de Brooks [Bro86] (1986). En esta arquitectura los comportamientos se dividen en capas, estando en las capas más bajas los comportamientos básicos que proporcionan la supervivencia (como evitación de obstáculos) y en las capas más altas los comportamientos más orientados a la consecución de los objetivos. La implementación de cada capa se puede realizar de diferentes maneras, Brooks utiliza máquinas de estados finitos. En la figura 4.2 se muestra un ejemplo de esta arquitectura.

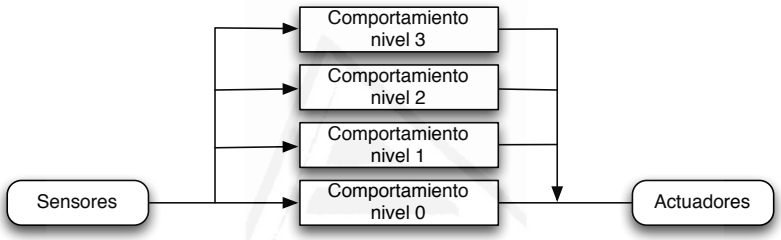


Figura 4.2: Ejemplo de arquitectura reactiva. Estructura del modelo de Subsunción de Brooks.

Otro ejemplo de arquitectura reactiva es la arquitectura *motor schema* propuesta por Arkin [Ark89] (1989) inspirada en ciencias biológicas. En esta arquitectura cada comportamiento utiliza la aproximación de campos de potencial para producir su salida en forma de vector. Estas salidas se combinan y la respuesta global del sistema es el sumatorio de estos vectores. Mientras que la arquitectura de subsunción aboga por una selección de comportamientos competitiva, la arquitectura *motor schema* se basa en una coordinación cooperativa.

El paradigma reactivo tiene una serie de ventajas importantes [NTMNM11]:

- Ausencia de representación interna. No requiere la creación ni almacenamiento de un modelo del entorno.

- Rapidez derivada de la eliminación de la planificación. Tiene una mayor rapidez de reacción en entornos desconocidos y dinámicos. Al existir menos computación el tiempo entre percepción y acción es menor y, por lo tanto, pueden reaccionar de manera adecuada a cambios en el entorno.

- Escalabilidad. El sistema tiene la capacidad de poderse construir de manera incremental, capa a capa.

- Robustez y fiabilidad. Si una unidad de comportamiento falla, no implica la caída de todo el sistema. Las otras unidades de comportamiento pueden seguir en funcionamiento.

Las principales desventajas de este paradigma son:

- Falta de planificación. La eliminación completa de la planificación puede ser una postura muy extrema para el desarrollo de tareas complejas.

- Dificultad en la coordinación de los comportamientos. La interacción entre el sistema y el entorno puede ser menos predecible.

- Dificultad en la definición de tareas de alto nivel. La mayoría de comportamientos especificados suelen ser comportamientos de bajo nivel.

4.1.3. Paradigma híbrido

Muchos sistemas robóticos han sido desarrollados en base a los dos paradigmas anteriores: el reactivo (basado en el principio de acción-reacción) y el deliberativo (que implica la planificación explícita de cada tarea). Una combinación de ambos, el paradigma híbrido, fue definido para intentar mejorar los problemas que tenían los dos modelos anteriores. Este tipo de arquitectura híbrida deliberativa/reactiva basada en los dos paradigmas anteriores surgió a principio de la década de los 90 [Ark90] y sigue siendo la más utilizada actualmente [PY09] [OC03] [ME10]. Este paradigma incorpora una capa deliberativa sobre la capa reactiva. El robot mantiene una capa reactiva que le proporciona supervivencia básica en el entorno y sobre ésta, incluye una capa deliberativa que le permite realizar tareas más complejas [PY09]. Con esta orientación se tiene la velocidad de actuación del nivel reactivo y la potencia de navegación y de realización de tareas complejas que tiene el nivel deliberativo.

En la definición de arquitecturas robóticas se ha vuelto común utilizar un diseño basado en capas. Aunque no existe un número fijado de capas y las capas difieren de una arquitectura a otra, así como los mecanismos de comunicación y coordinación entre ellas, mayoritariamente se utilizan tres capas [OC03] [ME10] [NTMNM11]: una capa deliberativa, una reactiva y una capa intermedia de control de la ejecución. La capa superior, deliberativa, está más orientada a los objetivos llevando a cabo la planificación, localización y

razonamiento a nivel global. La capa inferior, reactiva, proporciona respuestas rápidas basadas en los sensores. Esta capa se basa en comportamientos, con un firme acoplamiento entre sensores y actuadores. La capa intermedia es un puente entre la capa deliberativa y la reactiva. Es responsable de supervisar la interacción y coordinación entre la capa superior e inferior.

Existen una gran variedad de arquitecturas que se pueden englobar en este paradigma que hacen difícil su clasificación. En [Mur00] y otros trabajos basados en éste, como [NTMNM11] y [PY09], realizan una clasificación de estas arquitecturas dividiendo las arquitecturas híbridas en: arquitecturas híbridas organizativas, basadas en jerarquías de estados y orientadas a modelos. A continuación se describirán cada una de estas aproximaciones, incluyendo una categoría centrada en las arquitecturas basadas en sistemas multi-agente.

Arquitecturas híbridas organizativas

Las arquitecturas híbridas organizativas se basan en una descomposición de las responsabilidades de manera similar a como se realiza en la gestión empresarial. Estas arquitecturas distinguen claramente la capa reactiva y la deliberativa. Es esta última la que contiene un conocimiento global del mundo, mientras que la capa reactiva es la que contiene los comportamientos básicos. En la parte superior se encuentran los módulos que están a cargo de la planificación de alto nivel. Estos planes son enviados a los subordinados, que los refinan y pasan al nivel inferior. Este nivel, es el nivel reactivo y lleva a cabo las tareas básicas. De este tipo de arquitecturas destacan la arquitectura AuRA [AB97] y SFX [MA92]. Otras arquitecturas más recientes son Yavuz [YB02] y Tripodal [KC06].

Arquitectura AuRA

La arquitectura AuRA (*Autonomous Robot Architecture*, 1987) se basa en la teoría de esquemas. Se compone de cinco subsistemas divididos en dos capas diferenciadas, como se puede observar en la figura 4.5. La capa superior, deliberativa, contiene dos subsistemas: el planificador y el sistema cartográfico. El planificador es responsable de planificar la misión y las tareas. El sistema cartográfico encapsula las funciones de modelado del entorno más cercano y lectura de los sensores, necesario para la navegación. El nivel inferior, reactivo, se compone de: el subsistema motor y el subsistema de percepción. El subsistema de percepción se compone de esquemas de percepción que adquieren la información de los sensores. El subsistema motor se compone de esquemas motores que forman los comportamientos.

Entre las dos capas anteriores se encuentra el subsistema homeostático. Este sistema es muy importante porque modifica los comportamientos en función de las necesidades internas del robot. Es decir, lleva a cabo las tareas de supervivencia del robot en momentos peligrosos modificando su comportamiento.

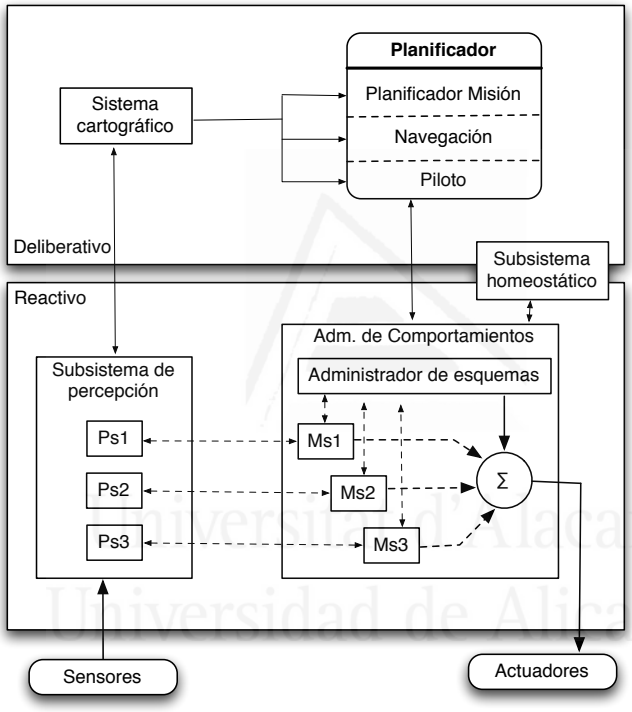


Figura 4.3: Ejemplo de arquitectura híbrida organizativa: Arquitectura AuRA.

Esta arquitectura tiene las ventajas de modularidad y flexibilidad. Está compuesta por diferentes módulos que pueden ser remplazados y además, es posible añadir nuevos esquemas de motor y de percepción. Por el contrario, es una arquitectura centralizada que ha sido diseñada específicamente para tareas de navegación en robots móviles. Además, dispone de un módulo muy importante, el subsistema de control homeostático, imprescindible para gestionar las tareas de supervivencia.

Arquitectura SFX

La arquitectura SFX (*Sensor Fusion Effects*, 1992) es una extensión de

la arquitectura AuRA centrada en la fusión sensorial y en la gestión de errores [Mur00]. Se trata también de una arquitectura con dos niveles, uno deliberativo y otro reactivo. La capa reactiva se divide en dos capas, una relativa a los comportamientos estratégicos y otra a los tácticos. SFX utiliza un método de filtrado donde algunos comportamientos tácticos inhiben a otros comportamientos estratégicos. De esta manera para situaciones transitorias se priman comportamientos tácticos capaces, por ejemplo, de evitar un determinado obstáculo. En ausencia de contingencias se priman los comportamientos estratégicos, encargados de desarrollar objetivos de un nivel superior. El componente deliberativo está dividido en módulos, implementados como agentes, que interactúan entre ellos. Existe un agente supervisor, llamado *planificador de la misión*, que establece la interfaz a alto nivel con el usuario y controla la evolución del sistema. A un nivel inferior se compone de tres subsistemas encargados de atender a los sensores y actuadores: administrador de sensores, administrador de tareas y administrador de actuadores. El agente administrador de sensores es capaz de obtener mediante fusión la fiabilidad de los sensores y solventar problemas con los mismos.

Una de las principales características de esta arquitectura es la robustez, con la inclusión de mecanismos de tolerancia a fallos en el sistema. Por el contrario, se basa en módulos centrales, el planificador de misiones, que especifica las directrices de la misión y controla el sistema. Además, también utiliza estructuras de datos globales, tipo pizarra, para compartir la información sensorial tanto en el nivel reactivo como en el deliberativo.

Arquitectura Yavuz

En [YB02] se presenta una arquitectura modular jerárquica (Yavuz, 2002) con dos niveles, deliberativo y reactivo, formados por varias capas. La capa deliberativa selecciona los comportamientos activos dependiendo del modo de funcionamiento utilizando un módulo de generación de comando difuso y un módulo de arbitraje. Esta arquitectura permite al robot tener tres modos de funcionamiento: manual, donde un operador dirige al robot; aprendizaje, donde el operador indica al robot la manera de resolver la tarea y el objetivo; y reproducción (*playback*), donde el robot lleva a cabo de manera autónoma la tarea aprendida. Por lo tanto, es una arquitectura apropiada para robots que deben llevar a cabo tareas establecidas.

Aunque es una arquitectura modular, flexible y escalable, de manera interna se basa en un módulo central imprescindible que recibe tres señales: los comandos del usuario, la información del sistema sensorial e información de la tarea; y produce tres señales: información para la interfaz del usuario, los comandos de control e información para la tarea.

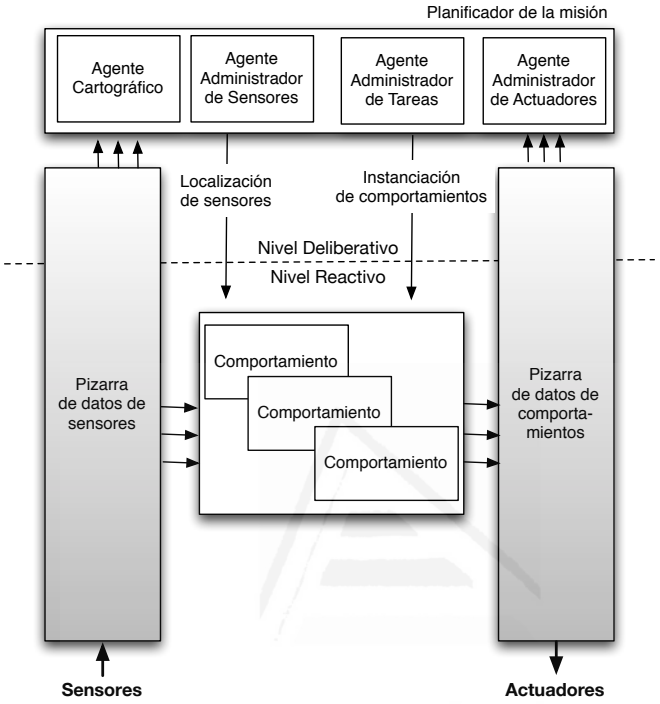


Figura 4.4: Ejemplo de arquitectura híbrida organizativa: Arquitectura SFX.

Arquitectura Tripodal

La arquitectura Tripodal [KC06] (*Tripodal Schematic Control Architecture*, 2006) está basada en tres capas: una capa deliberativa; una capa reactiva; y una capa intermedia de secuenciación. La capa deliberativa interactúa con los usuarios y lleva a cabo el proceso de planificación central. Recibe la tarea a realizar indicada por el usuario a través del módulo HRI. El planificador descompone esta tarea en procesos secuenciales teniendo en cuenta la información almacenada en el módulo de configuración de alto nivel y envía uno a uno estos procesos al supervisor de procesos de la capa de secuenciación. Según el resultado del proceso el planificador decide si enviar el siguiente proceso o realizar una replanificación. La capa de secuenciación se compone de una parte supervisora (compuesta por el supervisor de procesos y la configuración de bajo nivel) que lleva a cabo los procesos gestionando los componentes de la capa reactiva y de la capa de secuenciación, y una parte de información (compuesta por los módulos de navegación y manipulación) que extraen información avanzada a partir de los datos obtenidos por los

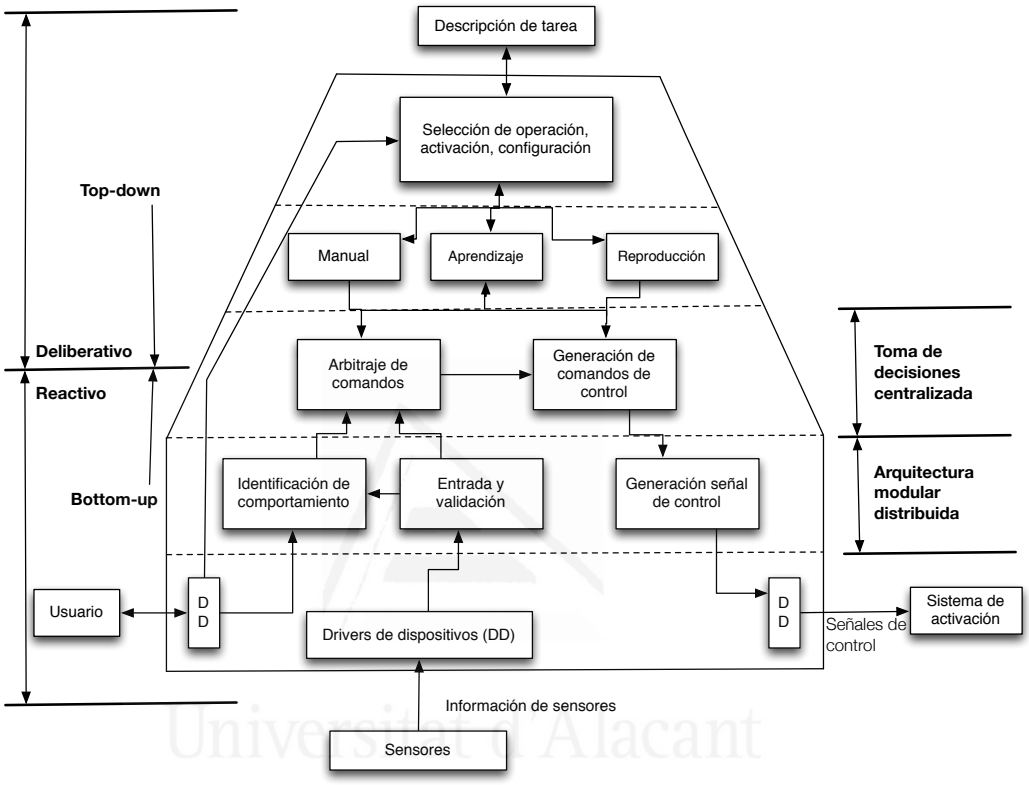


Figura 4.5: Ejemplo de arquitectura híbrida organizativa: Arquitectura Yavuz.

sensores. La capa reactiva contiene los componentes que trabajan en tiempo real y que se relacionan con el hardware. La arquitectura recibe información de las sensores a través del módulo *fuentes*. El módulo comportamiento controla el movimiento del robot ejecutando cálculos muy simples a partir de la información recibida de los sensores o de la información enviada desde la capa de secuenciación. El coordinador de comportamientos coordina los comandos de control enviados desde diferentes comportamientos y los envía a los actuadores del robot. El esquema de la arquitectura mostrado en la figura 4.6 indica la disposición de los diferentes módulos que la forman y el flujo de información entre ellos.

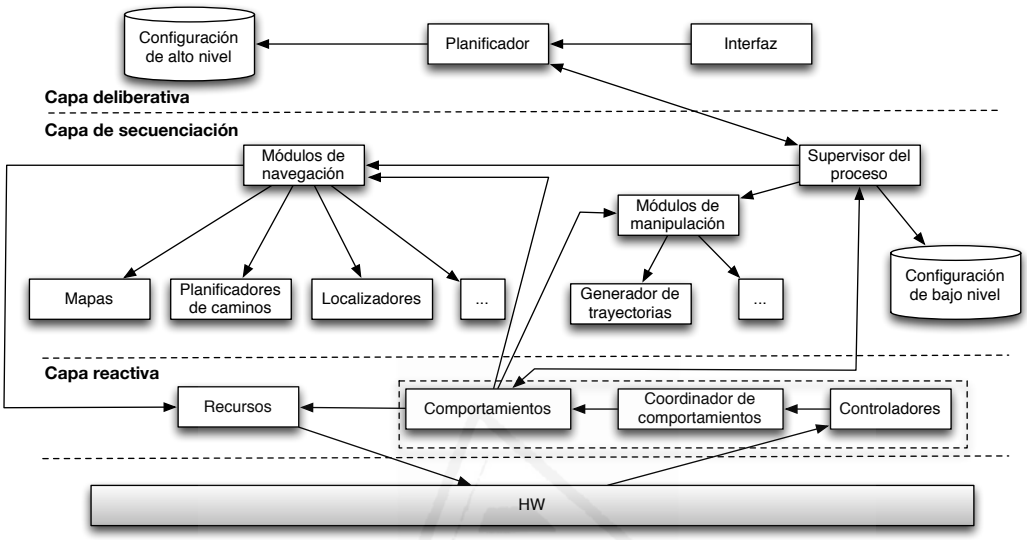


Figura 4.6: Ejemplo de arquitectura híbrida organizativa: Arquitectura *Tripodal Schemantic Control Architecture*.

Arquitecturas híbridas basadas en jerarquía de estados

Las arquitecturas basadas en jerarquías de estados organizan las actividades en base al conocimiento temporal. Suelen tener tres capas, basadas en el estado de conocimiento, dedicadas al pasado, presente y futuro. La capa reactiva y la deliberativa se distinguen en función del instante temporal en el que trabajan. La reactividad se establece como un comportamiento puramente reflexivo, con conocimientos locales, y, por tanto, requiere únicamente conocimiento del presente. Por otra parte, la capa deliberativa trabaja con el conocimiento pasado (lo que el robot ha hecho) y futuro (suposiciones y predicciones). El mejor ejemplo de esta arquitectura es la arquitectura 3T usada por la *NASA* [BKJJ98]. Otros ejemplos de este paradigma son BERRA [LOC00] y SSS [Con92].

Arquitectura 3T

La arquitectura 3T (*3 Tiered*, 1997) pretende alcanzar un comportamiento robusto para la resolución de tareas utilizando tres niveles: un nivel reactivo, uno deliberativo, y un nivel intermedio que sirve como interfaz de los dos anteriores. En la capa superior se encuentra el planificador. Es la capa deliberativa. Se encarga de proporcionar una perspectiva global del

sistema. Establece los objetivos y planifica las estrategias. Para ello trabaja con información del pasado, presente y futuro. Esta información se pasa a la capa intermedia, el secuenciador. Esta capa dispone de una librería de habilidades recomendadas para cada tarea a realizar. Por lo tanto, transforma las tareas proporcionadas por el nivel superior en habilidades a desarrollar por el nivel inferior. Esta capa trabaja con información del pasado y presente. Estas habilidades forman los comportamientos del nivel inferior, la capa reactiva. De esta manera se llega a una representación uniforme para las capacidades que facilita su manipulación. Las capacidades siempre operan en el presente. La figura 4.7 muestra un esquema de esta arquitectura.

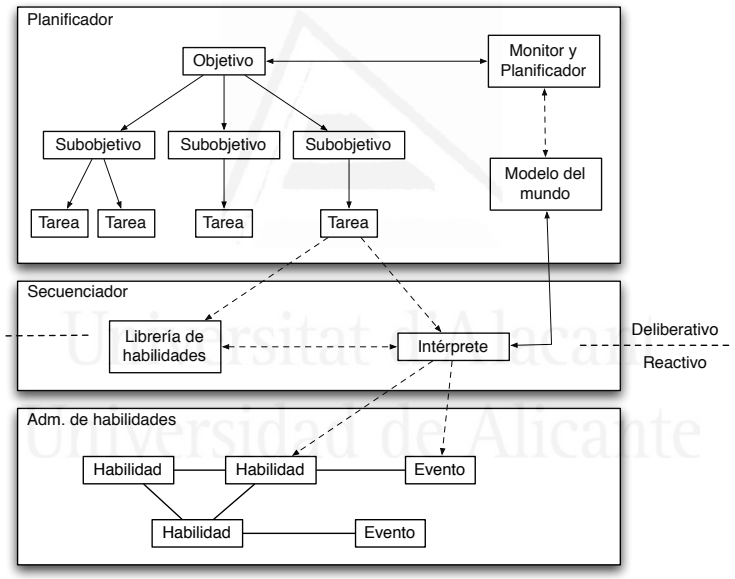


Figura 4.7: Ejemplo de arquitectura híbrida basada en jerarquía de estados: Arquitectura 3T.

La arquitectura 3T pretende que los tres niveles definidos operen de manera concurrente y asíncrona. De esta manera, los algoritmos lentos deben ir en el planificador y los rápidos en el gestor de capacidades. No obstante, esta división puede implicar clasificaciones no del todo lógicas, por ejemplo, los sistemas de visión, que corresponden a funciones sensoriales de bajo nivel, se deberían disponer en el planificador en lugar de en las capacidades.

Arquitectura Berra

La arquitectura BERRA (*BEhavior-based Robot Research Architecture*, 2000) [LOC00] ha sido diseñada para un robot móvil capaz de llevar a cabo tareas ordinarias de una oficina. Puede ser dividida en tres capas: una deliberativa, otra para la ejecución de tareas y una última capa reactiva. La capa deliberativa recibe órdenes de una persona, por lo tanto, es capaz de entender estas órdenes y posteriormente realizar una planificación del camino y de la tarea que se le ha encomendado. Por lo tanto, esta capa se compone de una interfaz humano-robot y de un planificador. El planificador convierte las órdenes en una lista de estados consecutivos. Cada uno de estos estados representa cierta configuración de la capa reactiva. Cada estado se envía individualmente a la capa inferior y cuando esta capa contesta que la tarea ha sido llevada a cabo se envía el siguiente estado. Si se recibe que la tarea no ha podido ser llevada a cabo se revisa la planificación. Si no es posible realizar la tarea se avisa al operador humano. La capa intermedia actúa de puente entre las dos capas anteriores. La capa reactiva necesita ser configurada y monitorizada conforme establece la capa deliberativa. Esta capa de ejecución de tareas gestiona la capa reactiva. Recibe los estados de la capa superior y los traduce en una configuración adecuada de la capa reactiva. En esta capa se encuentra el localizador que realiza un seguimiento de la posición del robot. La capa reactiva se compone de diferentes comportamientos. Esta capa contiene un módulo de recursos que recibe la información sensorial del robot y envía a cada comportamiento la información que necesita. Así mismo, cada comportamiento define una tarea sencilla y envía a un módulo controlador las acciones del robot. Este módulo controlador controla el movimiento del robot a partir de las diferentes órdenes recibidas de los comportamientos. Es el encargado de enviar los comandos a los actuadores.

Esta arquitectura es totalmente distribuida, siendo una de sus principales ventajas, ya que cada componente se desarrolla como un proceso individual, lo que facilita la distribución de la arquitectura en diferentes máquinas. Por otro lado, el proceso de planificación necesita información sobre la topología del entorno para realizar la planificación. Además, la arquitectura dispone de un módulo central que gestiona todos los procesos.

Arquitecturas híbridas orientadas a modelos

Las arquitecturas orientadas a modelos utilizan un modelo global del mundo que se utiliza también como percepción para los comportamientos, llamado en este caso sensor virtual. Estas arquitecturas plantean la deliberación como todo aquello que esté relacionado con un comportamiento

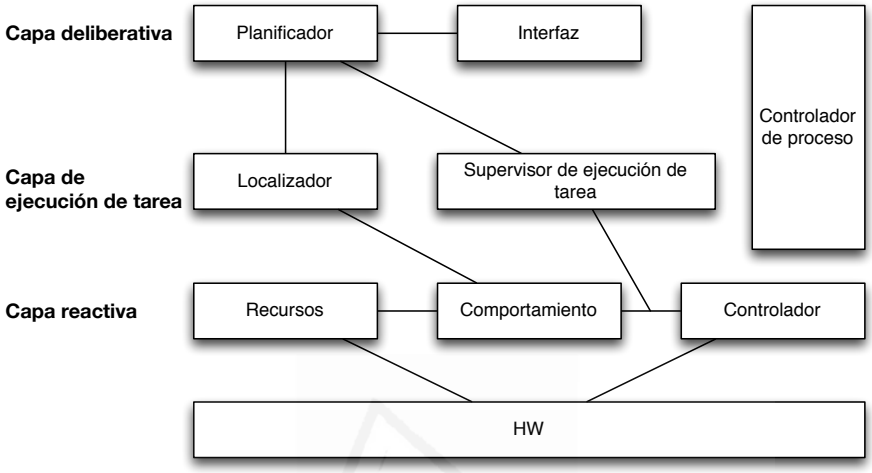


Figura 4.8: Ejemplo de arquitectura híbrida basada en jerarquía de estados: Arquitectura BERRA.

condicionado por una meta o un objetivo. Por otro lado, la reactividad se establece como pequeñas unidades de control que operan en el presente pero pudiendo utilizar conocimiento global proporcionado por un sensor virtual. El uso de un modelo global del mundo parece una vuelta a las arquitecturas jerárquicas, pero estas arquitecturas se diferencian en los siguientes aspectos [Mur00]:

- El modelo se restringe a regiones de interés. El modelo es menos ambicioso que en las arquitecturas jerárquicas y mejor organizado. En muchas ocasiones simplemente se etiquetan regiones.
- El procesamiento perceptual es distribuido y asíncrono para rutinas de percepción lentas.
- Los errores sensoriales y la incertidumbre pueden ser filtrados utilizando mecanismos de fusión sensorial.

Dos de las arquitecturas más conocidas de este tipo son Saphira [KMRS97] y TCA [Sim94].

Arquitectura Saphira

La arquitectura Saphira (1997) plantea como finalidad principal la construcción de agentes móviles autónomos con capacidad para atender, aprender y ejecutar tareas con robustez [KMRS97]. Es una arquitectura que se puede dividir en dos capas; una capa inferior reactiva y una capa superior deliberativa. La figura 4.9 muestra un esquema de esta arquitectura. Toda la arquitectura está construida en torno a un componente central de representación interna, el espacio de percepción local LPS (*Local Perceptual Space*). Existe una parte perceptora que se encarga de añadir la información sensorial al LPS y de procesarla para extraer información que puede ser utilizada para el reconocimiento de objetos y la navegación. Y una parte efectora donde se ejecutan los diferentes comportamientos. El nivel de control de Saphira se basa en comportamientos. Los comportamientos reactivos se definen y coordinan utilizando lógica difusa [Azn06]. De esta manera, la salida de los comportamientos son reglas difusas que serán combinadas para obtener los comandos de velocidad y dirección del robot. La selección y coordinación de comportamientos la realiza el controlador PRS-Lite (*Procedural Reasoning System-lite*). PRS-Lite es un sistema para la representación y la deliberación sobre acciones y procedimientos, que está basado en la arquitectura de agentes BDI. Es un sistema de planificación híbrido que intenta combinar la deliberación con la reactividad mediante el uso de planes meta-control. Algunas características de este controlador son: capacidad de integración de actividades dirigidas por objetivos o por eventos y descomposición jerárquica de tareas. Una de las principales ventajas de la arquitectura Saphira se basa en la independencia de los componentes, pudiéndose ejecutar en diferentes nodos. Por el contrario, se basa en un nodo central, el LPS, fundamental para el control del sistema.

Arquitectura TCA

Según Murphy en [Mur00], la arquitectura TCA (*Task Control Architecture*, 1994) es más parecida a un sistema operativo que a una arquitectura de propósito general. Esta arquitectura no proporciona unos comportamientos concretos, para unas tareas concretas, sino que proporciona una serie de estructuras de control para desarrollar estos comportamientos. El sistema se compone de diversos módulos de tareas específicas que se comunican por medio del envío de mensajes a través de un servidor central que programa y enruta los mensajes.

Una de las principales limitaciones de esta arquitectura, especificada en su propio manual, es que no es apta para tareas que requieran ejecución en tiempo real.

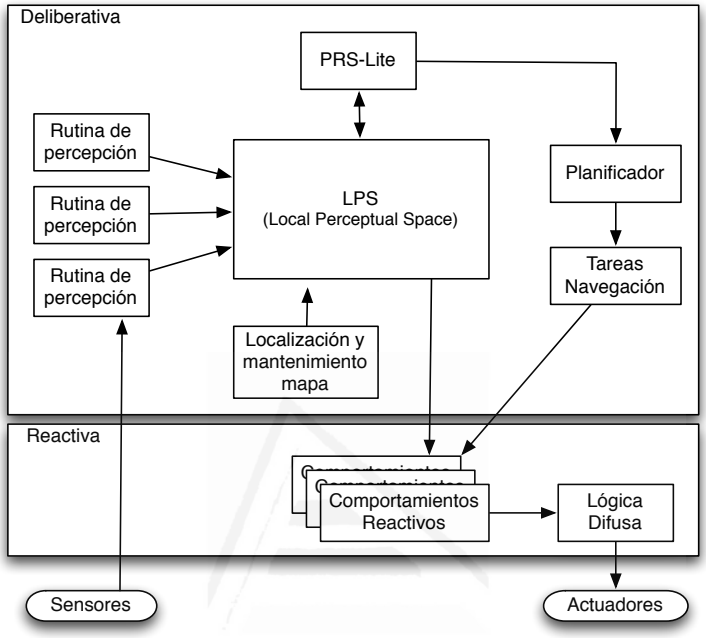


Figura 4.9: Ejemplo de arquitectura híbrida orientada a modelo: Arquitectura Saphira.

Arquitecturas basadas en agentes

La mayoría de arquitecturas vistas en los puntos anteriores están basadas en módulos. Recientemente, muchos estudios introducen los sistemas multi-agente (MAS, *Multi-Agent Systems*) para construir sistemas robóticos. Dentro de las arquitecturas híbridas, el tradicional modelo de control ha sido extendido en el uso de sistemas distribuidos basados en agentes [XWM11]. A continuación se describen algunas arquitecturas basadas en sistemas multi-agente para el control de un robot individual.

Arquitectura Busquets

En [BSdM03] (2003) se define una arquitectura para un sistema de navegación basado en visión. Esta arquitectura se estructura en tres capas (como se observa en la imagen 4.10): una capa inferior donde se sitúan los sensores y actuadores; una capa intermedia donde se encuentran los sistemas ejecutores que tienen acceso a los sensores y actuadores del robot y ofrecen servicios al resto de sistemas; y una capa superior donde se sitúan los sistemas delibe-

rativos que llevan a cabo tareas de alto nivel. Esta arquitectura, a su vez, se compone de tres sistemas: el sistema piloto, el sistema de visión y el sistema de navegación. Los dos primeros sistemas son sistemas ejecutores y el último es deliberativo. El sistema piloto es responsable de todos los movimientos del robot. Selecciona los movimientos para llevar a cabo los comandos que recibe del sistema de navegación y, de manera independiente, para evitar obstáculos. El sistema de visión es responsable de identificar y hacer un seguimiento de los puntos de interés. Finalmente, el sistema de navegación es responsable de tomar las decisiones de alto nivel para guiar al robot hasta el objetivo. Por un lado requiere la información obtenida por el sistema de visión y, por otro lado, envía al sistema piloto las órdenes para mover el robot. Este sistema se compone de varios agentes, donde cada uno es competente de una tarea específica. Dependiendo de las responsabilidades que tenga el agente y de la información recibida de otros agentes, cada agente propone al sistema de navegación la acción que debe llevar a cabo. Además, existe un agente especial, llamado coordinador que coordina todo el sistema.

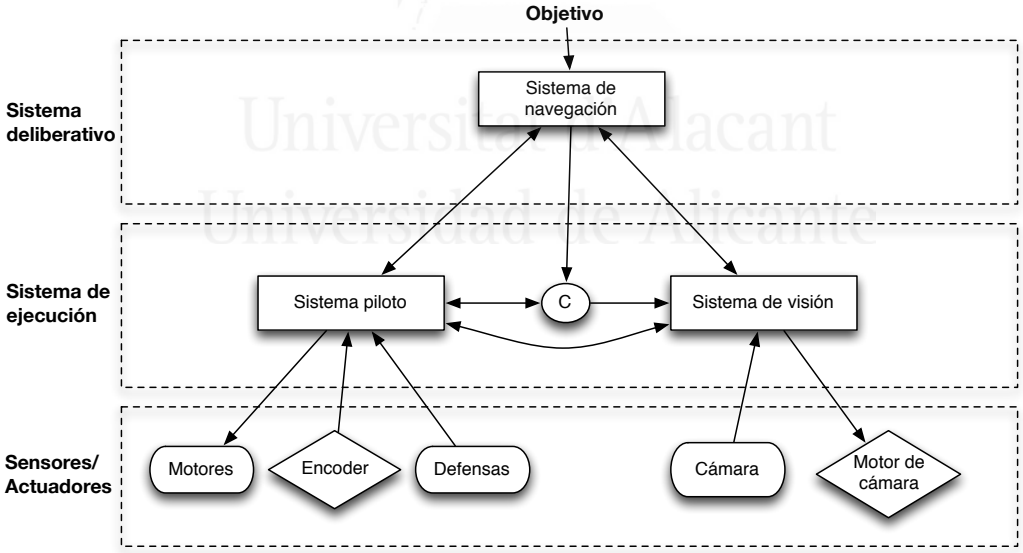


Figura 4.10: Ejemplo de arquitectura basada en sistemas multi-agente: Arquitectura Busquets.

Estos sistemas necesitan cooperar, ya que se necesitan unos a otros para alcanzar el objetivo global, pero también, compiten por controlar los actuadores disponibles del robot. A pesar de la distinción que se realiza en capas,

la arquitectura no es jerárquica y la coordinación se realiza como si todos los sistemas estuvieran en un único nivel pudiendo compartir información entre todos los sistemas. Por otro lado, todo el sistema depende del agente central coordinador.

Arquitectura Innocenti

En [ILS07] se define una arquitectura multi-agente (2007) combinada con control cooperativo. La arquitectura global está definida como un sistema multi-agente, como se muestra en la figura 4.11, mientras que utilizan el control cooperativo para diseñar y desarrollar cada agente individual. Esta arquitectura se compone de cuatro subsistemas: percepción, comportamiento, deliberación y actuador. Además, se compone del agente cliente que contiene la interfaz con el usuario y la plataforma de agentes que proporciona todos los servicios necesarios para garantizar el correcto funcionamiento de la plataforma. El subsistema de percepción obtiene información sobre el entorno y sobre las condiciones internas del robot. Recolecta los datos de los sensores y los adapta para proporcionarles esta información al resto de agentes del sistema. El subsistema de comportamiento lleva a cabo acciones específicas, como evitar obstáculos, dirigirse a un punto, etc. Estos agentes reaccionan a la información enviada desde los agentes de percepción. El subsistema deliberativo se compone de agentes que llevan a cabo las tareas de alto nivel. Llevan a cabo las tareas de localización, planificación de tareas y planificación del camino. El subsistema de actuación es responsable de enviar al robot las órdenes de movimiento según la información recibida del resto de subsistemas.

Aunque esta arquitectura tiene una serie de ventajas derivadas del uso de un sistema multi-agente como la flexibilidad, escalabilidad y autonomía de cada agente, tiene también varias desventajas como el uso de un agente directorio (*directory facilitator agent*) imprescindible para el funcionamiento de la plataforma y para la interacción entre agentes, lo que resulta en una sobrecarga de comunicación.

Arquitectura SC-Agent

En [PPS+08] (2008) presentan un diseño de arquitectura modular portable para el control de robots móviles. Definen una arquitectura híbrida distribuida multi-nivel basada en agentes software que interactúan a través de una pizarra de comunicaciones. La arquitectura se compone de tres partes distribuidas (como se muestra en la figura 4.12): un nivel deliberativo, un nivel reactivo, y una plataforma intermedia de comunicaciones. Las relaciones e interacciones entre los diferentes componentes de la arquitectura se

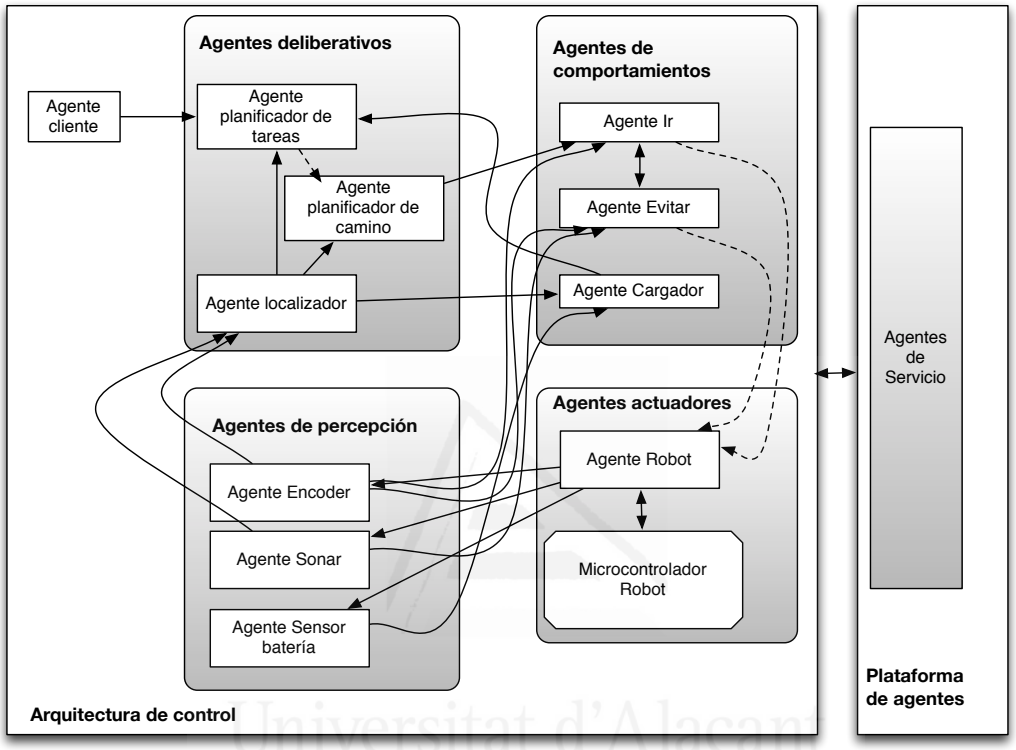


Figura 4.11: Ejemplo de arquitectura robótica basada en agentes: Arquitectura definida por B. Innocenti en [ILS07].

llevan a cabo en esta plataforma intermedia. El nivel deliberativo representa el nivel más alto de conocimiento. Este nivel ejecuta tareas de planificación basándose en el estado de un modelo simbólico interno del entorno. Como resultado de esta planificación de tareas, el planificador de la misión divide los objetivos en patrones de comportamiento que son enviados al nivel reactivo. El nivel reactivo recibe los esquemas perceptuales y de motor del nivel deliberativo. Estos esquemas son agentes ejecutándose concurrentemente que utilizan la información de los sensores para calcular las acciones a llevar a cabo por los actuadores. Este nivel se compone de: los sensores y actuadores; los esquemas perceptuales (que acceden a los sensores y producen percepciones) y motores (que acceden a las percepciones y producen acciones); y las motivaciones (cada esquema motor está unido a un proceso de motivación para llevar a cabo una actividad de comportamiento). Cada uno de los

agentes del nivel reactivo está unido a un objeto de la pizarra intermedia. La plataforma de esquemas y comunicación es la capa intermedia entre los dos niveles anteriores. Proporciona la infraestructura para acceder a los sensores y actuadores. El nivel deliberativo le envía los esquemas necesarios al nivel reactivo a través de esta plataforma intermedia y accede al valor de los sensores a través de esta capa intermedia e incorpora la información obtenida al modelo simbólico interno del entorno.

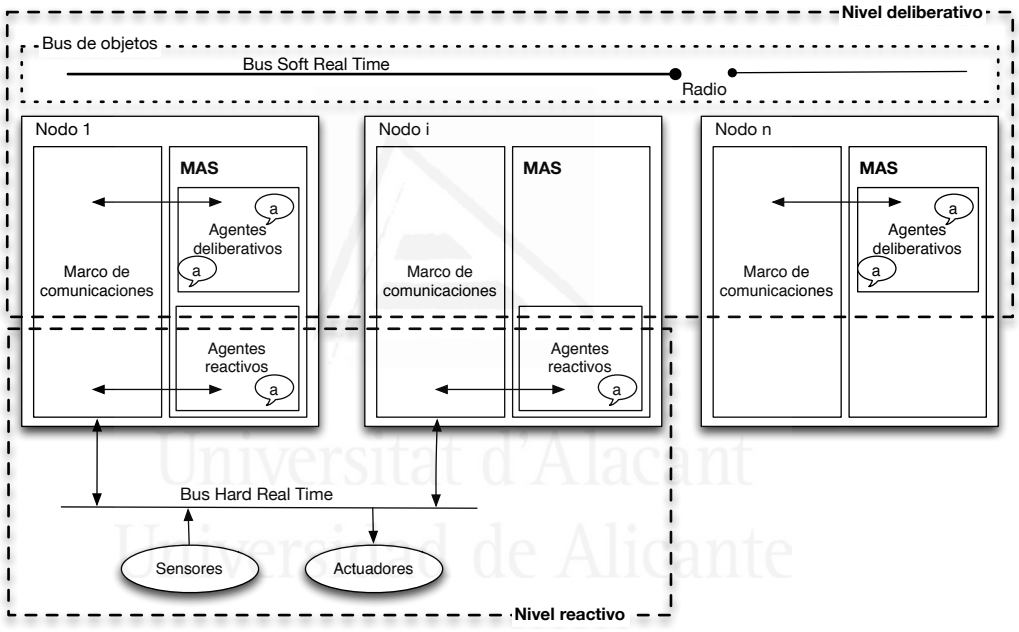


Figura 4.12: Ejemplo de arquitectura robótica basada en agentes: Arquitectura SC-Agent.

La plataforma intermedia de comunicaciones en la cual se basa es la columna vertebral de la arquitectura. Esta plataforma es el módulo que permite la interacción entre todos los agentes del sistema.

4.2. Arquitecturas multi-robóticas

En los últimos años se ha producido un creciente interés por el desarrollo de sistemas formados por múltiples robots autónomos, de manera que el comportamiento colectivo emerja a partir de las interacciones entre los diferentes individuos que forman el grupo. Este interés radica en que el hecho de

disponer de varios robots puede hacer al sistema más flexible, robusto y efectivo para el desarrollo de determinadas tareas, por su distribución intrínseca y su paralelismo en cuanto a sensores y actuadores [WZLZ10].

A continuación se describen algunas arquitecturas para sistemas multi-robóticos. Entre éstas se encuentran algunas muy referenciadas, como es el caso de ALLIANCE y arquitecturas definidas recientemente, como Lei o Marino.

Arquitectura ALLIANCE

ALLIANCE [Par98] (1998) es una arquitectura distribuida basada en comportamientos con tolerancia a fallos. Es una arquitectura que facilita el control cooperativo en un grupo de robots heterogéneo. Permite a los equipos de robots, cada uno de ellos dotado con un conjunto de funciones de alto-nivel, seleccionar individualmente la acción apropiada en cada momento basándose en los requerimientos de la misión, las actividades de otros robots, las condiciones actuales del entorno y el estado interno del robot. Para esta selección de la acción se proporciona a cada robot motivaciones modeladas matemáticamente, llamados *comportamiento motivacional*. En cada momento, cada comportamiento motivacional recibe información de los sensores, comunicación con otros robots, retroalimentación de otros comportamientos activos y motivaciones internas. Se definen dos tipos de motivaciones internas: impaciencia del robot(*robot impatience*) y consentimiento del robot (*robot acquiescence*). Las primeras motivaciones permiten al robot controlar situaciones donde los otros robots fallan en su tarea. Las segundas motivaciones permiten al robot controlar situaciones donde son sus propias acciones las que fallan.

Toda esta información se combina para generar la salida del comportamiento que define el nivel de activación de su correspondiente conjunto de comportamientos. Cada conjunto de comportamientos se corresponde con los niveles de competencia requeridos para llevar a cabo alguna tarea de alto-nivel.

Esta arquitectura ha sido extendida a L_ALLIANCE, donde se utiliza el aprendizaje por refuerzo para ajustar los parámetros de activación del conjunto de comportamientos.

ALLIANCE asume, por un lado, que un robot conoce las acciones que están llevando a cabo los otros robots del equipo y, por otro lado, que es capaz de detectar el fallo de otros robots. Estas suposiciones serían un problema difícil de tratar manteniendo las suposiciones de comunicación limitada o escalabilidad.

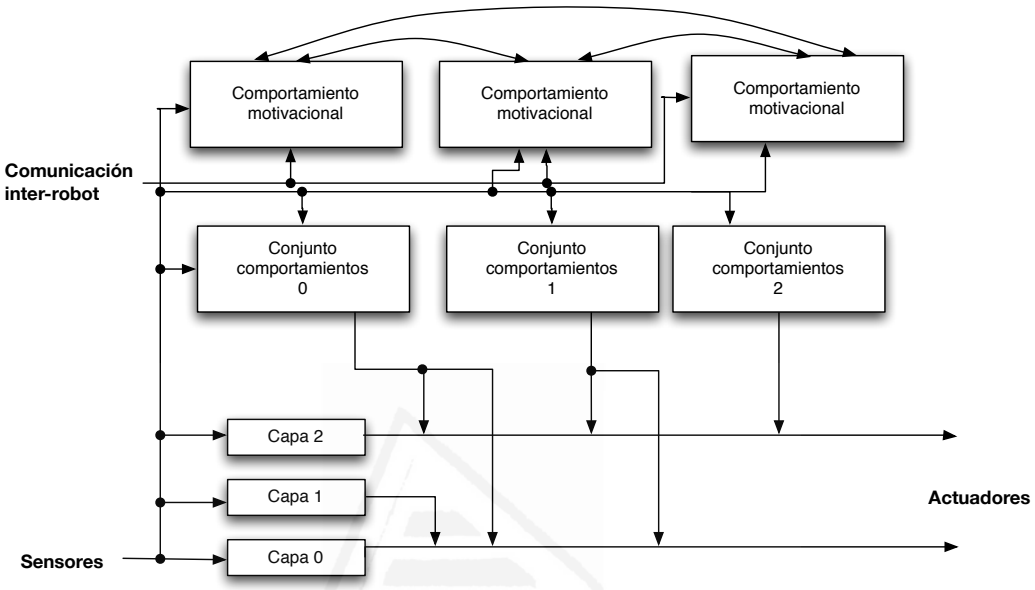


Figura 4.13: Ejemplo de arquitectura multi-robótica: Arquitectura Alliance. Este esquema es implementado en cada uno de los robots.

Arquitectura HEIR

La arquitectura HEIR [Pia99] (1999) se organiza en 3 componentes diferenciados por el tipo de conocimiento con el que tratan: componente simbólico (S), que trata con un formalismo declarativo, explícito y proposicional; un componente diagramático (D), que utiliza representaciones analógicas e icónicas que son más sintetizadas que las anteriores; y un componente basado en un comportamiento reactivo (R). Cuando un robot tiene que realizar una actividad compleja la mayoría de actividad se situará en el nivel simbólico, en el cual se representa la planificación explícita. Cuando tiene que realizar una tarea más sencilla la actividad se situará en el componente diagramático donde tiene que hacer pequeños razonamientos sobre la información sensorial. Si tiene que realizar tareas de evitación de obstáculos se ejecutará el componente de comportamientos reactivos. No es una arquitectura con una organización jerárquica, como se puede observar en la figura 4.14, sino que el foco de la actividad estará centrada en diferentes componentes en diferentes momentos. Cualquiera de los tres componentes puede ocupar el nivel superior dependiendo de los eventos internos o externos.

En el componente simbólico los agentes operan a una base de conoci-

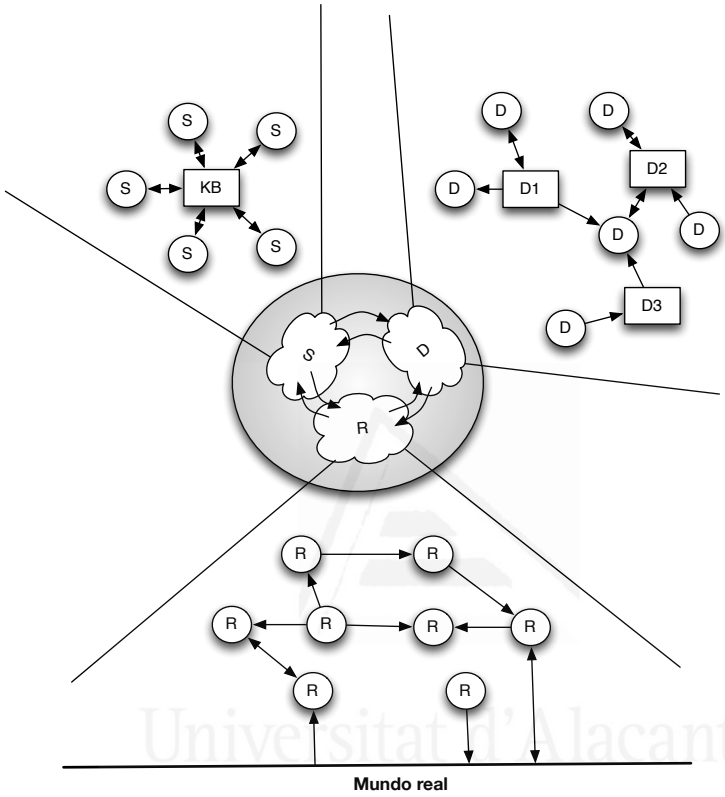


Figura 4.14: Ejemplo de arquitectura multi-robótica: Arquitectura HEIR.

miento común llamada KB para llevar a cabo todas sus actividades. En la componente diagramático los agentes se basan en representaciones sintetizadas nombradas DN que deben ser definidas previamente.

Arquitectura Goldberg

En [GCD+03] presentan una arquitectura (2003) de tres niveles definida para cada uno de los robots individuales. Cada una de las capas puede interactuar directamente con la misma capa de otros robots, como se puede observar en la figura 4.15. De esta manera, cada robot puede actuar de manera autónoma, pero puede coordinarse a diferentes niveles con otros robots. En el nivel de conducta coordinan los comportamientos; en el nivel de ejecución sincronizan la ejecución de tareas; y en el nivel de planificación utilizan técnicas basadas en mercado para asignar las tareas y localizar recursos.

La capa de comportamientos crea bucles de comportamientos conectando

los comportamientos sensitivos de un robot con los comportamientos actuadores de otro. La capa de ejecución descompone jerárquicamente las tareas en subtareas y monitoriza la ejecución de tareas. La capa de planificación basada en estudios de mercado tiene dos componentes: un agente de mercado que participa en la subasta de tareas y un planificador que determina la viabilidad de la tarea y su coste, e interactúa con la capa de ejecución para llevar a cabo las tareas.

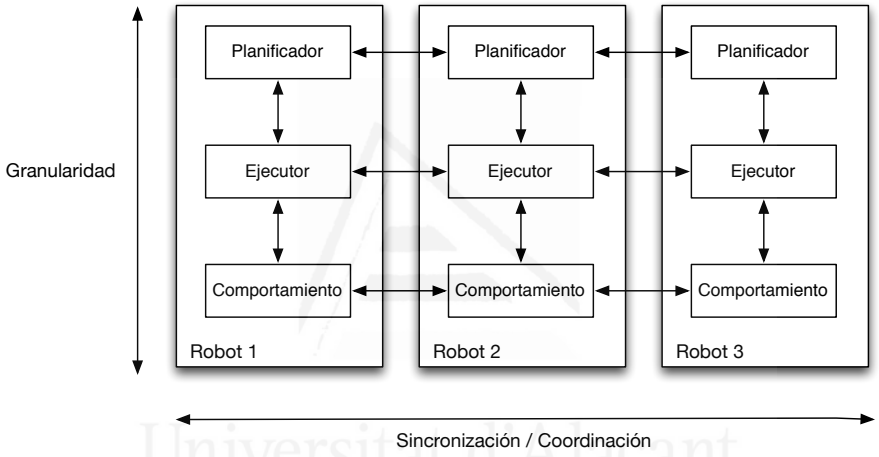


Figura 4.15: Ejemplo de arquitectura multi-robótica: Arquitectura definida por D. Goldberg en [GCD+03].

En esta arquitectura todos los robots pueden interactuar entre sí y, además, a diferentes niveles, lo que conlleva una carga de comunicación importante en el sistema.

Arquitectura Lei

En [LZF10] (2010) presentan una arquitectura jerárquica para la planificación de movimiento de un sistema multi-robótico. Esta arquitectura se divide en cuatro capas: sistema de monitorización, planificación de la misión, coordinación del movimiento y control de comportamiento. La capa superior *Capa del sistema de monitorización* está pensada, principalmente, para la teleoperación, aunque es una tarea que puede realizar una máquina. Cuando está dirigida por una máquina es una capa que sólo entra en funcionamiento si ocurre una excepción, conflicto o bloqueo. La capa de planificación de la misión, es donde se encuentra el robot de control. Éste lleva a cabo la planificación de la misión, distribuyendo a los robots coordinadores el punto de

partida y el objetivo. Serán los robots coordinadores los que distribuyan esta información a los robots autónomos. La capa de coordinación de movimiento es donde se lleva a cabo la planificación del camino. Esta tarea es llevada a cabo por los robots coordinadores que además de realizar la planificación también llevan a cabo la planificación de la evitación de obstáculos. La capa de control de comportamiento está formada por los robots autónomos que llevan a cabo la planificación recibida.

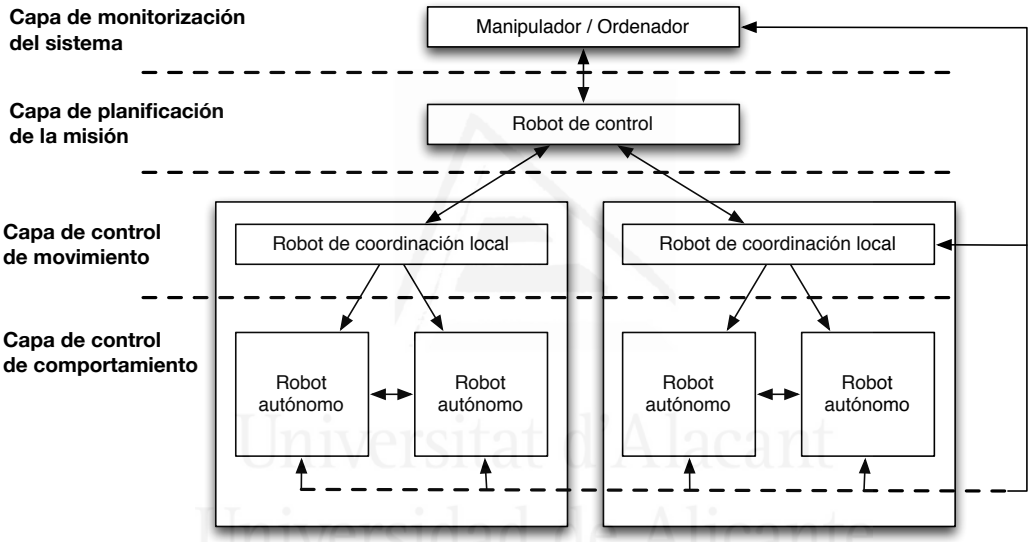


Figura 4.16: Ejemplo de arquitectura multi-robótica: Arquitectura definida por Y. Lei en [LZF10].

Aunque en el artículo se indica que es una arquitectura descentralizada que soporta la tolerancia a fallos y la escalabilidad porque está permitida la comunicación entre robots, depende de un robot central que lleva a cabo la planificación global. Esta dependencia puede hacer que todo el sistema sea vulnerable ante el fallo de este robot central.

Arquitectura Marino

En [MPAC12] (2012), A. Marino presenta una arquitectura de control para un sistema multi-robótico con una misión de vigilancia. Esta arquitectura ha sido desarrollada en el modelo de control reactivo conocido como *Null-Space-Based Behavioral Control* (NSB). Este método consiste en dividir una tarea compleja en tareas más sencillas denominadas comportamientos,

las cuales se ordenan de acuerdo a su prioridad que no puede ser cambiada dinámicamente por restricciones en el entorno o en la tarea. En esta arquitectura se intenta evitar este inconveniente. Es una arquitectura dividida en tres niveles, como se muestra en la imagen 4.17. En la capa inferior se sitúan los agentes (unidades robóticas individuales) que llevan a cabo la tarea. En la capa intermedia se definen los comportamientos elementales que posteriormente son combinados, por NBS, en acciones más complejas. La capa superior de supervisión está a cargo de seleccionar dinámicamente la acción apropiada para ser ejecutada.

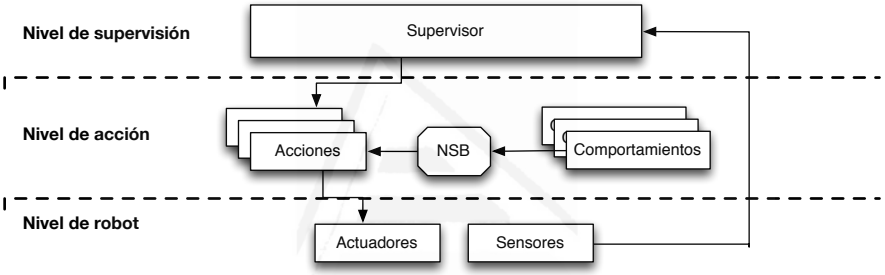


Figura 4.17: Ejemplo de arquitectura multi-robótica: Arquitectura definida por A. Marino en [MPAC12].

En la documentación de esta arquitectura se indica que es descentralizada y sin uso de comunicación, que es robusta y previene colisiones, pero se asume que los robots conocen en todo momento su localización y tienen una descripción del entorno cercano.

4.3. Análisis comparativo

Para dotar a los robots autónomos de las capacidades y funcionalidades necesarias para llevar a cabo determinadas tareas, es adecuado definir una arquitectura que describa los diferentes componentes del sistema y cómo se organizan e interactúan éstos.

Como se comentó en el capítulo 2 las características de la robótica de enjambre derivadas de su inspiración en los insectos sociales son: robustez, flexibilidad y escalabilidad. Además, para que un sistema multi-robótico sea considerado apropiado para la robótica de enjambre, debe cumplir una serie de características: debe estar formado por un elevado número de robots autónomos; y estos robots deben ser relativamente simples e ineficientes con

sensores locales y mecanismos de comunicación limitada. Estas características obligan a que la cooperación del grupo sea esencial. Teniendo en cuenta estas propiedades, a continuación se describen las características tenidas en cuenta para analizar las arquitecturas estudiadas.

- **Robustez (o tolerancia a fallos):** donde se mide el grado por el cual un sistema puede seguir en funcionamiento ante la presencia de fallos parciales o condiciones anormales en cualquier parte del sistema. Se diferencia entre sistemas que no soportan tolerancia a fallos (No), sistemas que tienen algunos mecanismos de tolerancia a fallos (S/N) y sistemas que tienen una alta robustez (Sí).

- **Flexibilidad:** se tiene en cuenta la capacidad del sistema de adaptarse a entornos nuevos, situaciones inesperadas o cambios en los requerimientos.

- **Escalabilidad:** se mide la facilidad para expandir el sistema dando soporte a un número mayor (o menor) de componentes sin afectar a su funcionamiento. Es decir, los mecanismos de coordinación deben soportar cambios en el tamaño del sistema. En las arquitecturas multi-robóticas se tendrá en cuenta también que el sistema debe permitir trabajar con un número elevado de robots, por lo tanto, no deben existir mecanismos que puedan sobrecargarse ante esta situación. Se distingue entre sistemas que posean una baja escalabilidad (B) y sistemas con una alta escalabilidad (A).

- **Coordinación:** se diferencia entre sistemas centralizados (C) y descentralizados (D).

- **Comportamiento:** especifica si el comportamiento es emergente (E) o por el contrario es un comportamiento establecido (S).

- **Comunicación:** indica si es necesaria la comunicación explícita entre los diferentes componentes del sistema.

- **Capacidad limitada:** la robótica de enjambre enfatiza que los robots que forman el sistema deben ser robots relativamente simples, con sensores locales y habilidades de comunicación limitadas. Esta característica especifica si la arquitectura está pensada para este tipo de robots o por el contrario está diseñada para robots con mayores capacidades.

Las arquitecturas definidas para un único robot siguen uno de los tres paradigmas establecidos según las tres primitivas básicas (sentir, planificar, actuar): deliberativo, reactivo e híbrido. Este último es donde es posible enmarcar la gran parte de arquitecturas ya que intenta obtener las ventajas de los dos paradigmas anteriores y reducir sus puntos débiles.

La mayoría de estas arquitecturas se estructuran (o pueden estructurarse) en capas o niveles, cada uno de ellos formado por varios módulos. Esta división en módulos hace que la mayoría de arquitecturas incluyan algún mecanismo genérico de robustez, ya que el fallo de un módulo no impediría el funcionamiento del resto del sistema, siempre y cuando no fuera un módulo indispensable. Algunas de estas arquitecturas incluyen directamente mecanismos de tolerancia a fallos como SFX [MA92] que permite conocer la fiabilidad de los sensores, ALLIANCE [Par98] que permite controlar las situaciones donde fallan otros robots o donde fallan las acciones del propio robot o Lei [LZF10] que permite la comunicación directa entre robots.

Todas estas arquitecturas disponen de un módulo planificador o de supervisión que modifica los comportamientos según las necesidades establecidas, por lo tanto, en mayor o menor medida, todas las aproximaciones son capaces de adaptarse a cambios en los requerimientos o demuestran algún grado de flexibilidad.

Un aspecto importante a tener en cuenta en el diseño de una arquitectura para robótica de enjambre es la escalabilidad del sistema. Muchas de estas arquitecturas tienen una escalabilidad media o baja donde la inclusión de nuevos módulos supondría algunos cambios significativos en los existentes como en Yavuz [YB02], 3T [BKJJ98] o Busquets [BSdM03]. Otras arquitecturas permiten, de manera sencilla, únicamente la adición de nuevos comportamientos como AuRA [AB97], SFX [MA92], Saphira [KMRS97] o Innocenti [ILS07]. Algunas de las arquitecturas multi-robóticas permiten la escalabilidad del sistema, pero los sistemas de comunicación utilizados pueden suponer una sobrecarga cuando se trabaja con un número elevado de robots como en ALLIANCE [Par98] o en Goldberg [GCD+03]. Por otro lado, Lei [LZF10] o Marino [MPAC12] ofrecen una alta escalabilidad.

Las arquitecturas robóticas individuales estudiadas tienen una coordinación centralizada. Algunas de ellas, incluidas las basadas en agentes, a pesar de ser arquitecturas distribuidas, están basadas en, al menos, un módulo central necesario para la coordinación como BERRA [LOC00], Saphira [KMRS97], Busquets [BSdM03], Innocenti [ILS07] o SC-Agent [PPS+08]. Entre las arquitecturas multi-robóticas estudiadas algunas tienen una coordinación descentralizada como ALLIANCE [Par98], Goldberg [GCD+03] o Marino [MPAC12]. Lei [LZF10], sin embargo, se basa en un robot central

que lleva a cabo toda la planificación y en HEIR [Pia99] la coordinación depende del tipo de conocimiento necesario.

Las arquitecturas robóticas para un único robot muestran un comportamiento global con una baja emergencia, dado que este comportamiento surge a partir de una planificación establecida. Sin embargo, en las arquitecturas multi-robóticas es posible distinguir cuando el comportamiento global del sistema es totalmente emergente, donde el comportamiento global del sistema no puede ser exhibido por un único robot individual, como en ALLIANCE [Par98] o Goldberg [GCD+03].

En las arquitecturas individuales la comunicación entre los diferentes componentes del sistema es necesaria en mayor o menor medida. Nos centraremos en la arquitecturas multi-robóticas y en la comunicación entre los diferentes robots que forman el sistema. En ALLIANCE [Par98] para llevar a cabo la suposición de que un robot conoce las acciones que están llevando a cabo el resto de robots del sistema hace necesaria la comunicación entre ellos. En HEIR [Pia99] según el tipo de conocimiento con el que trabaje la arquitectura será necesaria un grado de comunicación mayor o menor. Goldberg [GCD+03] define de manera explícita la comunicación entre todos los robots del sistema a todos los niveles. En Lei [LZF10] es necesaria la comunicación entre robots, tanto entre el robot central y los coordinadores, como entre el resto de robots del sistema. En Marino [MPAC12] se indica que la arquitectura no necesita comunicación pero suponen que cada robot conoce el entorno cercano y su localización exacta.

Una de las características de la robótica de enjambre que no resulta significativa en las arquitecturas robóticas para un único robot es la capacidad limitada. En éstas el robot debe ser relativamente simple con habilidades limitadas. Las arquitecturas individuales tratan con sistemas sensoriales complejos, como visión o sistemas de localización. Respecto a las arquitecturas multi-robóticas estudiadas, en éstas no se indica los tipos de capacidades y habilidades de los que se compone un robot. Incluso en algunas de ellas, las suposiciones que realizan no podrían llevarse a cabo con capacidades de comunicación limitadas como en ALLIANCE [Par98], Goldberg [GCD+03] o Lei [LZF10].

La tabla 4.1 muestra un resumen de estas características.

4.4. Definición del modelo de arquitectura

Debido a las características específicas que debe cumplir un sistema robótico de enjambre, no es una tarea sencilla la implementación de una arquitec-

	Arquitecturas \ Características	Robustez	Flexibilidad	Escalabilidad	Modularidad	Coordinación	Comportamiento	Comunicación	Limitación
Robóticas	AuRA	S/N	S	B	S	C	S	S	N
	SFX	S	S	B	S	C	S	S	N
	Yavuz	S/N	S	B	S	C	S	S	N
	Tripodal	S/N	S	B	S	C	S	S	N
	3T	S/N	S	B	S	C	S	S	N
	BERRA	S/N	S	B	S	C	S	S	N
	Saphira	S/N	S	B	S	C	S	S	N
	TCA	S/N	S	B	S	C	S	S	N
	Busquets	S/N	S	B	S	C	S	S	N
	Innocenti	S/N	S	B	S	C	S	S	N
	SC-Agent	S/N	S	B	S	C	S	S	N
Multi-robóticas	ALLIANCE	S	S	B	S	D	E	S	N
	HEIR	S/N	S	B	S	C/D	S	S	N
	Goldberg	S/N	S	B	S	D	E	S	N
	Lei	S	S	A	S	C	S	S	N
	Marino	S/N	S	A	S	D	S	N	N

Cuadro 4.1: Resumen del cumplimiento de las características de la robótica de enjambre por las arquitecturas robóticas y multi-robóticas descritas.

tura totalmente distribuida, sin ningún mecanismo de control centralizado, robusta y con alta independencia de todos sus componentes, para el control de un grupo de robots con capacidades limitadas y que además deben ser capaces de desarrollar una tarea compleja.

Si intentamos modelar un sistema de enjambre robótico con las ideas obtenidas de las arquitecturas vistas en el punto anterior nos encontramos con una serie de problemas.

Las arquitecturas híbridas anteriores, establecen procesos deliberativos de alto nivel, que controlan a procesos reactivos de bajo nivel, o bien directamente (como en las arquitecturas organizativas) o bien indirectamente (como en las arquitecturas con modelo del mundo). Estos procesos están

pensados para un sistema centralizado y por tanto son difíciles de implementar en un sistema totalmente descentralizado como exige la robótica de enjambre.

Por otra parte, la robustez a fallos suele no estar contemplada a nivel de plataforma en las arquitecturas anteriores. De esta manera si un módulo deliberativo falla, el control del robot se pierde. Esto no sería tolerable en el caso de la robótica de enjambre, ya que el fallo de un módulo del sistema (por ejemplo un planificador deliberativo) no puede poner en riesgo la estabilidad de todo el sistema (el fallo de un módulo no puede poner en riesgo a todo el enjambre que puede estar formando por decenas de robots).

Las arquitecturas anteriores suelen contemplar el uso de robots con sensores complejos y capacidades de comunicación no limitadas. La robótica de enjambre exige que los robots que forman el sistema deben tener capacidades limitadas, incluyendo los mecanismos de comunicación. Esta característica ayuda a que el sistema sea totalmente escalable con mecanismos de coordinación sencillos que pueden soportar grandes tamaños.

Teniendo en cuenta todas estas propiedades y las vistas en el capítulo 3 donde se comenta que la robótica de enjambre y los sistemas multi-agente comparten varias características, y que estos últimos aportarían una serie de ventajas al sistema como la distribución de datos y control, la integración, mejora de la eficiencia, escalabilidad, flexibilidad y reutilización; en este punto se presenta un modelo para un sistema de enjambre basado en un sistema multi-agente que cumple las características exigidas por la robótica de enjambre.

4.4.1. Descripción general

Para la definición de comportamientos para el control de un sistema robótico de enjambre se define un modelo de arquitectura híbrida organizativa. A grandes rasgos este modelo se compone principalmente de dos niveles bien diferenciados, como se puede observar en el diagrama mostrado en la figura 4.18. El nivel inferior define el enjambre puro. Está formado por los robots que cumplen de manera estricta todas las restricciones que señala la robótica de enjambre. Es decir, está formado por un gran número de robots homogéneo, con capacidades de comunicación muy limitadas, lo que conlleva que los robots trabajen con información local. En este nivel las tareas están fuera del alcance de un único robot y, por tanto, es necesaria la cooperación para alcanzar el comportamiento global deseado. Además, en este grupo de robots no existe un mecanismo de control centralizado. Todas estas características proporcionan al enjambre las propiedades básicas de robustez, flexibilidad y

escalabilidad.

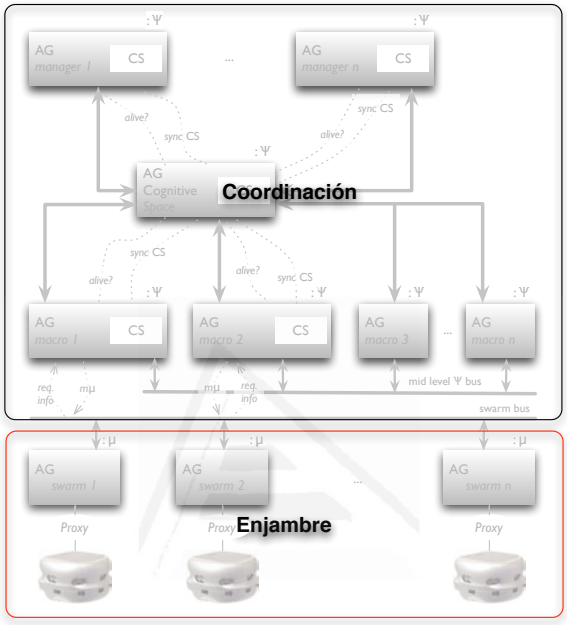


Figura 4.18: Diagrama básico. El nivel inferior estaría formado por el enjambre puro. El nivel superior lo formarían los agentes de coordinación.

Robustez debido a que el fallo de un agente no afectará al resto del sistema, permitiendo que éste continúe en funcionamiento. Flexibilidad por la emergencia implícita en el comportamiento del enjambre que hace que éste pueda adaptarse a posibles cambios. Y escalabilidad porque estas características permiten que la agregación o eliminación de un agente en el sistema no afecte a su rendimiento.

En muchas ocasiones es complicado realizar tareas complejas utilizando únicamente un grupo de robots que cumplen estrictamente las características de la robótica de enjambre. Bien porque sea necesario algún tipo de deliberación sobre la tarea que se está llevando a cabo, o bien porque sea necesario algún mecanismo de nivel superior para poder unir la información de los diferentes robots que forman el enjambre para obtener conocimiento de más alto nivel.

Por este motivo, se define una capa superior, sobre la capa de enjambre puro, para complementar las funcionalidades de enjambre. Esta capa tiene funciones de deliberación, coordinación y control del enjambre. Se divide

en dos niveles, lo que ofrece una arquitectura de tres capas siguiendo la definición de muchos sistemas híbridos, como se muestra en la figura 4.19: una capa superior deliberativa, una capa inferior reactiva (enjambre) y una capa intermedia entre estas dos. En este tipo de sistemas la planificación y la acción deben estar correctamente acopladas ya que el planificador debe guiar la reacción configurando y estableciendo los parámetros de los módulos de control reactivos.

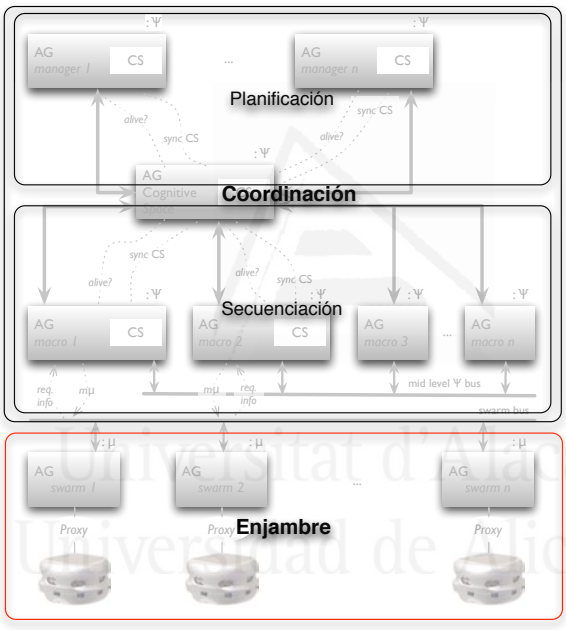


Figura 4.19: Diagrama de niveles. El nivel inferior es un nivel reactivo, el enjambre; mientras que el nivel superior es la capa de coordinación de la arquitectura formada por dos niveles.

De manera general, en este tipo de arquitecturas organizativas, la capa superior realiza la planificación de alto nivel. Esta planificación es pasada al nivel inferior, que se encarga de refinar ese plan y reunir a los recursos necesarios para llevarlo a cabo. Así mismo, éste le pasa la información a los individuos de más bajo nivel, que llevan a cabo los comportamientos más básicos.

En las arquitecturas organizativas los agentes de alto nivel verán los resultados de sus subordinados de nivel inferior y les darán directrices. Como en la arquitectura de subsunción, una capa sólo puede modificar la capa bajo

ella. Cada capa intenta llevar a cabo sus directivas, detectar problemas y corregirlos de manera local. Sólo cuando un agente no es capaz de resolver sus propios problemas pedirá ayuda a un agente superior.

En la capa superior se encuentra la parte deliberativa de la arquitectura formada por agentes de más alto nivel. Estos agentes deciden la tarea que debe realizar el sistema, es decir, cuál sería el comportamiento global deseado. Esta información se pasa al nivel intermedio.

Los agentes de la capa intermedia se encargan de reclutar a los agentes de más bajo nivel indicándoles la tarea que deben realizar. Estos agentes controlarán al enjambre y lo dirigirán hacia el objetivo.

Por último, la capa inferior es la parte reactiva, por lo tanto, los agentes de esta capa son los encargados de realizar las tareas del enjambre más cercanas a la reactividad.

Esta división entre acción y reacción permite que el nivel inferior, el enjambre puro, sea utilizada aparte, en solitario, para aplicaciones puramente de enjambre. Es decir, para tareas donde el comportamiento global deseado emerge únicamente de la interacción local entre los agentes reactivos, donde no es necesario el uso de ningún agente deliberativo de más alto nivel, cumpliendo puramente las exigencias de la robótica de enjambre.

4.4.2. Sistema de agentes

Para esta arquitectura en niveles se ha definido un sistema multi-agente con tres tipos de roles diferentes que desarrollarán los agentes del sistema. Según el rol que tenga cada agente, éste tendrá una ubicación diferente y podrá realizar una serie de tareas específicas de ese rol. Los roles definidos son μ, Ψ *nivel medio* y Ψ *alto nivel*. Mientras que los agentes μ están enfocados en el control de los robots físicos, los dos tipos de agentes Ψ, son agentes cognitivos de más alto nivel y se centran en el control de todo el enjambre. Además, se ha definido un agente controlador CS (*Cognitive Space*) que gestiona y controla el correcto funcionamiento de los agentes Ψ, además de proporcionarles un espacio de comunicación e intercambio de información.

La figura 4.20 muestra la distribución de los agentes en los diferentes niveles de la arquitectura. En la parte inferior se encuentran los agentes μ, cercanos a los robots físicos, en la parte central se encuentran los agentes Ψ *nivel medio* que guían a los anteriores hacia la consecución de la tarea y en el nivel superior se encuentran los agentes Ψ *alto nivel* que se encargan del control de todo el enjambre. El agente controlador CS se encuentra entre los dos anteriores.

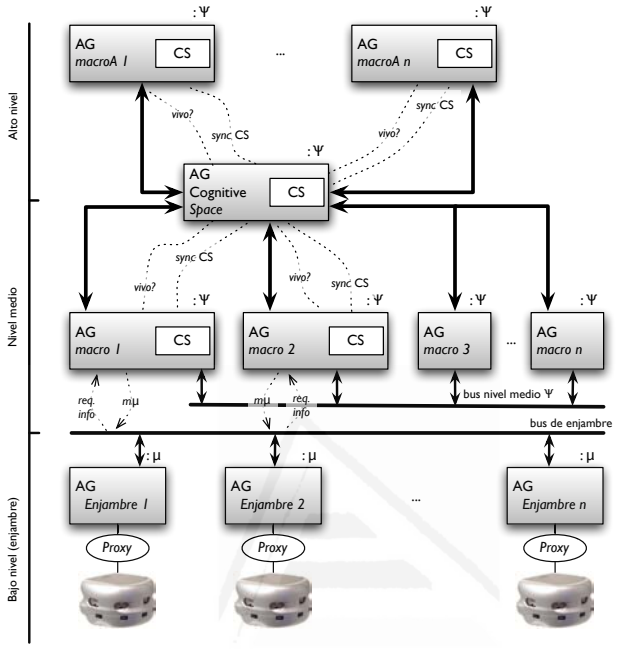


Figura 4.20: Diagrama de la arquitectura propuesta. Se observan los diferentes tipos de agentes y los canales de comunicación más importantes. Los agentes μ en el nivel inferior, los agentes Ψ *nivel medio* en el nivel medio y los agentes Ψ *alto nivel* en el nivel superior.

4.4.3. Los agentes μ

Este tipo de agentes son los más simples del sistema. Son agentes de bajo nivel que se encargan de controlar a los robots físicos, y normalmente estarán situados en ellos. Por lo tanto, además del control de los robots, se encargarán de realizar las tareas del enjambre que están cercanas a la reactividad y que cumplen de manera estricta las propiedades de la robótica de enjambre.

Estos agentes podrán ser utilizados de manera aislada para realizar tareas de enjambre donde el comportamiento deseado emerge únicamente a partir de la interacción de los robots con el entorno.

Si fuera necesario estos agentes se podrán comunicar, por un lado, con los agentes Ψ *nivel medio* ya que recibirán de éstos la tarea que deben realizar y además, les comunicarán el estado de la tarea cuando estos agentes se lo requieran. Por otro lado, se podrán comunicar también con otros agentes μ

si en el desarrollo de su tarea fuera necesario. Esta comunicación punto a punto con otros agentes de su mismo tipo evita que se sobrecargue el sistema. Además, estos agentes podrán utilizar los servicios de páginas blancas AMS y páginas amarillas DF para la búsqueda de otros agentes μ.

4.4.4. Los agentes Ψ

Los agentes Ψ *nivel medio*

Los agentes Ψ son agentes cognitivos que realizan tareas de más alto nivel para controlar todo el enjambre y dirigirlo hacia su objetivo. Estos agentes no tienen por qué estar situados en un robot físico, pueden estar en uno de ellos o en otra máquina del sistema. El número de agentes Ψ *nivel medio* será menor que el número de agentes μ ya que para la coordinación del enjambre, y por tanto de los agentes μ, es necesario un número menor de agentes.

Estos agentes, tienen mayor complejidad que los anteriores ya que, por un lado, se comunicarán con los agentes Ψ *alto nivel* que les indicarán la tarea a nivel macroscópico que debe realizar el enjambre y para informarles del estado de la tarea y del enjambre, bien cuando el agente superior se lo requiera o bien cuando haya un cambio en la tarea. Una vez el agente Ψ *alto nivel* le ha indicado la tarea de todo el enjambre, debe determinar qué tarea debe realizar cada robot para alcanzar esta tarea global y por tanto, se comunicarán con los agentes μ para indicarles la tarea que deben realizar a nivel microscópico y para obtener información sobre el estado de la tarea que está ejecutando el robot con el fin de monitorizar y controlar dichas tareas. Además, también se comunicará con el agente CS para acceder a información compartida con otros agentes Ψ.

Los agentes Ψ *alto nivel*

Como se ha comentado anteriormente, estos agentes, son agentes cognitivos encargados de controlar todo el enjambre y dirigirlo hacia su objetivo. Estos agentes deciden la tarea que debe realizar todo el enjambre a nivel macroscópico y controlan que se lleve a cabo. Como en el caso de los agentes Ψ *nivel medio*, estos agentes pueden estar situados en cualquier máquina del sistema y su número es menor que los dos tipos de agentes anteriores.

Se comunicarán con los agentes Ψ *nivel medio* para, por un lado, comunicarles la tarea que el enjambre debe realizar y por otro lado para conocer el estado de la tarea. Se comunicarán también con el agente CS, no sólo para acceder a información compartida, sino también para comprobar la

existencia de éste y asegurar su presencia en el sistema, creándolo si éste desapareciera del sistema. Los agentes Ψ *alto nivel* no podrán comunicarse con los agentes μ.

En la figura 4.20 se observan las posibles vías de comunicación entre los agentes. Los agentes μ pueden comunicarse entre ellos y con los agentes Ψ *nivel medio* pero no con los agentes Ψ *alto nivel* que únicamente pueden comunicarse con otros agentes Ψ.

4.4.5. Agente Controlador CS

La robótica de enjambre especifica que no debe existir en el sistema ningún módulo centralizado que se encargue del control de los diferentes componentes para que no se pueda comprometer el funcionamiento de todo el enjambre por un único módulo. Pero debido a la importancia que tienen los agentes Ψ encargados del control de todo el enjambre, es necesario especificar un mecanismo de control que asegure la permanencia de estos agentes en el sistema [1].

Para ello, se ha definido un agente controlador (CS) encargado de gestionar y almacenar el estado de todos los agentes Ψ del sistema. Este agente tiene principalmente cuatro funciones:

- Proporciona un espacio común donde los diferentes agentes Ψ (tanto de alto nivel como de nivel medio) pueden compartir información.
- Realiza tareas de verificación de tolerancia a fallos. El agente CS controlará la existencia de todos los agentes Ψ del sistema, de manera que si un agente de este tipo *cae*, el agente CS se encargará de volverlo a crear. Para que la pérdida de información sea mínima y la restauración del agente inmediata, los agentes Ψ deben actualizar sus parámetros en el agente CS cada vez que estos varíen.
- Comprueba la duplicidad del sistema. El agente CS comprobará si dos agentes están realizando la misma tarea, si es así y uno de ellos es prescindible se encargará de su eliminación.
- Distribuye la información entre los agentes encargados de su control. Este agente distribuye la información que tiene almacenada entre los

[1] La replicación podría proporcionar una solución para la tolerancia a fallos de ciertas partes del sistema. No obstante, debido a las características de la robótica de enjambre, incluso la replicación de todos los agentes podría ser insuficiente, ya que si un agente Ψ y su réplica fallan, se comprometería el funcionamiento de todo el enjambre.

agentes Ψ *alto nivel*, de esta manera, los agentes cognitivos dispondrán de una copia del contenido del agente CS, con toda la información importante de los agentes cognitivos, pudiendo restaurar todo el sistema cognitivo sin pérdidas de información, en caso de ser necesario.

De esta manera, el control de tolerancia a fallos es doble. Por un lado, el agente CS verifica el correcto funcionamiento de todos los agentes Ψ pudiendo restaurarlos en el caso de caída de alguno de ellos. Y por otro lado, algunos agentes Ψ sincronizados con el CS verifican continuamente su estado y en caso de que este agente caiga, uno de ellos lo restaurará utilizando su copia interna del CS.

En la figura 4.20 se observa el agente CS situado entre los agentes Ψ *nivel medio* y los agentes Ψ *alto nivel*. Además, también se observa cómo algunos agentes Ψ *alto nivel* guardan una copia del CS para poderlo restaurar si fuera necesario.

Muchas de las funcionalidades de este agente CS podrían ser gestionadas a través de la plataforma *JADE* pero este agente ha sido definido con el fin de establecer un modelo general de control de un enjambre.

4.5. Implementación y funcionamiento de la arquitectura

Como se ha comentado en el capítulo 3 esta arquitectura se ha implementado utilizando la plataforma de agentes *JADE*. En ese capítulo se ha descrito el sistema de gestión de agentes de *JADE* y los mecanismos de tolerancia a fallos que ofrece (replicación del AMS y persistencia del DF). Se ha explicado la implementación y el funcionamiento de la plataforma y cómo se reconfigura su topología pasando a tener forma de anillo. Además, también se ha abordado el uso de una red *MANET* que permite la formación de una red distribuida auto-organizada y auto-administrada con la incorporación de módulos *XBee* que permiten utilizar el protocolo *DigiMesh*.

4.5.1. El motor de agentes *AgentEgine*

Para la definición de esta arquitectura en primer lugar se ha desarrollado un motor de agentes llamado *AgentEngine*. Es una librería situada por encima de *JADE* que facilita la creación de los diferentes tipos de agentes del sistema. Esta librería ha sido desarrollada utilizando el lenguaje de programación *Scala* [Hor12]. *Scala* es un lenguaje de programación que integra

características de los lenguajes orientados a objetos y de los lenguajes funcionales. Por un lado, cada valor es un objeto y su comportamiento se describe mediante clases y *traits*. Y, por otro lado, cada función es un valor. La programación funcional tiene una alta abstracción y por lo tanto, en general, el código suele ser más claro y conciso. Además, permite programar de manera sencilla procesos concurrentes donde el acceso a datos es fundamental.

AgentEngine permite la creación y gestión sencilla de un sistema multiagente utilizando como base *JADE*. La implementación de *AgentEngine* tiene un doble objetivo. Por una parte, se basa en el patrón de diseño fachada que ayuda a ocultar *JADE* al resto de componentes del sistema y, por lo tanto, proporciona una interfaz simple a un subsistema más complejo. Por otra parte, es un sistema de agentes modular, donde las funcionalidades disponibles para los agentes se organizan en módulos. De esta manera, cuando se desea que un agente desarrolle una funcionalidad determinada únicamente se le debe agregar el módulo correspondiente a dicha funcionalidad.

La librería *AgentEngine* se estructura principalmente en 4 paquetes:

- *agentengine*: donde se encuentra la clase de agente principal *AEAgent* de la cual deben derivar todos los agentes creados y el motor encargado de la gestión de la plataforma, llamado *AgentEngine*.

- *agentengine.modules*: donde se encuentran todos los módulos que proporcionan funcionalidades extras a los agentes. Por ejemplo *AEAgentModuleCyclic* que permite agregar un comportamiento cíclico al agente o *AEAgentModuleDFSubscriber* que permite realizar una suscripción al DF.

- *agentengine.ontology*: ofrece funcionalidades para crear ontologías de manera sencilla utilizando un tipo especial de clases de java llamadas *JavaBeans*.

- *agentengine.test.**: con ejemplos de uso de la librería.

4.5.2. Agentes básicos

En el motor se incluyen cuatro clases básicas para la creación de los cuatro tipos de agentes que forman la arquitectura: $MuAgent$, $PsiAgentM$, $PsiAgentH$, $CSAgent$.

MuAgent

Como hemos comentado en el diseño de la arquitectura, la clase $MuAgent$ (agente μ) permite crear los agentes que controlarán los robots físicos. Se creará un $MuAgent$ por cada robot del sistema. Esta clase se conecta a los robots físicos a través de un módulo *proxy*, que se vinculará al agente utilizando la facilidad de comunicación *object*2*agent* de *JADE*.

Este tipo de agentes son los más sencillos del sistema, al crearse dan de alta sus servicios (las tareas que pueden realizar, según sus componentes) en el DF para que los agentes $PsiAgentM$ (Ψ *nivel medio*) puedan encontrarlos. Estos agentes se componen de un comportamiento cíclico básico (*cyclic behaviour*) que controla el estado de las tareas y que responde a las peticiones realizadas por los agentes $PsiAgentM$.

PsiAgent

Los agentes $PsiAgentM$ (Ψ *nivel medio*) en primer lugar darán de alta sus servicios en el DF para que los agentes $PsiAgentH$ (Ψ *alto nivel*) puedan comunicarse con ellos e indicarles el objetivo y estado global de todo el enjambre. Además, estos agentes deberán estar suscritos al agente CS con el fin de sincronizar su contenido y acceder a la información compartida por todos los agentes Ψ. De esta manera, mediante al acto comunicativo *subscribe*, estos agentes establecen una suscripción con el agente CS, y cualquier cambio en su contenido les es notificado. Por otra parte, utilizan el DF para buscar los $MuAgent$ con el fin de comunicarse con ellos para indicarles la tarea a bajo nivel y obtener el estado de ésta.

Estos agentes ejecutarán un comportamiento cada cierto tiempo (*ticker behaviour*) con el fin de comunicarse tanto con los agentes μ como con los agentes Ψ para establecer, gestionar y controlar la tarea y el estado de todo el enjambre.

Como los agentes anteriores, los agentes $PsiAgentH$ se darán de alta en el DF. Estos agentes, utilizarán el DF para buscar los agentes $PsiAgentM$ para indicarles los objetivos globales del enjambre y controlar su estado.

Son agentes de alto nivel que además participan en tareas de sincronización y recuperación de información del CS. Para ello, deben suscribirse al CS para estar sincronizados con su contenido y deben realizar tareas de verificación periódica de este mismo agente. Utilizarán el acto de comunicación *subscribe* para suscribirse al CS, éste les notificará ante cualquier cambio en su contenido, y por tanto estarán sincronizados y la pérdida de información ante un fallo será mínima.

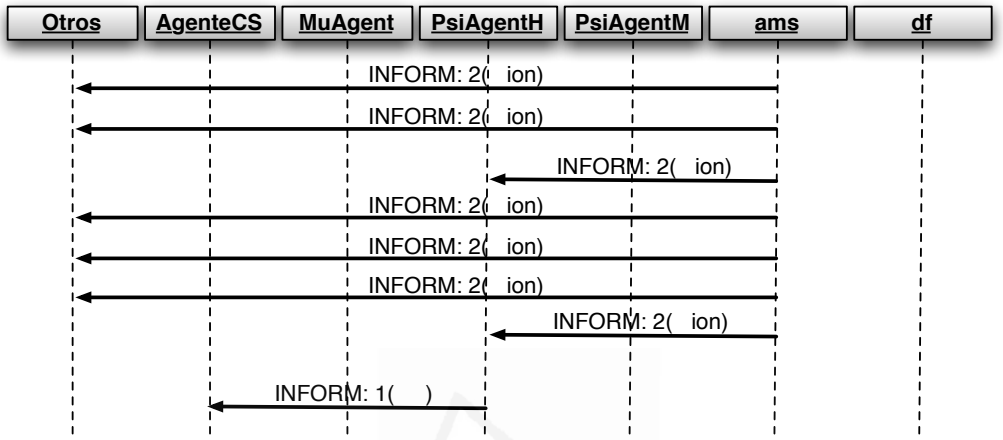


Figura 4.21: Intercambio de mensajes ante la caída del agente CS. El primer $inform$ que recibe $PsiAgentH$ desde el AMS es para notificarle la caída del agente CS mediante un evento *dead-agent*. El segundo $inform$ que recibe es para notificarle el nacimiento del nuevo agente CS mediante un *born-agent*. El agente $PsiAgentH$ le notifica al nuevo agente CS mediante un mensaje $inform$ el contenido de su base de datos.

Para verificar el estado del agente CS, comprobando que esté disponible, los agentes $PsiAgentH$ están suscritos a eventos de la plataforma, concretamente a la acción del JADE AMS *dead-agent*. En caso de que se detecte la caída del agente CS, el agente $PsiAgentH$ de mayor ID de los disponibles en el sistema será el encargado de solicitar la recreación del CS. El nuevo agente CS permanecerá a la espera, hasta recibir un mensaje de su creador, especificando el contenido actual de la base de datos. Una vez actualizado, reanudará su funcionamiento normal. En la figura 4.21 se observa el intercambio de mensajes ante la caída del agente CS. AMS le notifica la caída del agente CS a $PsiAgentH$ mediante un mensaje $inform$, $PsiAgentH$ volverá a crear el agente CS y AMS le notifica el nacimiento mediante otro mensaje $inform$, en ese momento el agente $PsiAgentH$ le envía un mensaje al CS con la información del enjambre.

Los agentes $PsiAgentH$, además, ejecutarán un comportamiento cada cierto tiempo (*ticker behaviour*) para verificar, controlar y gestionar el estado del enjambre.

La figura 4.22 muestra el intercambio de mensajes entre los tres tipos de agentes para la realización de una tarea del enjambre. Un agente $PsiAgentH$

le indica la tarea global del enjambre a un agente $PsiAgentM$ mediante un mensaje *request*. Los agentes confirman estos mensajes de petición mediante un mensaje *agree*.

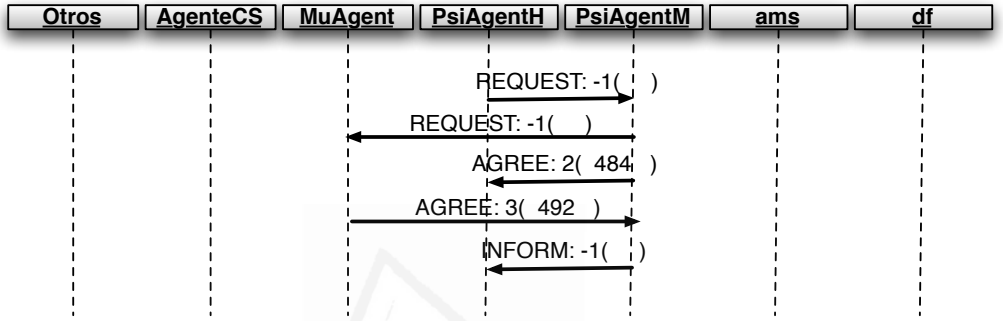


Figura 4.22: Intercambio de mensajes para la ejecución de una tarea. Cuando los agentes Ψ *alto nivel* deciden el comportamiento global del enjambre, se lo indican a los agentes Ψ *nivel medio* (*request*), éstos deliberan qué tarea de bajo nivel se debe realizar. Cada agente confirma la recepción del mensaje con un mensaje tipo *agree*.

Hemos visto cómo todos los agentes al inicio de su ejecución ($onInit$) dan de alta sus servicios en el DF. En la figura 4.23 se muestra un agente de cada tipo y se observa que el primer mensaje enviado por cada agente es un mensaje tipo *request* para dar de alta sus servicios en el DF, el siguiente mensaje tipo $inform$ lo envía el DF para confirmar esta alta.

En la figura 4.23 vemos cómo los agentes Ψ después de darse de alta en el DF le solicitan dos suscripciones, una de ellas al CS y otra a los $MuAgent$ en el caso de los agentes $PsiAgentM$ o a los $PsiAgentM$ en el caso de los $PsiAgentH$. Posteriormente estos agentes envían una solicitud de suscripción (*subscribe*) al agente CS confirmándoles éste la suscripción mediante un mensaje tipo *agree*.

CSAgent

Por último, la clase $CSAgent$, como en los casos anteriores, se da de alta en el DF. Esta clase se compone principalmente de dos comportamientos. Dispone de un comportamiento de respuesta a la suscripción (*subscription Responder*) encargado de notificar a los agentes la aceptación o denegación de la suscripción. Por otro lado, dispone de un comportamiento que se ejecu-

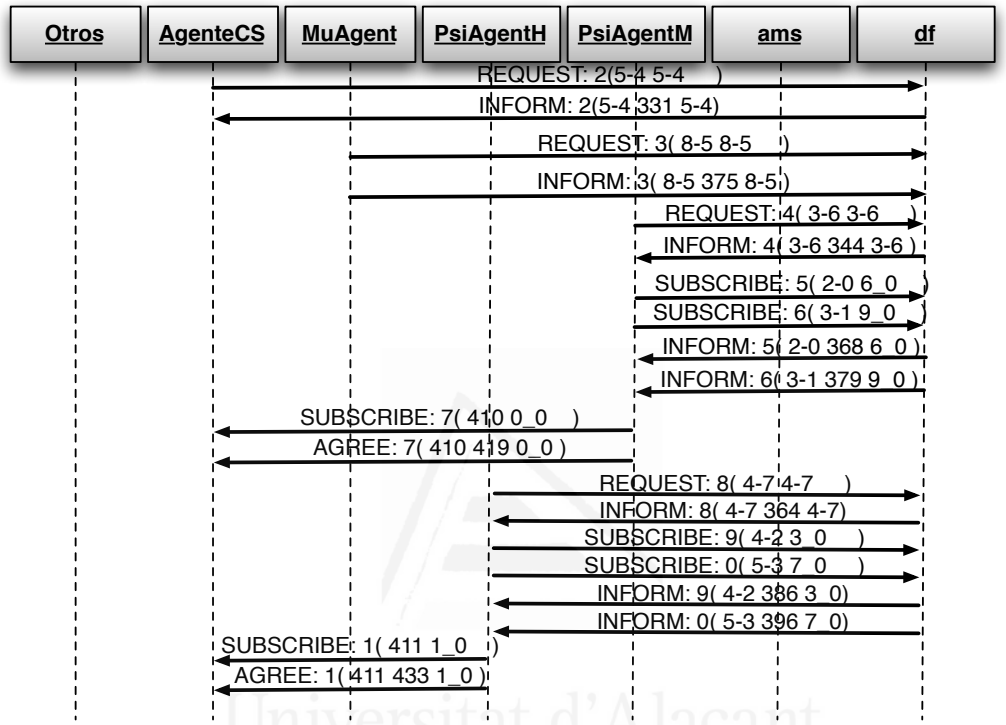


Figura 4.23: Intercambio de mensajes en la ejecución inicial de los agentes. En la figura se observa los mensajes iniciales de los agentes para dar de alta sus servicios en el DF, las peticiones al DF para buscar otros agentes y los mensajes de suscripción de los agentes Ψ al CS.

ta cada cierto tiempo (*ticker behaviour*) que informa a los agentes suscritos si se ha producido algún cambio en su información y además obtiene la información de los diferentes agentes Ψ para almacenarla en su base de datos interna. En la figura 4.24 podemos observar los mensajes que envía el CS a los agentes suscritos, además también se observa un mensaje enviado desde un agente $PsiAgentM$ al CS para que éste actualice el estado de ese agente en su base de datos interna.

El agente CS además debe verificar la permanencia de los agentes Ψ en el sistema, para ello, está suscrito a la acción del $AMS\ dead-agent$. Si detecta que un agente Ψ *cae*, la clase $CSAgent$ lo volverá a crear con la copia interna de ese agente.

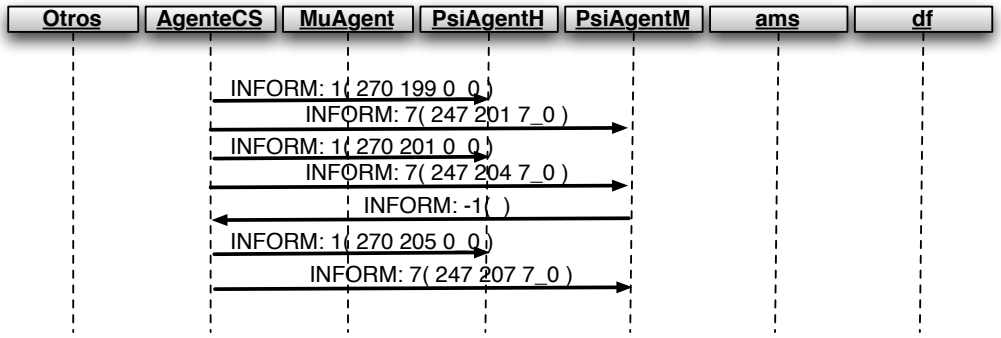


Figura 4.24: Mensajes que envía el CS a los agente suscritos y mensaje que envía un agente $PsiAgentM$ al CS ante un cambio en su estado.

4.6. Verificación del cumplimiento de las características de robótica de enjambre

Como se ha comentado en el punto 4.3 la robótica de enjambre tiene una serie de características que la distinguen de un sistema multi-robótico común. A continuación se analizarán estas características en base a la arquitectura propuesta:

- **Robustez (o tolerancia a fallos):** se cumplen los requisitos de tolerancia a fallos de un sistema de enjambre en dos niveles. En primer lugar, mediante mecanismos de verificación y control llevados a cabo por el agente CS que comprobará el estado de los agentes Ψ y por algunos agentes Ψ que comprobarán el estado del agente CS, que aseguran la permanencia en el sistema de los agentes importantes. Y, por otro lado, como se comentó en el capítulo 3, por el uso de los mecanismos de tolerancia a fallos de *JADE*, replicación del AMS y persistencia del DF, que aseguran el funcionamiento de la plataforma.

- **Flexibilidad:** el sistema tiene la capacidad de adaptarse a los diferentes cambios que se puedan producir. Por la emergencia en el comportamiento de los agentes μ es posible su adaptación a nuevos entornos; por el uso de una red *MANET* que utiliza el protocolo *DigiMesh* es posible actuar de manera adecuada ante una situación inesperada de pérdida de cobertura; o por el uso de agentes Ψ de más alto nivel, el sistema puede adaptarse a cambios en los requerimientos.

- **Escalabilidad:** se ofrece la posibilidad de añadir o eliminar agentes de cualquiera de los tipos en el sistema si la tarea lo requiere. Estos cambios se pueden llevar a cabo en cualquier momento sin afectar a su funcionamiento.

- **Coordinación:** se permite la definición de sistemas totalmente descentralizados y distribuidos para tareas que se pueden llevar a cabo con un comportamiento emergente. Si la tarea lo requiere permite también la definición de algún agente de control y cambiar la coordinación del sistema.

- **Comportamiento:** por el diseño de la arquitectura el comportamiento del enjambre puede ser totalmente emergente cuando únicamente entran en funcionamiento los agentes μ o puede ser un comportamiento más guiado cuando entran en funcionamiento los agentes Ψ.

- **Comunicación:** de manera similar a la característica anterior esta arquitectura puede trabajar sin comunicación explícita entre los diferentes componentes del sistema, cuando únicamente desarrollan la tarea los agentes μ y el comportamiento es emergente a partir de la información recibida por sus sensores, o por el contrario, puede existir también comunicación explícita entre agentes para tareas más complejas donde algún tipo de comunicación sea necesaria.

- **Capacidad limitada:** los agentes que componen la arquitectura están diseñados para controlar robots relativamente simples, con sensores locales que no requieren procesamiento y habilidades de comunicación limitadas.

4.7. Resumen

Debido a las características específicas que debe cumplir un sistema robótico de enjambre, no es una tarea sencilla el desarrollo de una arquitectura totalmente distribuida, sin ningún mecanismo de control centralizado, robusta y con alta independencia de todos sus componentes, para el control de un grupo de robots con capacidades limitadas y que además deben ser capaces de desarrollar una tarea compleja.

Por las características que comparten la robótica de enjambre y los sistema multi-agente, estos últimos pueden ser una buena alternativa para implementar un sistema robótico de enjambre. Sin embargo, éstos no se pueden

aplicar directamente debido a algunas características particulares de la robótica de enjambre como la naturaleza física de los robots, los mecanismos de comunicación distribuidos y las estructuras de control descentralizadas.

Además, para llevar a cabo tareas complejas es necesario agentes de alto nivel que establezcan los objetivos globales del sistema, en este caso de todo el enjambre. El modelo propuesto tiene en cuenta tanto las necesidades de alto nivel (agentes cognitivos) como las de bajo nivel (robótica de enjambre), proporcionando una arquitectura híbrida para la robótica de enjambre basada en agentes.

Se ha propuesto un modelo de arquitectura híbrida organizativa de tres capas: una capa deliberativa, una capa inferior reactiva y una capa intermedia entre estas dos. En la capa deliberativa se encuentran los agentes Ψ *alto nivel* encargados de establecer el comportamiento global del sistema. En la capa intermedia se encuentran los agentes Ψ *nivel medio* que controlan el enjambre y lo dirigen hacia el objetivo. En el nivel inferior se encuentran los agentes μ que realizan las tareas del enjambre cercanas a la reactividad.

Esta división entre deliberación y acción permite que la capa inferior, el enjambre puro, sea utilizada en solitario. Es decir, que la tarea sea realizada únicamente por los agentes μ, donde el comportamiento global del sistema emerge a partir de las interacciones de estos agentes con el entorno.

La arquitectura propuesta cumple los requisitos que establece la robótica de enjambre. Respecto a la tolerancia a fallos de un sistema de enjambre, mediante mecanismos de verificación y control llevados a cabo por diferentes agentes en un doble sentido, el agente CS comprobará el estado de todos los agentes Ψ y algunos agentes Ψ *alto nivel* comprobarán el estado del agente CS. Además, la arquitectura tiene en cuenta los mecanismos de tolerancia a fallos disponibles en $JADE$ (replicación del AMS y persistencia del DF). Por otro lado, también cumple con las características de flexibilidad, escalabilidad y capacidades limitadas. Por último, al permitir el uso únicamente de los agentes reactivos μ, la arquitectura cumple los requisitos de coordinación descentralizada, comportamiento emergente a partir de las interacciones locales y comunicación limitada.

Esta arquitectura servirá para dar soporte a los comportamientos de enjambre definidos en los siguientes capítulos.

Capítulo 5

Comportamientos básicos

En robótica de enjambre es posible encontrar una serie de problemas considerados como fundamentales ya que son precursores de otros comportamientos más complejos. Entre estos comportamientos se encuentran la agregación y dispersión.

El objetivo de esta tesis es la definición de comportamientos para el control de un enjambre robótico. En este capítulo se desarrollarán tres comportamientos fundamentales: agregación, movimiento coordinado (*flocking*) y dispersión, basados en comportamientos ya existentes.

El comportamiento de agregación hace referencia a la agrupación autoorganizada de los individuos del enjambre sin utilizar indicaciones del entorno. Éste puede ser considerado un comportamiento fundamental de un enjambre ya que es precursor de otros comportamientos más complejos.

El comportamiento de movimiento coordinado (*flocking*) hace referencia a la coordinación del movimiento de un grupo de robots, de manera que sean capaces de realizar un movimiento suave, en grupo, y capaces de evitar obstáculos.

El comportamiento de dispersión puede entenderse como un comportamiento opuesto a la agregación. Consiste en la obtención de una distribución uniforme de los robots en el espacio, de manera que maximicen el área cubierta.

Para realizar el estudio de estos comportamientos se ha utilizado el entorno de simulación *MASON* (*Multi-Agent Simulator Of Neighborhoods*) [LCRPS04]. Es un simulador de un sistema multi-agente, potente y versátil. Ha sido diseñado para soportar un gran número de agentes de manera eficiente en una única máquina.

El modelo de robot utilizado en el simulador es una simplificación de un

robot diferencial, por lo tanto, dispone de dos ruedas paralelas situadas en el mismo eje.

Todas las simulaciones utilizan un entorno limitado y los robots se sitúan inicialmente de manera aleatoria. Las pruebas se realizarán para diferentes tamaños de entorno y diferentes tamaños de enjambre.

5.1. Agregación

El comportamiento de agregación está considerado uno de los comportamientos fundamentales de los enjambres. Por un lado, se observa en muchos organismos desde bacterias a insectos sociales y mamíferos. La agregación ayuda a los organismos a sobrevivir en entornos hostiles, a evitar depredadores y a encontrar compañeros. Por ejemplo, algunas especies de escarabajos inducen la agregación debajo de las piedras para protegerse del sol y el calor, en otras especies los machos forman un grupo agregado formando una mancha de mayor tamaño para protegerse de depredadores y otras especies utilizan las feromonas de agregación sexuales para atraer a miembros de un determinado sexo con fines reproductivos [Bel03]. Por otro lado, la agregación puede actuar como precursor de otros comportamientos más complejos como *flocking*, formación de patrones o auto-ensamblado.

En la naturaleza algunos organismos desarrollan un comportamiento de agregación basándose en algún elemento de entorno, por ejemplo, algunas especies de gusanos se agregan alrededor de zonas con un nivel de humedad elevado, mientras que algunas especies de moscas utilizan la luz o la temperatura. Sin embargo, existe también la agregación auto-organizada que no se basa en ningún elemento del entorno, sino en un comportamiento de cooperación emergente entre los individuos que forman el enjambre, como por ejemplo, en los bancos de peces, los pingüinos o las cucarachas [SBS07].

En esta sección se verá un ejemplo de agregación auto-organizada. A continuación se analizará este comportamiento, donde los individuos son capaces de agregarse teniendo en cuenta únicamente la información percibida por sus sensores, sin tener conocimiento del tamaño del entorno o del enjambre.

5.1.1. Definición del comportamiento

En los artículos [SBS07] y [SS07] especifican un comportamiento de agregación básico para robótica de enjambre. En ellos se indica que la agregación auto-organizada, que no requiere ninguna pista en el entorno y con control

descentralizado, es una competencia esencial requerida de los sistemas robóticos de enjambre.

A continuación se definirá un comportamiento de agregación genérico, basado en los dos artículos anteriores, con cuatro estados básicos: evitación de obstáculos, aproximación, repulsión y espera. Estos estados se dividen en una arquitectura de subsunción de dos capas. La capa inferior contiene un comportamiento genérico de evitación de obstáculos, mientras que la capa superior contiene una máquina de estados finita con tres estados (aproximación, repulsión y espera). La figura 5.1 muestra un esquema de esta arquitectura.

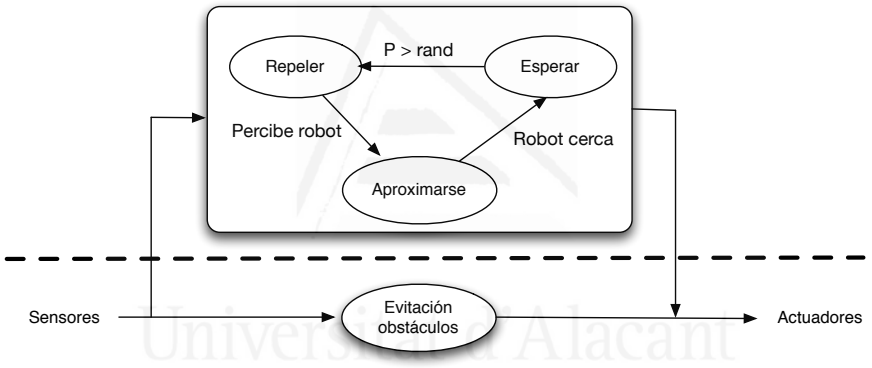


Figura 5.1: Arquitectura para el comportamiento de agregación básico. Define una arquitectura de subsunción de dos capas. La capa inferior define un comportamiento básico de evitación de obstáculos, la capa superior define una máquina de estados finita con tres estados.

El comportamiento del nivel inferior, evitación de obstáculos, entra en funcionamiento cuando se detecta un obstáculo a una distancia menor que un umbral. Es un comportamiento básico para prevenir las colisiones con paredes, objetos u otros robots.

En el nivel superior, se sitúa una máquina de estados finita compuesta de tres comportamientos. El comportamiento de *aproximación* se basa en los sensores, de manera que si se detecta otro robot se dirige hacia él. Cuando el robot se encuentra lo suficientemente cerca de otro robot pasa al comportamiento de *espera*. Este comportamiento frena el robot haciendo que el robot quede parado junto al robot que ha detectado. Cuando el robot se encuentra en estado *espera* el comportamiento de evitación de obstáculos debe ser desactivado para permitir que el robot esté parado cerca de otros

robots. En este estado *espera* el robot calculará un número aleatorio, si este número es menor que un cierto valor P, el robot entrará en el estado *repeler* que hace que el robot se separe del grupo en el que estaba para buscar otro grupo. Según este valor P el enjambre tendrá comportamientos diferentes.

El objetivo final del comportamiento de agregación básica auto-organizada es la formación de un único grupo que contenga todos los individuos del enjambre sin utilizar ningún elemento del entorno, ni ningún tipo de control centralizado.

La posición inicial de cada uno de los robots será aleatoria, y éstos no tendrán conocimiento del tamaño del entorno, ni del enjambre. Los robots comenzarán en el estado *repeler*. En este estado se desarrolla un comportamiento de movimiento aleatorio por el entorno si no es detectado ningún robot. En el momento en que un robot sea detectado, se pasará al estado de *aproximación*.

Este comportamiento básico, donde los robots no tienen ninguna información adicional a parte de la detectada por sus propios sensores, ha sido implementado en la arquitectura propuesta utilizando únicamente los agentes μ ya que no es necesario ningún mecanismo de coordinación llevado a cabo por los agentes superiores Ψ.

En el primero de los artículos [SBS07] se estudia el comportamiento del enjambre para valores constantes de P, analizando varias estrategias, $P \in (0,1)$, $P = 0$ y $P = 1$. En el siguiente artículo [SS07] estudian el comportamiento del enjambre para valores de P dependientes del tamaño del grupo en el que se encuentra el robot. La métrica que utilizan para comparar la ejecución o el desarrollo de los diferentes comportamientos de agregación es el tamaño del clúster de mayor tamaño para diferentes tamaños de enjambre y de entorno. Por este motivo, a continuación se analizará el funcionamiento del comportamiento de agregación para diferentes valores de P basándose en el clúster de mayor tamaño.

5.1.2. Experimentación

Valores de P constantes

Para entender el efecto de la probabilidad de transición P en el comportamiento de agregación se han estudiado tres estrategias que varían la cantidad de tiempo que los robots están en los diferentes estados del comportamiento. Para cada valor de P se han llevado a cabo cinco simulaciones para diferentes tamaños del enjambre (50, 100, 150) y para diferentes tamaños del entorno (100×100, 150×150, 200×200).

Primera estrategia $P = 0$**.** En este caso los robots permanecerán en el estado *esperar* y el comportamiento se reducirá al mostrado en la figura 5.2. Cuando los robots llegan al estado *esperar* no salen de este estado, es decir, los robots estarán en movimiento hasta que detecten a otro robot, en ese momento, quedarán parados en estado *esperar*.

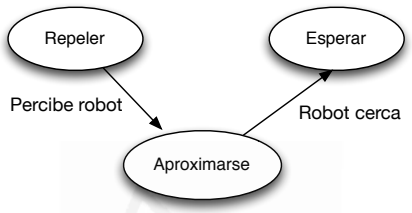


Figura 5.2: Representación de la máquina de estados cuando $P = 0$.

Esta estrategia formará aleatoriamente varios grupos pequeños de robots sin ser significativo ni el tamaño del enjambre ni del entorno. En el gráfico 5.3 se observa el comportamiento del enjambre. En a) se muestra el tamaño del grupo más grande. En este caso, cuando $P = 0$ se forman pequeños grupos de robots como se observa en la figura 5.4, siendo el tamaño máximo de grupo menor que 10 robots. b) muestra el número de robots que se encuentran en estado *esperar*, se observa como inicialmente ningún robot está en este estado y con el número de iteraciones el número de robots en este estado se incrementa progresivamente hasta alcanzar la totalidad de los robots, donde el enjambre se estabiliza y no se producen cambios. Inversamente al número de robots que se encuentran en estado *esperar*, en c) se muestra el número de robots que se encuentran en estado *aproximar*. Inicialmente todos los robots se encuentra en estado *repeler* pero en pocas iteraciones todos los robots pasan al estado *aproximar* y conforme los robots se van agrupando disminuye el número de robots en este estado hasta llegar a 0, indicando que todos los robots están agrupados en algún grupo y por tanto en estado *esperar*.

Segunda estrategia $P = 1$**.** En este caso los robots no llegarán nunca a formar un grupo, dado que los robots no se quedarán en el estado *esperar*. Los robots pasarán del estado *repeler* a *aproximar* y viceversa de manera continua. El comportamiento se reduciría al mostrado en la figura 5.5.

En esta estrategia los robots no paran de moverse y nunca se forman grupos como se observa en la figura 5.6 teniendo el mismo comportamiento para diferentes tamaños del enjambre y del entorno.

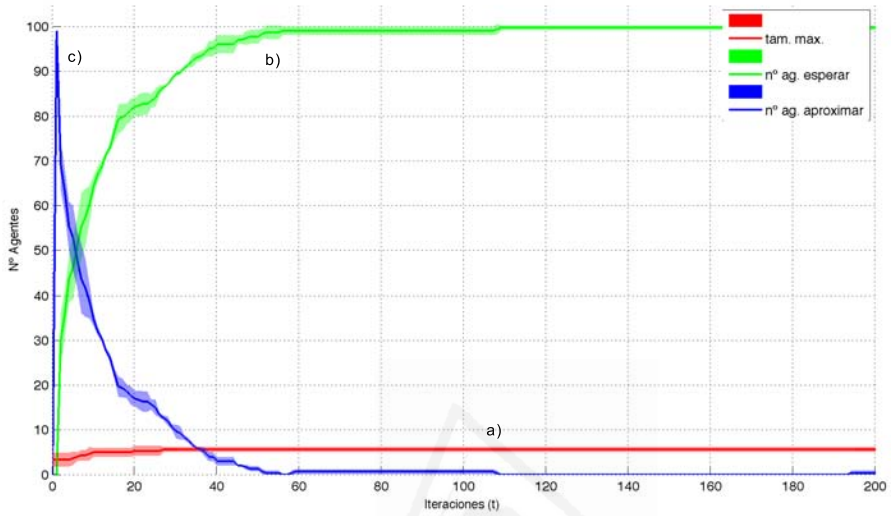


Figura 5.3: Resultado de los experimentos cuando $P = 0$. a) muestra el tamaño del grupo más grande, el tamaño máximo. b) número de robots en estado *esperar*. c) número de robots en estado *aproximar*. Ejemplo para un tamaño de enjambre de 100 robots donde se muestra la media y la desviación típica de 5 ejecuciones para cada uno de los valores.

Como se observa en el gráfico 5.7 los robots se dividirán entre los estados *repeler* y *aproximar* sin llegar a formarse ningún grupo. En a) se observa cómo el tamaño máximo de grupo no supera los tres robots.

Tercera estrategia $P \in (0, 1)$**.** El valor de P determinará el comportamiento del enjambre. Cuanto mayor sea el valor de P los robots estarán más tiempo en estado *repeler* y se reducirá el tamaño de las agrupaciones, pero por otro lado permite que más robots estén buscando grupo y, en este caso, será mayor la posibilidad de formar grupos más grandes. Por lo tanto, el valor de P determinará el comportamiento del enjambre. El gráfico 5.8 muestra el número de robots en estado *repeler* para diferentes valores de P. Se observa que para valores de P a partir de 0'1, el número de robots en este estado es muy alto, lo que conlleva un comportamiento similar a $P = 1$ donde no llegan a formarse grupos. Por otro lado, cuando P tiene un valor pequeño (sobre 0'0001) el número de robots en este estado es cercano a 0 y por tanto el comportamiento es similar a $P = 0$ formándose pequeños grupos de robots. Para valores de P entre estos dos (0'001) se mantiene un pequeño número de robots en estado *repeler* que hará que los robots dejen

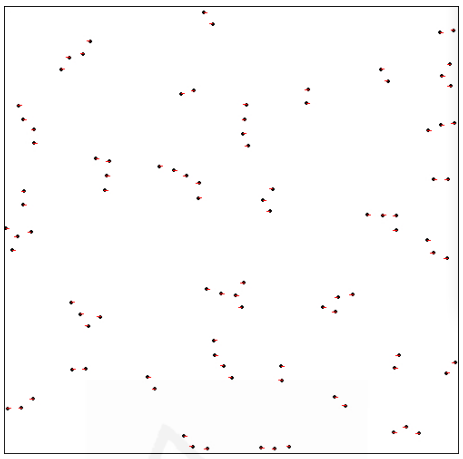


Figura 5.4: Resultado final de la ejecución para $P = 0$ para un enjambre de tamaño 100 en un entorno de tamaño 150×150. Se observa la formación de pequeños grupos de robots.

un grupo para buscar otro. Aunque en este caso los grupos de robots que se forman son también pequeños (menos de 10 robots) independientemente del tamaño del enjambre o del entorno, ya que los robots cambian de un grupo a otro sin llegar a formarse grupos grandes.

Con estas tres estrategias ha sido posible comprobar el efecto de la probabilidad de transición en el comportamiento del enjambre y el correcto intercambio entre los estados de los agentes, pero ninguna de estas estrategias es adecuada para llevar a cabo un comportamiento de agregación total del enjambre. Con el fin de mejorar estos resultados, en [SS07] retroalimen-

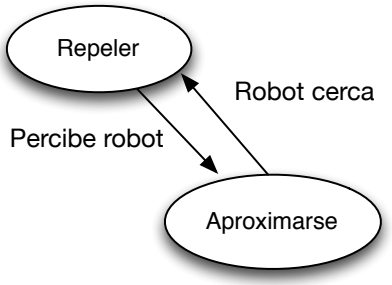


Figura 5.5: Representación de la máquina de estados cuando $P = 1$.

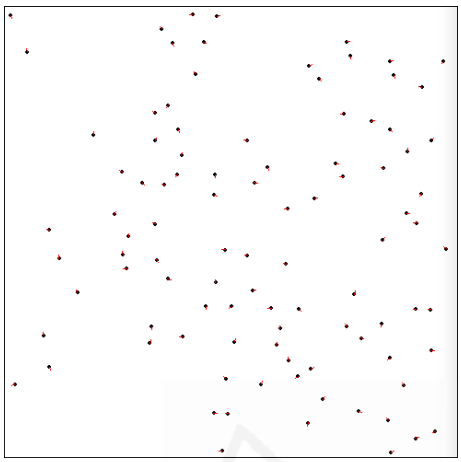


Figura 5.6: Resultado de la ejecución para $P = 1$ para un enjambre de tamaño 100 en un entorno de tamaño 150×150. Se observa que los robots no forman ningún grupo, debido a que están en continuo movimiento.

tan el sistema utilizando el tamaño del grupo en el que se encuentra el robot como veremos a continuación.

P dependiente del tamaño del grupo

En [SS07] definen la probabilidad de transición como $P = G/i^2$, siendo G una constante e i el tamaño del grupo en el que se ha agregado el agente. En [SS07] no definen el valor que debería tomar G pero según los estudios realizados, para poder obtener unos resultados óptimos, el valor de esta constante debería ser diferente para cada tamaño del enjambre. Para valores de G pequeños (en relación con el tamaño de los grupos que se pueden formar) se producirán muy pocos cambios entre los estados de los agentes y, por tanto, el enjambre tendrá un comportamiento similar a $P = 0$, formándose pequeños grupos de robots, como se puede observar para los valores de $G = 0'0001$ y $G = 0'001$ en el gráfico 5.9 donde el tamaño máximo de grupo no supera los 10 agentes. Por el contrario, para valores de G elevados el enjambre tendría un comportamiento similar a $P = 1$ y por tanto no se formarían grupos. En el ejemplo mostrado en el gráfico 5.9 se observa como $G = 0'1$ es el que obtiene mejores resultados para un enjambre de 100 robots formando grupos cercanos a los 40 robots.

El valor de G y el tamaño del enjambre tienen una relación lineal, con un coeficiente de correlación de Pearson cercano a 1. En la figura 5.10 se

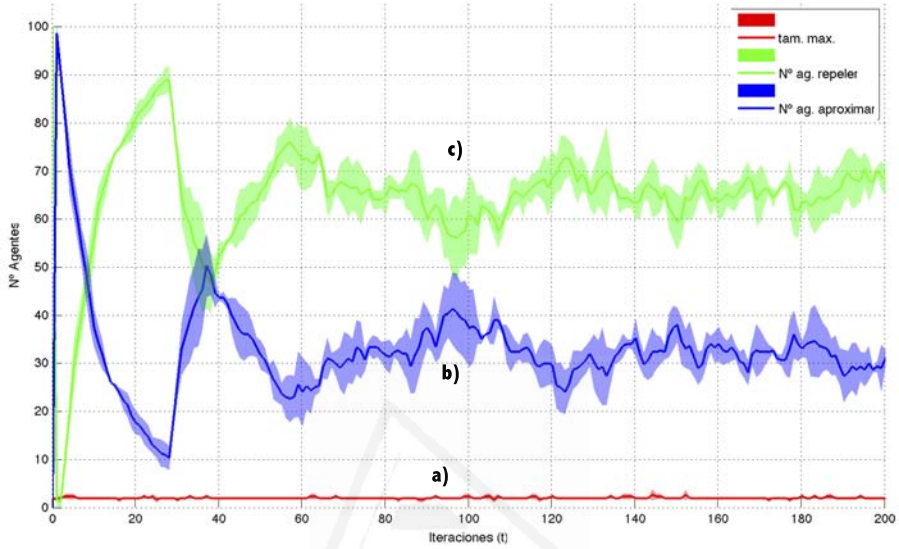


Figura 5.7: Resultado de los experimentos cuando $P = 1$. a) muestra el tamaño del grupo más grande, el tamaño máximo. b) número de robots en estado *aproximar*. c) número de robots en estado *repeler*. Ejemplo para un tamaño de enjambre de 100 robots donde se muestra la media y la desviación típica de 5 ejecuciones para cada uno de los valores.

observa el valor óptimo de G para diferentes tamaños de enjambre.

Los resultados obtenidos con esta estrategia son mejores que para valores constantes de P, pero aun con el uso de un valor de G adecuado para el tamaño del enjambre, los resultados no son óptimos (no se forma una única agrupación) debido a que los robots siguen teniendo una probabilidad de abandonar el grupo en el que se encuentran, por este motivo, en el gráfico 5.9 se observan las oscilaciones en cuanto al número máximo de agentes agrupados en el mejor de los casos.

P dependiente del tamaño del grupo más grande

Con los valores de la probabilidad de transición vistos anteriormente es suficiente el uso de agentes μ ya que no es necesario ningún conocimiento de nivel superior. En nuestra arquitectura es posible hacer uso de los agentes Ψ para proporcionar a los agentes inferiores μ información necesaria para el desarrollo de su tarea. En este caso, con el fin de conseguir un funcionamiento correcto del comportamiento de agregación y formar un único grupo de

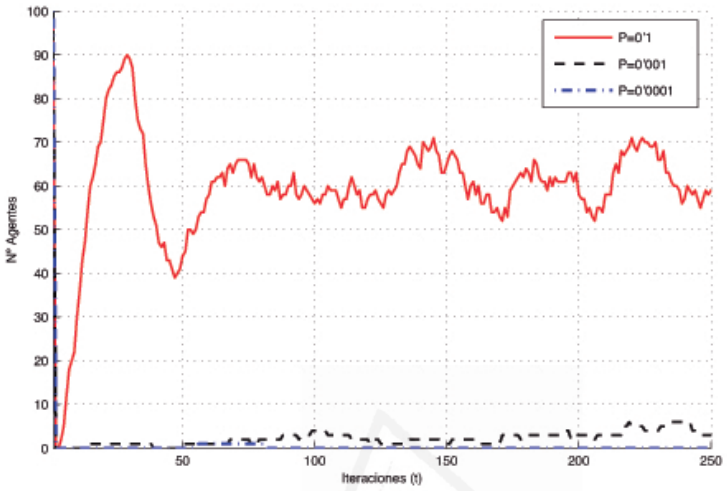


Figura 5.8: Resultado de los experimentos cuando $P \in (0, 1)$. Se muestra el número de robots en estado *repeler* para diferentes valores de P en un experimento modelo para cada valor.

robots que contenga todo el enjambre, se propone una modificación de la probabilidad de transición de manera que ésta dependa también del tamaño del grupo más grande. De esta manera, cuando el robot pertenezca al grupo más grande, tendrá menos probabilidades de abandonarlo. La probabilidad de transición sería $P = (G \cdot i\_max)/i^2$, siendo i_max el tamaño del grupo más grande. Con esta probabilidad los agentes que pertenecen al grupo más grande tendrán una probabilidad baja de abandonarlo, por el contrario, los agentes que pertenecen a otros grupos tendrán una probabilidad más alta de abandonar el grupo en busca de otro. Por este motivo, los agentes se irán agregando en el grupo más grande, lo que conlleva que se forme un único grupo con la totalidad de los agentes y se alcance el objetivo del comportamiento de agregación.

El gráfico 5.11 muestra ejemplos de ejecución de un enjambre de tamaño 100 para diferentes tamaños de entorno. Este gráfico muestra la evolución de la cantidad total de agregados en el grupo de mayor tamaño. Se observa cómo en todos los casos se consigue que todos los agentes se agreguen en un único grupo. Se observa cómo para los tres tamaños de entorno diferentes se alcanzan los 100 agentes agregados.

En este gráfico se puede observar también cómo el tiempo de agregación depende del tamaño del entorno. Para este ejemplo con un enjambre de 100

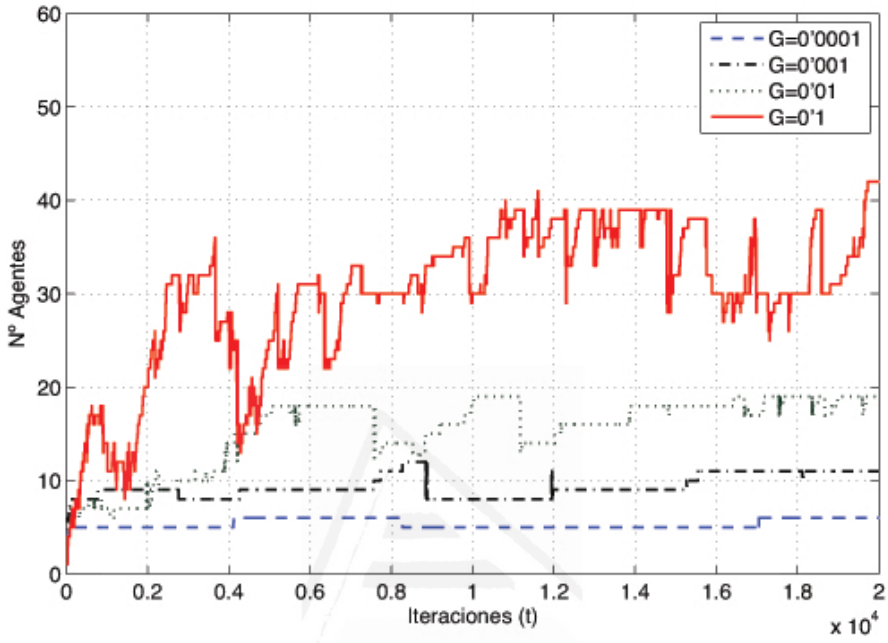


Figura 5.9: Resultado de los experimentos cuando $P = G/i^2$. Se muestra el tamaño máximo de agrupación para diferentes valores de G. Ejemplo de un experimento modelo para cada valor de G con un enjambre de 100 robots con un entorno de 150×150.

robots, se produce la agregación completa sobre las 17.000 iteraciones para un entorno de tamaño 100×100, en un entorno de 150×150 la agregación se produce sobre las 25.000 iteraciones y en un entorno de 200×200 el enjambre se agrega sobre las 35.000 iteraciones. Por lo tanto, el tiempo de agregación dependerá del tamaño del entorno, pero el enjambre formará un único grupo agregado con la totalidad de los robots.

La imagen 5.12 muestra un ejemplo de la evolución de este comportamiento de agregación para un enjambre de 100 robots.

5.1.3. Discusión

Para probar el funcionamiento del comportamiento de agregación se han realizado una serie de pruebas. Estas pruebas se han dividido en dos conjuntos, en el primer grupo de pruebas se ha utilizado una probabilidad de transición constante, y en el segundo grupo una probabilidad de transición

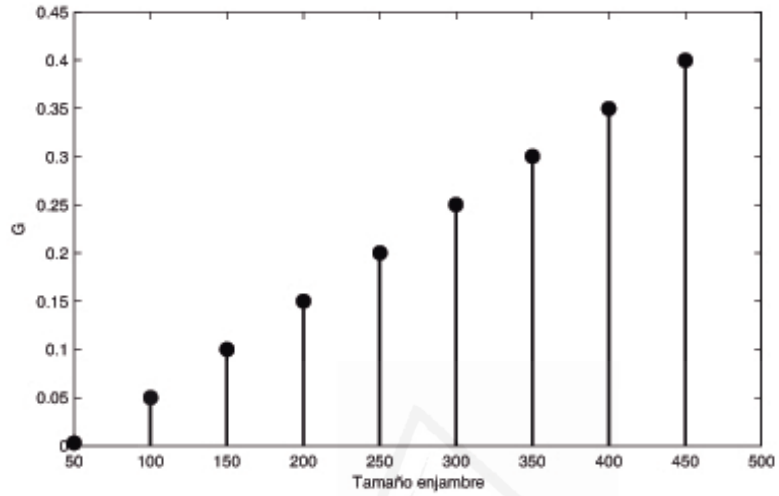


Figura 5.10: Relación entre el valor de la constante G y el tamaño del enjambre.

dependiente del tamaño del grupo en el que se encuentra el robot.

El primer conjunto de pruebas he permitido comprobar el efecto de la probabilidad de transición en el comportamiento de los agentes, observándose el correcto intercambio entre los estados de los agentes. Se han obtenido los mismos resultados que los estudios realizados en [SBS07]. Para $P = 0$ los robots permanecen en estado *esperar* y se forman pequeños grupos de robots. Para $P = 1$ los robots nunca permanecen en estado *esperar* y por tanto nunca se quedan parados formando un grupo, están en constante movimiento. Para $P \in (0, 1)$ se obtienen mejores resultados que con los dos valores anteriores, pero sólo se obtienen pequeños grupos de robots. Estas estrategias no son adecuadas para obtener un grupo único con todos los robots, pero han servido para comprobar el correcto funcionamiento de la máquina de estados definida.

En el segundo conjunto de pruebas se ha utilizado una probabilidad de transición dependiente del tamaño del grupo en el que se encuentra el robot. En [SS07] utilizan $P = G/i^2$. Esta probabilidad ofrece un mejor rendimiento del comportamiento, llegando a formarse grupos grandes como se observa en el gráfico 5.9, pero no llega a cumplirse el objetivo de formar un único grupo agregado.

En base a estas pruebas anteriores, se ha propuesto una modificación de la probabilidad de transición, teniendo en cuenta el tamaño del grupo más

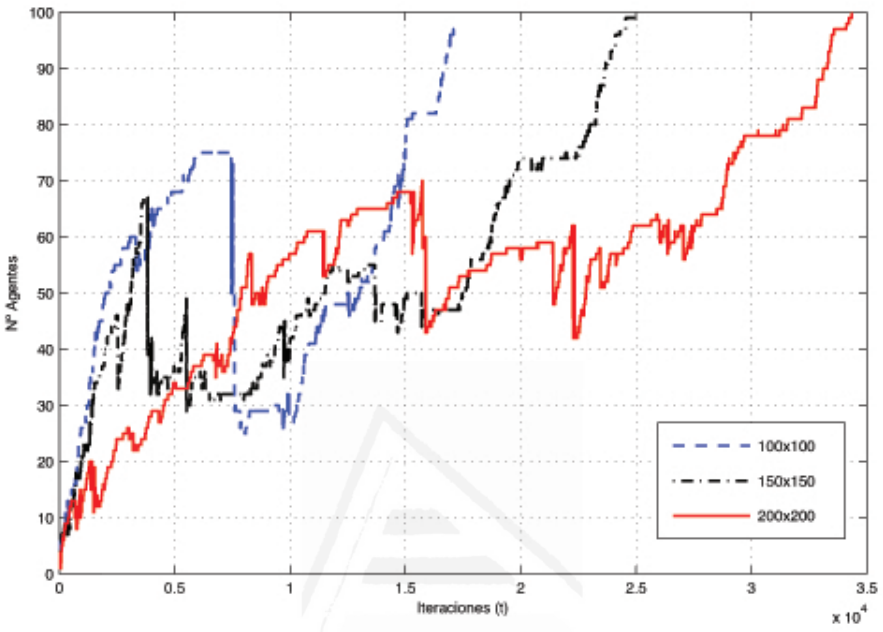


Figura 5.11: Tiempo de agregación para un tamaño de enjambre de 100 robots y diferentes tamaños de entorno. Se muestra el resultado de un experimento modelo para cada tamaño de entorno.

grande. De esta manera, si el robot pertenece al grupo más grande tiene menos posibilidades de entrar en el comportamiento de *repeler*. Se utilizaría $P = (G{*}i\_max)/i^2$. En el gráfico 5.11 se observa cómo el tiempo que necesita el enjambre para forman la agregación dependerá del tamaño del entorno, pero se conseguirá una única agregación con todos los agentes del sistema.

Con esta última probabilidad de transición se consigue el objetivo del comportamiento de agregación y el enjambre forma un único grupo como se observa en la imagen 5.12.

5.2. Movimiento coordinado (*flocking*)

Como se ha visto en el capítulo 2 el movimiento coordinado hace referencia a la coordinación del movimiento de un grupo de robots, de manera que sean capaces de realizar un movimiento suave, evitando obstáculos del entorno. Hace referencia a la manera en que las poblaciones de animales como pájaros, peces o insectos se mueven conjuntamente. Este comportamiento

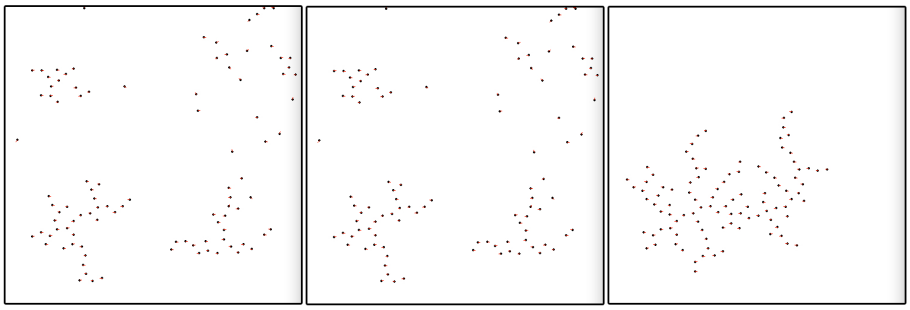


Figura 5.12: Evolución del comportamiento del enjambre. Ejemplo para un enjambre de 100 robots y un entorno 150×150.

les aporta, principalmente, dos beneficios al grupo: por un lado asegura la supervivencia del grupo evitando ser depredados; y por otro, les ayuda a encontrar comida [LJYL12]. Cada posición dentro del grupo conlleva una serie de ventajas o inconvenientes, por ejemplo, los individuos situados en los extremos del grupo tienen más posibilidad de encontrar comida, pero también están más expuestos a los ataques de los depredadores, por otro lado, los individuos situados en la cola del grupo necesitan menos energía en su movimiento por el efecto hidrodinámico del grupo, pero los que se sitúan en la cabeza son los que, según ciertos estudios, dirigen el grupo [AC87] [KTI+11]. Estos comportamientos son intrínsecos a la posición en la que está situado el individuo (que varía en el tiempo) sin afectar al comportamiento global, que debe surgir como una consecuencia de las interacciones locales entre los miembros vecinos.

Este comportamiento es una evolución del comportamiento de agregación visto en el punto 5.1.1, donde cuando un robot se acerca a otro robot éste entra en el estado *esperar*, formándose un grupo de robots parados. En este comportamiento se pretende formar un grupo de robots pero capaces de moverse conjuntamente como las bandadas de pájaros o los bancos de peces.

Muchos de los trabajos relacionados con el movimiento coordinado de un grupo de robots [MSC11], [NGMMH08], [AAC10], [TCGS08a], [SFT+11] hacen referencia al trabajo desarrollado por Reynolds [Rey87], que muestra que el movimiento coordinado se puede obtener utilizando tres comportamientos simples: separación, donde los individuos tratan de mantener una distancia mínima con sus vecinos; cohesión, donde los individuos tratan de acercarse a sus vecinos; y alineación, donde los individuos tratan de mantener la velocidad adecuándola a la de sus vecinos.

En [MSC11] utilizan las lecturas activas y pasivas de los sensores de infra-

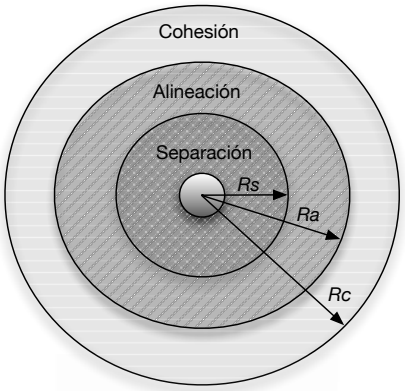


Figura 5.13: Diagrama de las diferentes zonas de actuación del robot según el esquema de Reynolds.

rrojos. Discretizan los campos de los sensores y utilizan diferentes umbrales de distancia para cohesión y separación, surgiendo de manera emergente la alineación. Utilizan una arquitectura de subsunción simple con cuatro posibles decisiones (evitar obstáculos, separación, cohesión y alineación). En [NGMMH08] utilizan los sensores de infrarrojos e información de un sistema de orientación situado en la base de cada robot que permite conocer su posición y orientación global. Cada robot tendrá un comportamiento de cohesión o de separación dependiendo de la distancia medida. En [AAC10] utilizan una arquitectura de control basada en comportamientos, donde definen una serie de comportamientos muy simples con unas prioridades establecidas. En [SFT+11] todos los robots tienen los comportamientos de separación y cohesión pero sólo unos pocos tienen el comportamiento de alineación.

En este punto se define una estrategia de agregación con movimiento basada en [MSC11] y [YT07] que se inspira en la formación de los bancos de peces, donde los agentes se van agregando al grupo que está en movimiento.

Siguiendo el esquema de Reynolds se definen tres zonas de actuación, como se muestra en la figura 5.13. En esta figura el círculo central representa al robot y los círculos que lo circunscriben indican las tres áreas de actuación. De manera que, cuando un robot detecta otro robot dentro del área de separación, el robot tenderá a separarse; cuando detecta un robot dentro del área de alineación, ejecutará un movimiento paralelo, adecuando su velocidad a la de sus vecinos; y cuando detecta un robot en el área de cohesión, el robot se moverá hacia éste.

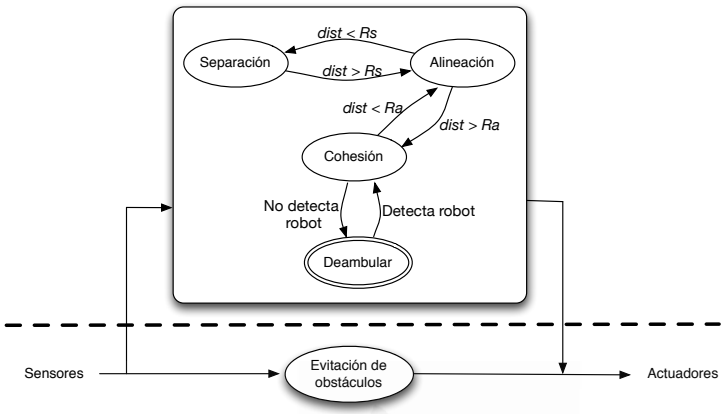


Figura 5.14: Esquema de la arquitectura definida para desarrollar un comportamiento de *flocking*. Arquitectura de subsunción de dos capas: la capa inferior define una conducta general de evitación de obstáculos; la capa superior contiene una máquina de estados finita con cuatro estados correspondientes a las diferentes zonas de actuación.

5.2.1. Definición del comportamiento

Para este comportamiento de movimiento coordinado se define una arquitectura de subsunción de dos capas. En la capa inferior se sitúa un comportamiento de evitación de obstáculos que entra en funcionamiento cuando se detecta un obstáculo a una distancia menor que un umbral. En la capa superior se sitúa una máquina de estados finita con cuatro posibles estados (deambular, cohesión, separación y alineación). En la figura 5.14 se muestra un esquema de esta arquitectura.

Los robots comienzan en estado *deambular*. En este estado los robots desarrollan un comportamiento de movimiento aleatorio por el entorno. Cuando un robot detecta otro robot a una distancia menor que R_c, ejecuta el estado de *cohesión* para acercarse. Al acercarse, llegará un momento que detecte al robot a una distancia menor que R_a, indicando que se encuentra en el área de alineación y, por tanto, ejecutará el estado de *alineación* para mantener una distancia continua con el otro robot. Si detecta un robot, a una distancia menor que R_s, estará dentro del área de separación y, por tanto, entrará en el estado de *separación* y tenderá a separarse del robot para volver al estado de *alineación*.

El estado de *alineación* será el estado deseado del enjambre. Para un comportamiento óptimo del grupo, los robots deben mantenerse en este esta-

do, variando entre los estados *alineación-separación-alineación* si se acercan demasiado y entre los estados *alineación-cohesión-alineación* si se separan.

Con este comportamiento cada robot ajusta su movimiento de manera autónoma teniendo en cuenta su vecino más cercano y basándose únicamente en la información percibida por sus sensores, lo que hace que esta estrategia sea flexible y robusta. Para este comportamiento únicamente son necesarios los agentes μ de la arquitectura, ya que el comportamiento global del grupo emerge automáticamente y los robots no necesitan ninguna información adicional, únicamente la lectura de sus sensores.

5.2.2. Experimentación

Las pruebas realizadas para comprobar el funcionamiento del comportamiento se dividen en tres grupos. En las primeras pruebas se han utilizado entornos vacíos para probar que la agregación y el movimiento del grupo es correcto. El segundo grupo de pruebas se han realizado para comprobar que el comportamiento es robusto ante entornos con obstáculos y robots que quedan parados por anomalías en su sistema. El último grupo de pruebas estudia los efectos que produce en el comportamiento cambios de los valores R_s, R_a y R_c.

Todas las pruebas se han llevado a cabo para enjambres de diferentes tamaños (50, 100 y 150 robots) y para entornos de diferentes tamaños (100×100, 150×150 y 200×200). Los valores iniciales de R_s, R_a y R_c serán establecidos a 5, 10 y 15, respectivamente. Se han realizado 5 simulaciones para cada conjunto de valores.

Agregación y movimiento

Inicialmente los robots se sitúan de manera aleatoria en el entorno, como se muestra en la imagen 5.15 a). Conforme ejecutan el comportamiento y los robots se van encontrando, se irán formando grupos de robots en movimiento hasta formar un único grupo, como se puede observar en la secuencia de la imagen 5.15.

Los robots comienzan en el estado *deambular* moviéndose de manera aleatoria por el entorno. Cuando detectan otro robot ejecutan el estado *cohesión* para acercarse a él. Una vez están a una distancia menor que R_a del robot, ejecutan el estado *alineación* para adecuar su movimiento al de su vecino y seguir un movimiento coordinado.

Este intercambio de estados se puede observar en la figura 5.16 donde se muestra el número de robots que se encuentran en cada estado. Se observa

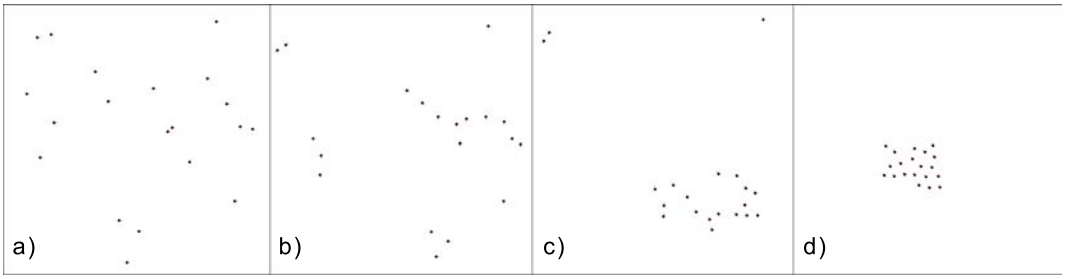


Figura 5.15: Secuencia del comportamiento de *flocking* desde la inicialización aleatoria hasta la formación del grupo. Ejemplo para un enjambre de 20 robots y un entorno de tamaño 100×100.

cómo inicialmente todos los robots se encuentran en estado *deambular*. El número de robots en este estado decrece con el tiempo, y por el contrario, el número de robots en estado *alineación* aumenta conforme los robots se van encontrando con otros robots y van formando grupos. En la figura se puede ver cómo el paso entre los estados *deambular-cohesión-alineación* es muy rápido, pareciendo que los robots pasan del estado *deambular* a *cohesión* directamente (sin ser así). Una vez el enjambre está estabilizado y los robots forman parte de un grupo, variarán entre los estados de *alineación-separación-alineación* para mantener una distancia adecuada mientras están en movimiento.

En esta figura (5.16) se puede observar cómo a partir de las 1000 iteraciones prácticamente la totalidad del enjambre se sitúa en el estado de *alineación*, lo que significa que todos los robots forman parte de algún grupo, y a partir de ese momento los robots se sitúan entre los estados de *alineación* y *separación*.

Igual que ocurre con el comportamiento de agregación básico, el tiempo de agregación completa (en un único grupo) dependerá del tamaño del entorno para enjambres con igual número de robots. El tiempo necesario para alcanzar una agregación completa será mayor cuanto mayor sea el entorno, como se muestra en la figura 5.17. En este gráfico se observa cómo para el menor tamaño de entorno (100×100) a partir de las 2600 iteraciones se alcanza la agregación de prácticamente todo el grupo, mientras que para el resto de tamaños de entorno es necesario un tiempo mayor. A pesar de esto se observa cómo el número de agentes del grupo de mayor tamaño se va incrementando gradualmente.

Con estos estados el enjambre alcanza la agregación completa y, además, es capaz de realizar un movimiento coordinado de todo el grupo, como se

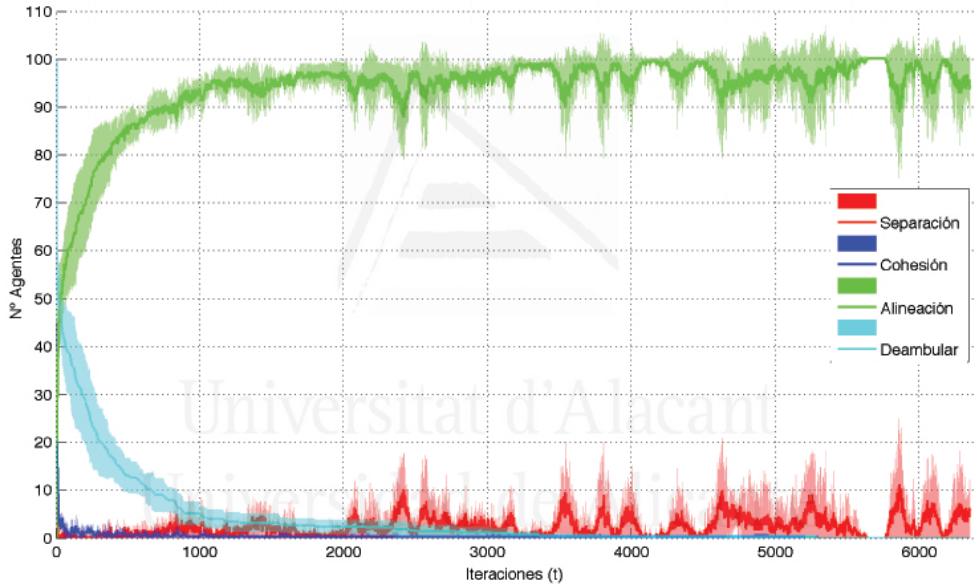


Figura 5.16: Representación del número de robots en cada estado para el comportamiento de *flocking*. Ejemplo para un enjambre de 100 robots y un entorno de tamaño 150 × 150 donde se muestra la media y la desviación típica de 5 ejecuciones.

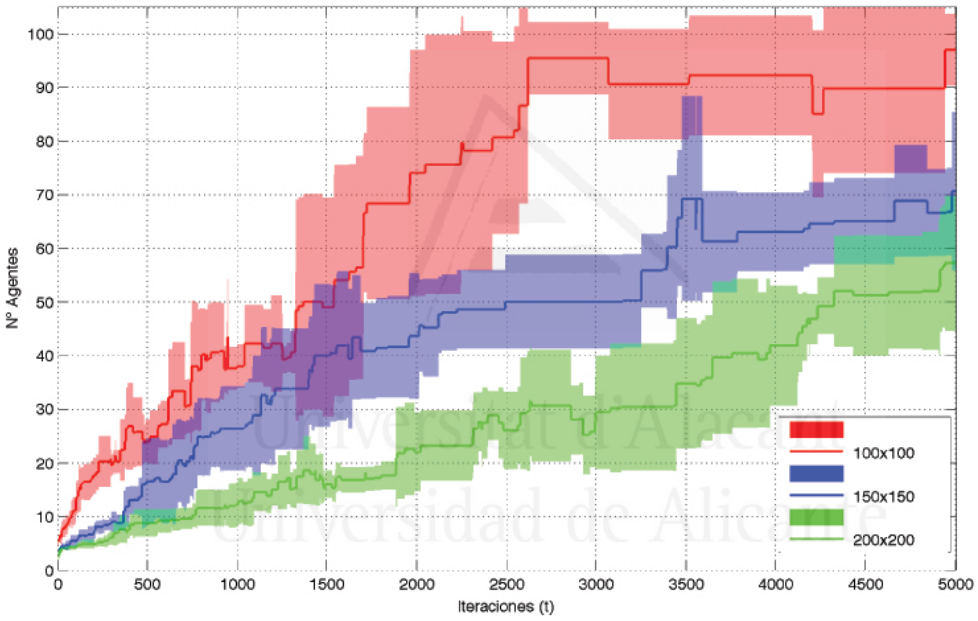


Figura 5.17: Representación del tamaño máximo de grupo en el comportamiento de *flocking* para diferentes tamaños de entorno. Ejemplo para un enjambre de 100 robots y entornos de diferentes dimensiones. Se muestra la media y la desviación típica de 5 ejecuciones para cada conjunto de valores.

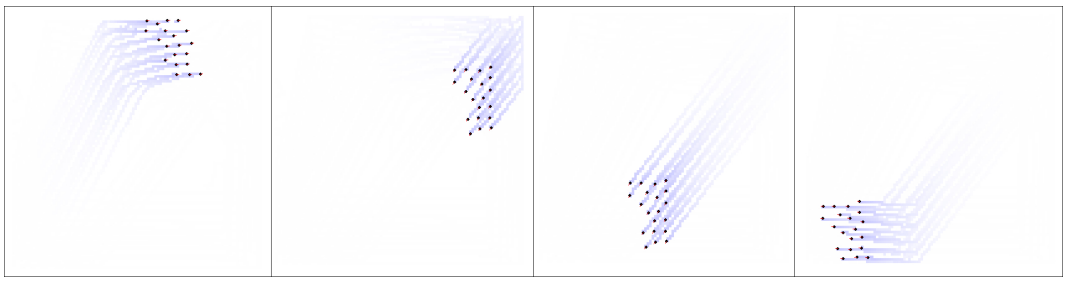


Figura 5.18: Secuencia del comportamiento de *flocking* donde se observa el movimiento que realizan los robots a partir de su rastro. Ejemplo para un enjambre de 20 robots y un entorno de tamaño 100×100.

puede observar en la figura 5.18. Esta figura muestra el comportamiento del enjambre en movimiento a partir de una secuencia de imágenes donde se muestra el rastro que deja cada robot. Se observa cómo los robots cuando llegan a los límites del mundo realizan un giro coordinado.

Flocking con obstáculos

Una vez comprobado que el enjambre se comporta de manera correcta para entornos vacíos, a continuación estudiaremos el comportamiento del enjambre ante entornos con obstáculos. Cuando en el entorno se encuentran obstáculos, robots parados o cualquier otro elemento que impida el paso de los agentes, el enjambre debe ser capaz de evitar el obstáculo manteniendo el comportamiento de movimiento coordinado de los agentes. Como en el apartado anterior, estudiaremos el número de agentes que se encuentran en cada estado y el tamaño máximo de grupo.

La figura 5.19 muestra el número de robots que se encuentran en cada estado. Como en el apartado anterior, se observa cómo en las primeras iteraciones aumenta el número de robots que se encuentran en estado *alineación* lo que nos indica que los robots van formando grupos hasta alcanzar prácticamente la totalidad del enjambre. Una vez el enjambre está estabilizado, se observa cómo los robots varían entre los estados *alineación-separación-alineación*. Siendo el mismo comportamiento que el observado en entornos sin obstáculos.

Con el comportamiento definido el enjambre es capaz de evitar obstáculos de manera correcta y seguir con el comportamiento de movimiento coordinado, como se observa en la secuencia de imágenes de la figura 5.20.

Cuando el enjambre está formado por un elevado número de robots y el

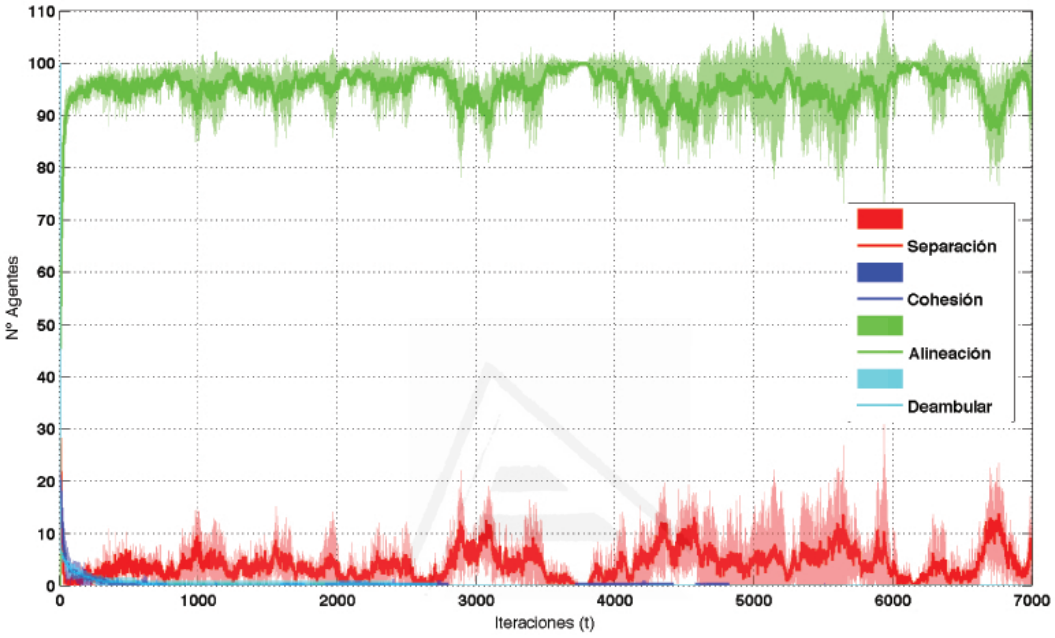


Figura 5.19: Representación del número de robots en cada estado para el comportamiento de *flocking* con obstáculos en el entorno. Ejemplo para un enjambre de 100 robots y un entorno de tamaño 150×150 donde se muestra la media y la desviación típica de 5 ejecuciones.

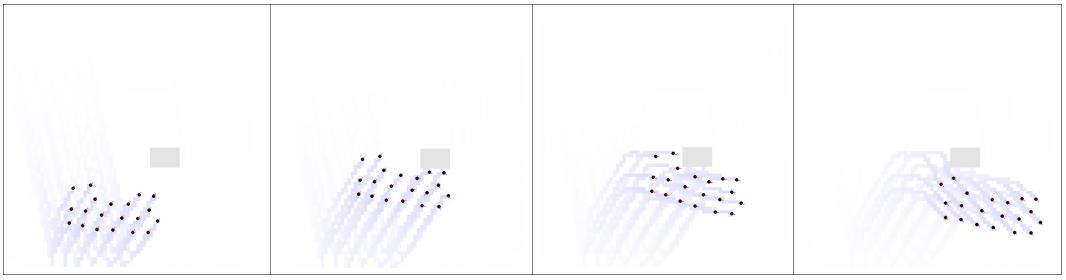


Figura 5.20: Secuencia del comportamiento de *flocking* donde se observa el movimiento que realizan los robots con un obstáculo en el entorno. Ejemplo para un enjambre de 20 robots y un entorno de tamaño 100×100.

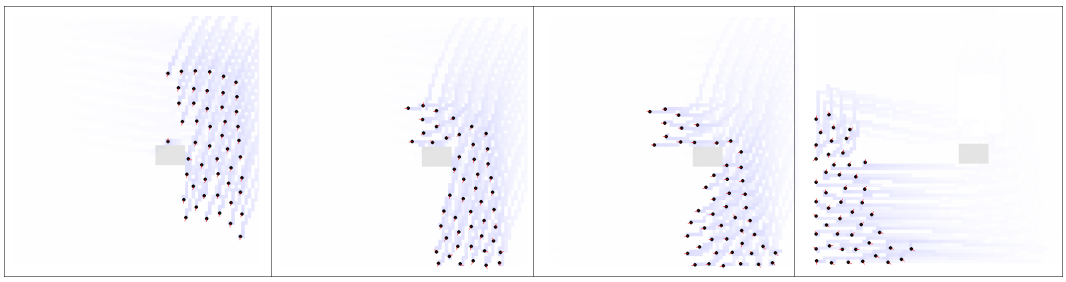


Figura 5.21: Secuencia del comportamiento de *flocking* donde se observa el movimiento que realizan los robots con un obstáculo en el entorno. Ejemplo para un enjambre de 50 robots y un entorno de tamaño 100×100.

grupo de agregados es muy grande, se puede dar el caso que al encontrar un obstáculo el enjambre se separe en varios grupos, que posteriormente volverán a unirse y continuarán con el comportamiento de movimiento coordinado, como se puede observar en la secuencia de imágenes de la figura 5.21.

La figura 5.22 muestra el tamaño máximo de grupo para una única simulación de 100 robots y tamaño de entorno 150×150. Se puede observar cómo en la primera parte de la ejecución el enjambre está en proceso de agregación hasta llegar a formar un grupo con la totalidad del enjambre (100 agentes). Una vez se consigue la agregación completa se observa cómo se reduce el número de agentes agregados en dos ocasiones (iteraciones 4500 y 8000). Esta reducción se produce por la separación del grupo ante la evitación de un obstáculo. Aunque se produzca esta separación se observa cómo el enjambre vuelve a unirse en un único grupo cuando el obstáculo ha sido evitado.

Cómo afectan los valores de R_s, R_a y R_c

Como se ha comentado anteriormente, este comportamiento se basa en tres áreas de actuación. Cada una de estas áreas viene delimitada por una distancia establecida por los valores R_s, R_a y R_c, para las áreas de *separación*, *alineación* y *cohesión*, respectivamente. En las pruebas realizadas anteriormente estos valores han sido establecidos a: $R_s = 5$, $R_a = 10$ y $R_c = 15$. A continuación se estudiará el comportamiento del enjambre ante variaciones en estos valores.

R_s establece la cohesión del enjambre, es decir, la distancia entre los robots. Para valores de R_s pequeños la distancia entre los robots será pequeña, los robots estarán más juntos. Por el contrario, para valores de R_s mayores,

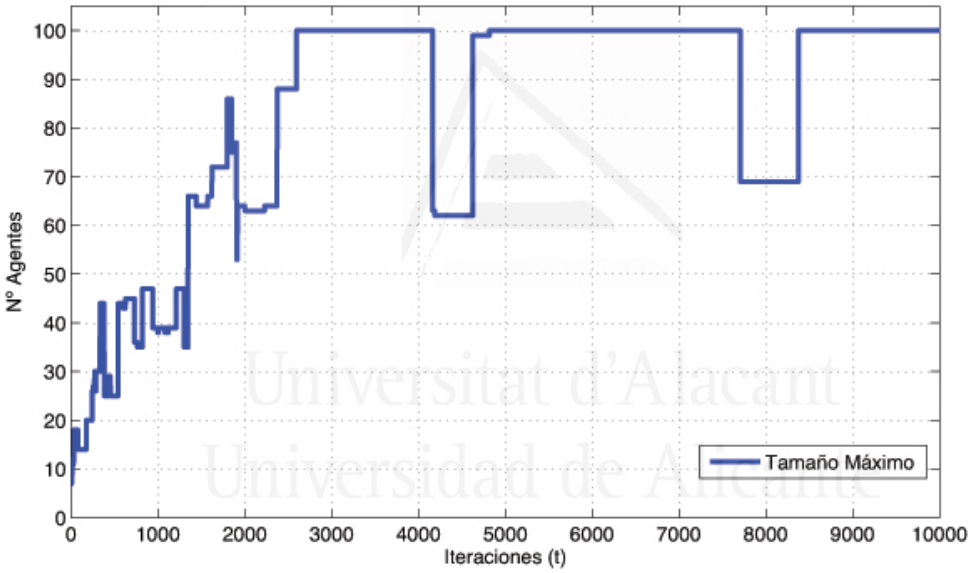


Figura 5.22: Representación del tamaño máximo de grupo para el comportamiento de *flocking* con obstáculos en el entorno. Ejemplo de una simulación para un enjambre de 100 robots y un entorno de tamaño 150×150.

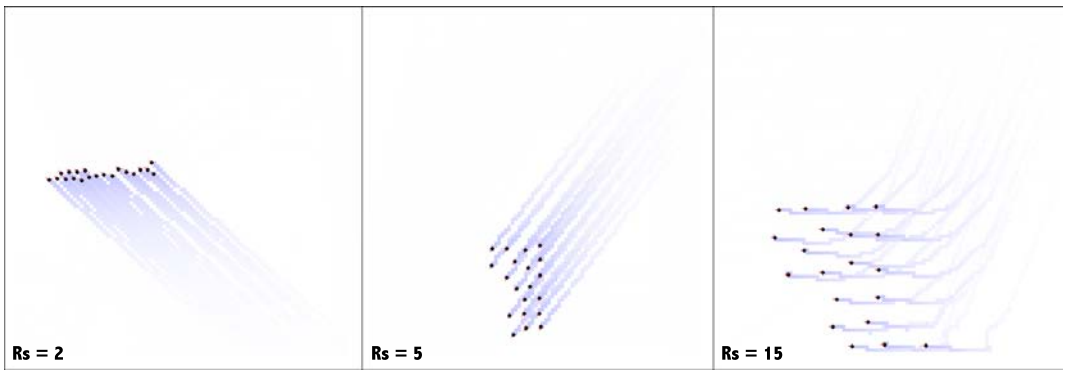


Figura 5.23: Capturas del enjambre para el comportamiento de *flocking* con diferentes valores de R_s. Ejemplo de una simulación para un enjambre de 20 robots y un entorno de tamaño 100×100.

los robots estarán más separados. Como se puede observar en la figura 5.23.

Cambios en R_s no suponen un cambio significativo en cuanto al comportamiento de movimiento coordinado, sino únicamente, en cuanto a la separación entre los robots. La figura 5.24 muestra el número de robots que se encuentran en los estados *alineación* (parte superior del gráfico) y *separación* (parte inferior) para diferentes valores de R_s. En color rojo se muestran los valores para $R_s = 2$, en color verde para $R_s = 5$ y en color azul para $R_s = 8$. Las gráficas obtenidas para los tres valores son muy similares a las obtenidas anteriormente. Únicamente para $R_s = 2$ se observa que el número de agentes en estado *separación* es más elevado. Esto es debido a que el valor utilizado es muy pequeño lo que hace que los robots estén muy cerca unos de otros y se produzcan más colisiones. Aún así, el comportamiento de movimiento coordinado se lleva a cabo adecuadamente.

R_a establece el tamaño del área de *alineación*. Cambios en este valor no suponen cambios significativos en el comportamiento del enjambre excepto si este valor está muy cercano a R_s. Cuando R_s y R_a tienen valores similares no se establece prácticamente un área de alineación, lo que conlleva que los robots no coordinen su movimiento con los de sus vecinos y, por lo tanto, se produzcan separaciones en el grupo. La figura 5.25 muestra la cantidad de robots en el grupo de mayor tamaño para un enjambre de 100 robots. En esta imagen se observa cómo el tamaño máximo de grupo se mantiene alrededor de los 50 agentes, es decir, la mitad del enjambre. No se llega a formar un único grupo debido a las separaciones que se producen al no tener área de *alineación*.

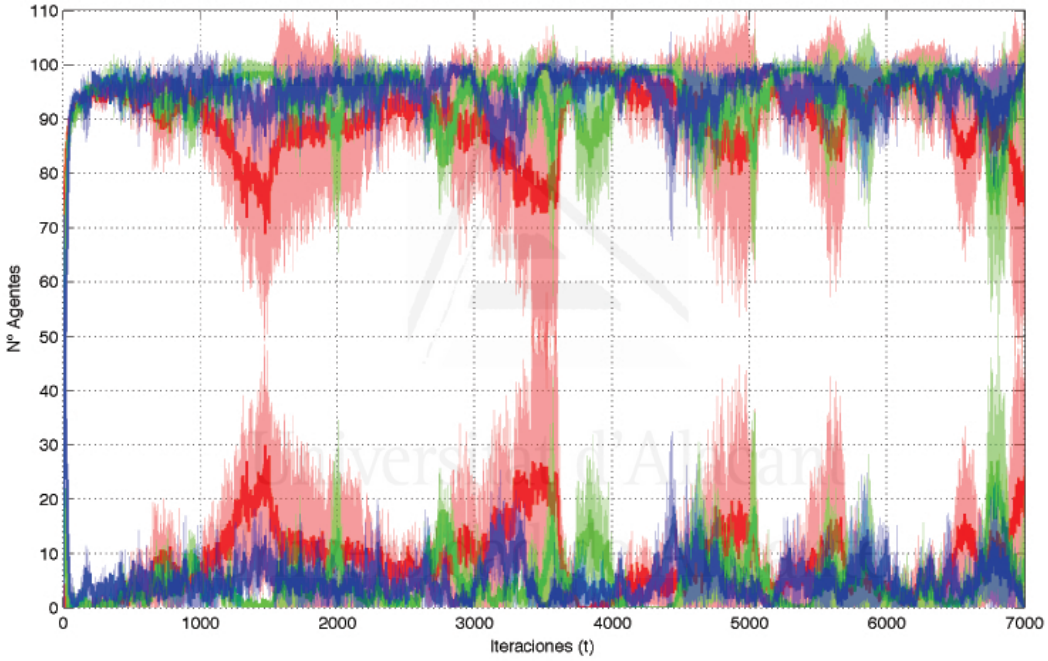


Figura 5.24: Cantidad de agentes es los estados *alineación* (parte superior) y *separación* (parte inferior) para el comportamiento de *flocking* con diferentes valores de R_s. Rojo: $R_s = 2$; Verde: $R_s = 5$; Azul: $R_s = 8$. Ejemplo para un enjambre de 100 robots y un entorno de tamaño 150×150 donde se muestra la media y la desviación típica de 5 ejecuciones.

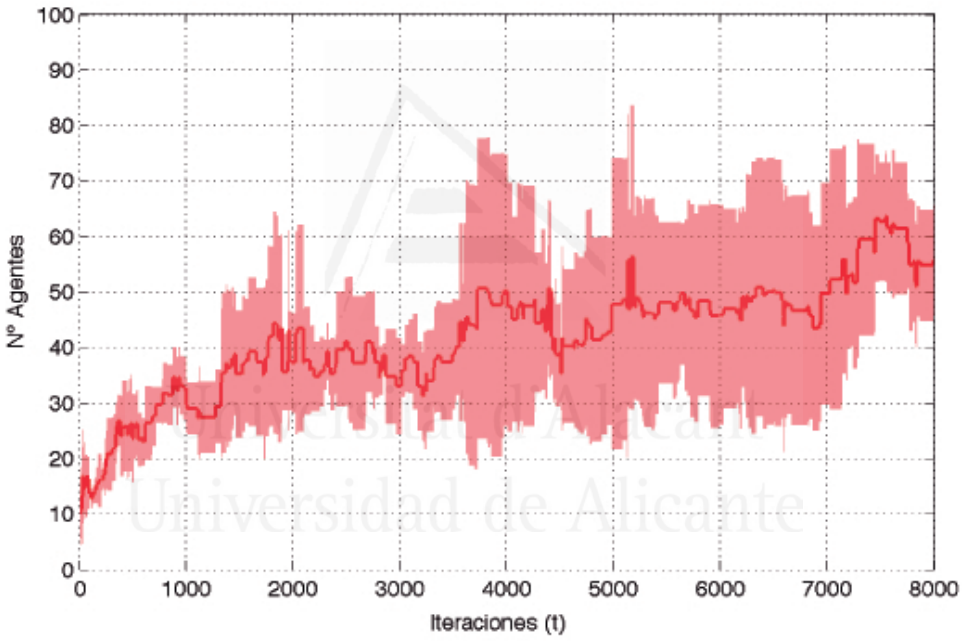


Figura 5.25: Tamaño del grupo más grande cuando R_a es cercano a R_s. Ejemplo para un enjambre de 100 robots y un entorno de tamaño 150×150 donde se muestra la media y la desviación típica de 5 ejecuciones.

Cambios en el valor de R_c no afectan al comportamiento del enjambre. El reducir o incrementar este valor sólo varía la distancia a la que un robot comienza el comportamiento de acercarse al robot que ha detectado, obteniendo los mismos resultados que las pruebas realizadas anteriormente.

5.2.3. Discusión

Para estudiar el comportamiento de movimiento coordinado (*flocking*) se han realizado diversas pruebas. En el primer conjunto de pruebas se ha estudiado que el comportamiento definido fuera correcto, comprobando los intercambios entre estados. En el segundo conjunto de pruebas se ha estudiado el comportamiento del enjambre ante obstáculos en el entorno. Y en el tercer grupo de pruebas se ha analizado el comportamiento ante cambios en los valores de R_s, R_a y R_c.

Las primeras pruebas demuestran el correcto funcionamiento del enjambre. Estas pruebas han permitido comprobar que los intercambios entre estados del comportamiento definido son adecuados. Mostrándose en la figura 5.16 cómo una vez estabilizado el enjambre los agentes se mantienen en los estados *alineación-separación-alineación* para mantener el movimiento coordinado del grupo. También se ha visto que el enjambre es capaz de formar un único grupo y mantener esta formación en movimiento, como se observa en la figura 5.17, que muestra el tamaño máximo del enjambre.

En el segundo conjunto de pruebas se ha verificado el comportamiento del enjambre ante obstáculos en el entorno. Estas pruebas han demostrado que el enjambre es capaz de mantener este comportamiento incluso cuando se encuentra un elemento en el entorno. El enjambre es capaz de evitar el obstáculo y continuar con su comportamiento. Se ha visto que para enjambres formados por un elevado número de agentes, es posible que al encontrar un obstáculo el enjambre se divida en varios grupos, pero una vez evitado el obstáculo el enjambre vuelve a unirse en un único grupo, como se ha visto en la figura 5.22 que muestra el tamaño máximo de grupo para una única simulación.

Por último, el tercer conjunto de pruebas se han realizado para estudiar el comportamiento del enjambre ante variaciones en los valores que definen el tamaño de las áreas de actuación *separación*, *alineación* y *cohesión*, R_s, R_a y R_c, respectivamente. El valor de R_s establece la cohesión del grupo, cómo de unidos estarán los agentes. Cuando este valor R_s es muy pequeño se da el caso que los agentes estarán muy juntos y se producirán más intercambios entre los estados *alineación-separación-alineación* al producirse más colisiones pero el comportamiento se ejecutará correctamente. El valor

de R_a afecta al comportamiento del grupo cuando este valor es muy similar a R_s. En este caso, prácticamente no existe zona de *alineación* y los robots tienden a separarse en pequeños grupos. Para otros valores de R_a y R_c no se producen cambios significativos en el comportamiento definido.

5.3. Dispersión. Cobertura de un área

La dispersión de un grupo de robots para maximizar la cobertura de un área es una tarea importante como paso inicial para tareas más complejas. Entre posibles aplicaciones podemos encontrar búsquedas, rescates, exploraciones, monitorización y vigilancia, limpieza y mantenimiento, etc. Este comportamiento ofrece la ventaja que los robots pueden ser utilizados en entornos peligrosos o de difícil acceso para las personas [JaGAJ10].

En [LLC11] presentan un algoritmo para la cobertura de un área dirigido a una tarea de monitorización de contaminación medioambiental. En este algoritmo la posición de los robots dependerá de la intensidad de polución y de la densidad de robots en el área. Los robots se moverán en dirección al punto con mayor nivel de polución detectado, evitando las zonas con mayor densidad de robots. A cada robot le asignan una energía inicial que va disminuyendo en cada paso. El robot se moverá mientras le quede energía. En [MG04] se centran en dispersar un grupo de robots en un entorno desconocido. Estudian cuatro algoritmos diferentes: paseo aleatorio, seguir pared, *seek open* y *fiducial*. En el algoritmo *seek open* los robots se mueven hacia zonas abiertas, donde detecten menor número de robots u obstáculos. El algoritmo *fiducial* asume que un robot es capaz de detectar a sus robots vecinos, de manera que un robot se alejará de los robots que detecte hacia zonas menos *pobladas*. En [SH08] estudian también varios algoritmos de dispersión de robots en un área. El primer algoritmo está basado en el movimiento aleatorio de las moléculas de gas en difusión. Cada robot tiene un vector de velocidad inicial, que sigue hasta que colisiona con otro robot u obstáculo. Entonces, calcula de manera aleatoria otro vector de velocidad y lo sigue hasta volver a encontrar un obstáculo. El robot repite este proceso indefinidamente. El segundo algoritmo está basado en la conservación del momento en una dispersión de gas. Este algoritmo actúa de la misma manera que el anterior con la diferencia de que cuando el robot encuentra un obstáculo y debe calcular un nuevo vector velocidad, este nuevo vector viene determinado por la conservación del momento. El tercer algoritmo está basado en minimizar la intensidad de dispersión. En este algoritmo cada robot emite una señal que puede ser detectada por los otros robots del enjambre. Los robots utilizan un

algoritmo modificado de optimización de partículas para buscar el mínimo valor de la señal emitida. Estos algoritmos los comparan con dos algoritmos de movimiento aleatorio. De manera similar a este último algoritmo, en [UTS07] y [LG06] presentan algoritmos de dispersión para maximizar la cobertura de un área a la vez que mantienen la conectividad del enjambre, basándose en la intensidad de la señal inalámbrica. Estos algoritmos se basan en un umbral de manera que cuando un robot detecta demasiada señal inalámbrica se aleja de la zona y cuando la intensidad de señal es muy baja se acerca para no perder la conectividad.

El objetivo principal de este problema es maximizar la exploración de un área minimizando el tiempo empleado para ello. En el siguiente punto se definen varias estrategias de dispersión de un enjambre.

5.3.1. Definición del comportamiento

Con el fin de estudiar el comportamiento de dispersión para un enjambre de robots, se estudiarán tres comportamientos diferentes para la exploración de un área.

En muchos de los trabajos descritos anteriormente se utiliza un comportamiento de paseo aleatorio para comparar su funcionamiento con otros comportamientos de exploración. Por lo tanto, en primer lugar, se define un comportamiento básico de movimiento aleatorio que servirá como base y elemento de comparación con el resto de comportamientos. A continuación se utilizará el algoritmo de *flocking* visto en el punto 5.2 orientado a una tarea de exploración. Por último, se define un comportamiento para la exploración del entorno basado en la densidad de robots de la zona, donde el robot de dirigirá hacia áreas abiertas (con menos cantidad de robots).

Paseo aleatorio

En primer lugar se ha definido un comportamiento básico de movimiento aleatorio. En este comportamiento los robots se mueven libremente por el entorno evitando los obstáculos que puedan encontrarse. Esta estrategia se define mediante una máquina de estados finita con dos únicos estados, como se puede observar en la figura 5.26. Los robots comienzan en el estado *Paseo aleatorio* moviéndose por el entorno aleatoriamente, cuando detectan un obstáculo u otro robot cerca, entran el estado *Evitar obstáculos*. Cuando el robot ha evitado el obstáculo vuelve al estado de *Paseo aleatorio*.

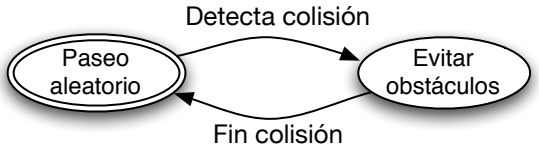


Figura 5.26: Diagrama del comportamiento de *Paseo aleatorio*.

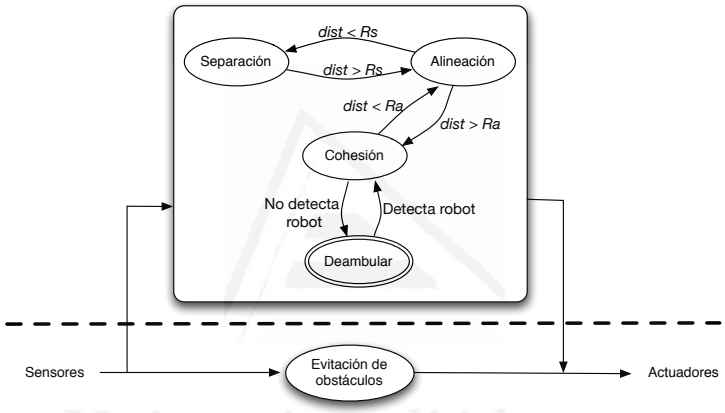


Figura 5.27: Diagrama del comportamiento de *Flocking*.

Movimiento coordinado

Se utilizará el comportamiento de movimiento coordinado (*flocking*) visto en el punto 5.2 para una tarea de exploración de un área. Este comportamiento está basado en tres zonas de actuación *Separación*, *Alineación* y *Cohesión* y ha sido definido en una arquitectura de subsunción de dos capas como se muestra en la figura 5.27. El funcionamiento de este comportamiento está definido en la sección 5.2.

Áreas abiertas

En este comportamiento los robots seguirán un movimiento aleatorio mientras no detecten ningún otro agente. Cuando un robot detecta otros agentes en su entorno, se dirigirá hacia áreas más abiertas, es decir, hacia áreas donde no se encuentren agentes. En este comportamiento los robots tienden a dispersarse de manera uniforme por el entorno ya que el robot será capaz de detectar la densidad de robots en el entorno y se dirigirá hacia

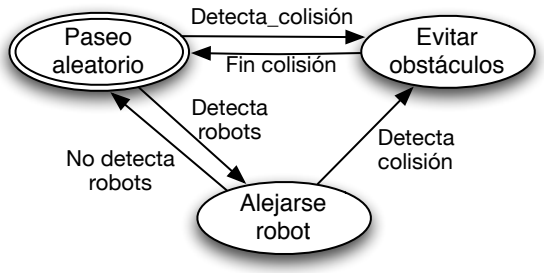


Figura 5.28: Diagrama del comportamiento de *áreas abiertas*.

áreas no ocupadas. Este comportamiento se define mediante una máquina de estados finita con tres estados, como se muestra en la figura 5.28.

El robot empieza su ejecución en el estado *paseo aleatorio*. En este estado el robot se moverá aleatoriamente por el entorno. Si detecta un obstáculo entrará en el estado *evitar obstáculos*. Cuando se evite el obstáculo, pasará al estado de *paseo aleatorio*. Si detecta otros robots, el robot se alejará de ellos, dirigiéndose a las áreas donde se encuentran menos robots, y volverá a seguir un comportamiento de paseo aleatorio.

Todos estos comportamientos han sido desarrollados utilizando únicamente la parte de enjambre de la arquitectura, los agentes μ, ya que no es necesario ningún mecanismo de coordinación llevado a cabo por los agentes superiores.

5.3.2. Experimentación

Las pruebas llevadas a cabo para analizar estos comportamientos se dividen en dos grupos. En el primer conjunto de pruebas los agentes se inicializan aleatoriamente en el entorno. Sin embargo, en el segundo conjunto de pruebas, todos los agentes comienzan su ejecución en la misma área.

En ambos conjuntos de prueba se utilizan 50 robots para tres tamaños de entorno diferentes: 100×100, 150×150 y 200×200. Se estudiará el tiempo que tarda cada enjambre en cubrir todo el entorno. Se muestran los resultados de 5 simulaciones para cada conjunto de valores.

Inicialización aleatoria

En el primer conjunto de pruebas los robots se inicializan de manera aleatoria en el entorno, como se muestra en la figura 5.29.

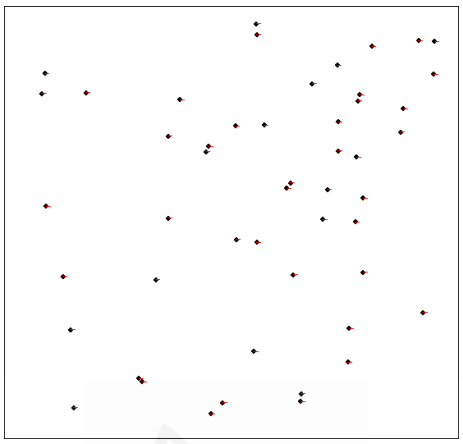


Figura 5.29: Ejemplo de inicialización aleatoria para un enjambre de 50 robots en un entorno de tamaño 100 × 100.

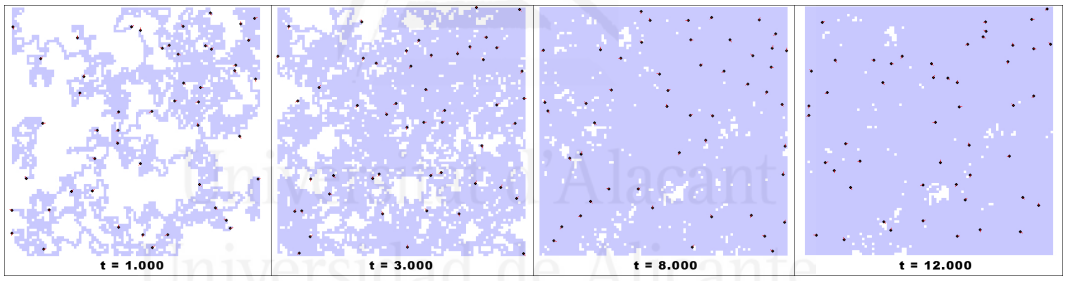


Figura 5.30: Secuencia de ejecución del comportamiento de paseo aleatorio para un enjambre de 50 robots en un entorno de tamaño 100 × 100.

En el comportamiento de paseo aleatorio los robots se mueven por el entorno de manera aleatoria. La figura 5.30 muestra una secuencia de este comportamiento. En color azul se indica la zona ya explorada. Cuando un robot pasa por un área, ésta queda señalada como explorada.

En el gráfico de la figura 5.31 se puede observar el porcentaje de entorno que ha sido explorado en el tiempo para tres entornos de diferentes tamaños en el comportamiento de paseo aleatorio. Se observa cómo el tiempo necesario para explorar el área es mayor cuanto mayor es el tamaño del entorno. El gráfico muestra cómo para el entorno de menor tamaño (100 × 100) sobre las 15.000 iteraciones se ha cubierto el 100 % del entorno, mientras que para el entorno de mayor tamaño (200 × 200) el porcentaje explorado en ese mismo número de iteraciones es sobre el 70 %, aunque ese porcentaje va

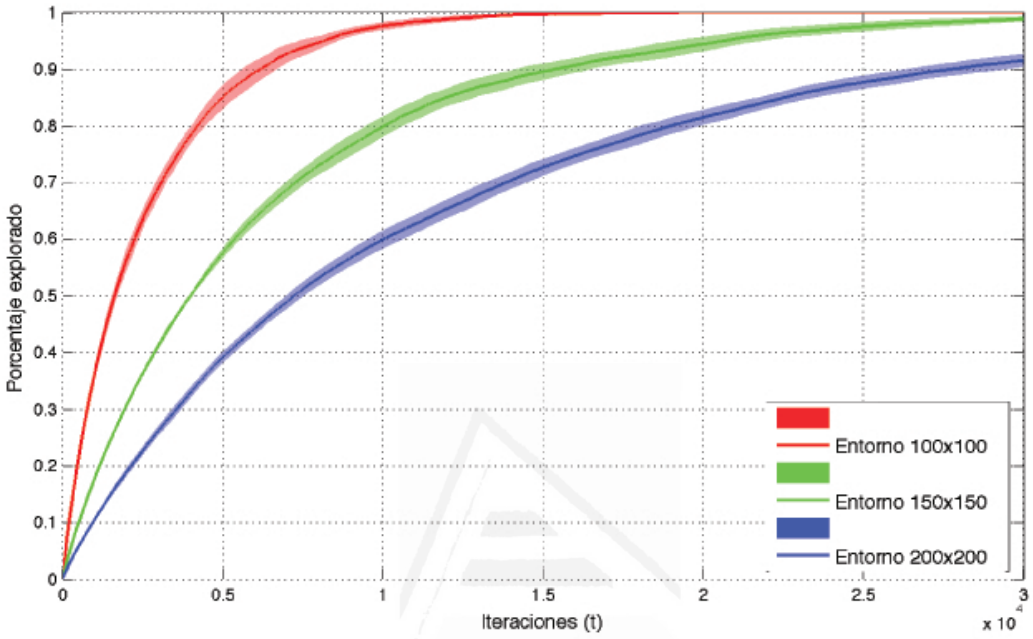


Figura 5.31: Porcentaje explorado para un enjambre de 50 robots en entornos de diferentes tamaños (100×100, 150×150 y 200×200) para el comportamiento de paseo aleatorio. Se muestra la media y la desviación típica de 5 ejecuciones para cada tamaño de entorno.

incrementando de manera continua en el tiempo.

En el comportamiento de *flocking* los robots forman un grupo coordinado capaz de moverse de manera aleatoria por el entorno. La figura 5.32 muestra una secuencia de este comportamiento para diferentes instantes de tiempo.

En el gráfico 5.33 se observa el porcentaje de entorno que ha sido explorado para un enjambre de 50 robots y tres entornos de diferentes tamaños. Como en el comportamiento de paseo aleatorio el tiempo necesario para cubrir todo el entorno es mayor cuanto mayor es el entorno aunque este porcentaje aumenta progresivamente con el tiempo para todos los entornos.

En el comportamiento de áreas abiertas los robots se dispersan por el entorno separándose unos de otros. En la figura 5.34 se observa una secuencia de este comportamiento en diferentes instantes de tiempo.

Como en los comportamientos anteriores en la figura 5.35 se muestra que el tiempo de exploración necesario es mayor cuanto mayor es el tamaño. En este gráfico se observa cómo la desviación mostrada es menor que en los

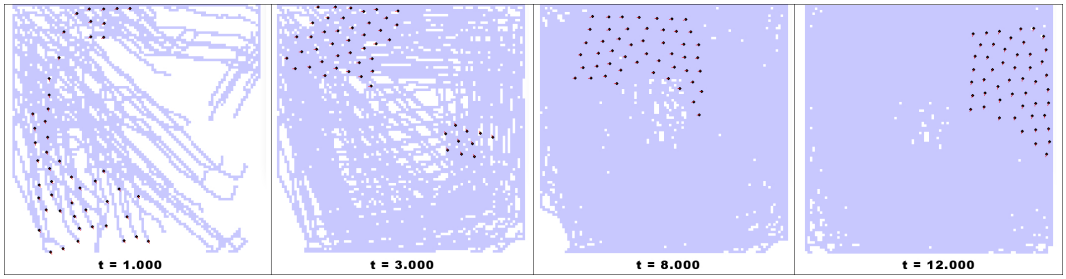


Figura 5.32: Secuencia de ejecución del comportamiento de flocking para un enjambre de 50 robots en un entorno de tamaño 100×100.

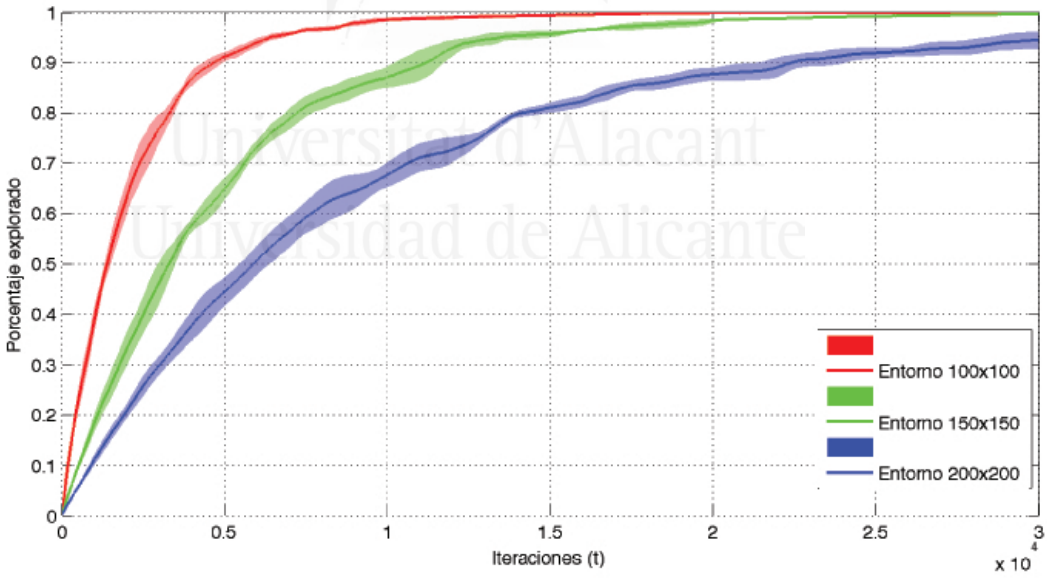


Figura 5.33: Porcentaje explorado para un enjambre de 50 robots en entornos de diferentes tamaños (100×100, 150×150 y 200×200) para el comportamiento de *flocking*. Se muestra la media y la desviación típica de 5 ejecuciones para cada tamaño de entorno.

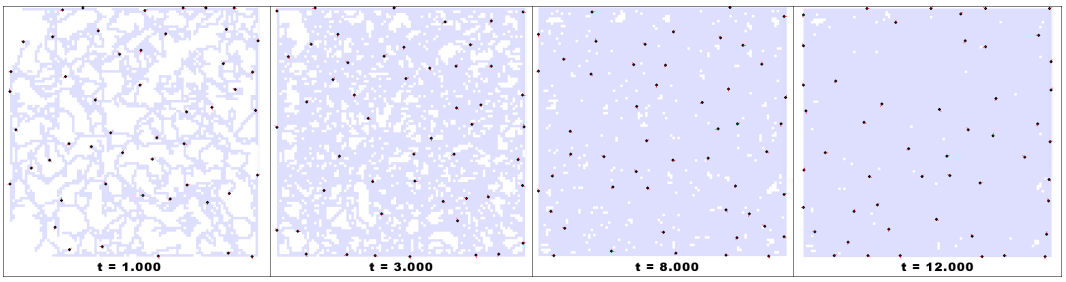


Figura 5.34: Secuencia de ejecución del comportamiento de áreas abiertas para un enjambre de 50 robots en un entorno de tamaño 100×100.

comportamientos anteriores para todos los entornos. Es decir, para todas las simulaciones el enjambre se comporta de manera muy similar.

La figura 5.36 muestra una comparativa de los tres comportamientos para los diferentes tamaños de entorno. En la parte superior izquierda se observa el funcionamiento de los tres comportamientos para un entorno de tamaño 100×100, donde se puede observar que todos los comportamientos se comportan de manera similar. En la parte superior derecha se muestra el resultado de los comportamientos para un entorno de tamaño 150×150, donde se puede observar que el comportamiento de *flocking* y el de áreas abiertas tienen un comportamiento muy similar. Y, por último, en la parte inferior, se muestran los resultados para un entorno de 200×200, donde se observa que el comportamiento de áreas abiertas mejora su rendimiento en comparación con los otros dos comportamientos para tamaños de entorno mayores. En los tres gráficos se muestra que el paseo aleatorio es el que obtiene peores resultados en todos los casos.

Inicialización en un área

Cuando se trabaja con robots reales es común que los robots inicien su ejecución en un mismo punto y no de manera aleatoria en el entorno. Este segundo conjunto de pruebas tiene en cuenta esta inicialización de los robots en un área concreta del entorno. En este caso, la posición inicial del enjambre forma una matriz de agentes en una zona determinada del entorno, como se observa en la figura 5.37.

La figura 5.38 muestra una secuencia del comportamiento de paseo aleatorio para esta inicialización. Se observa como los robots exploran en mayor medida el área cercana a la zona de inicio.

El gráfico de la figura 5.39 muestra el porcentaje explorado por el com-

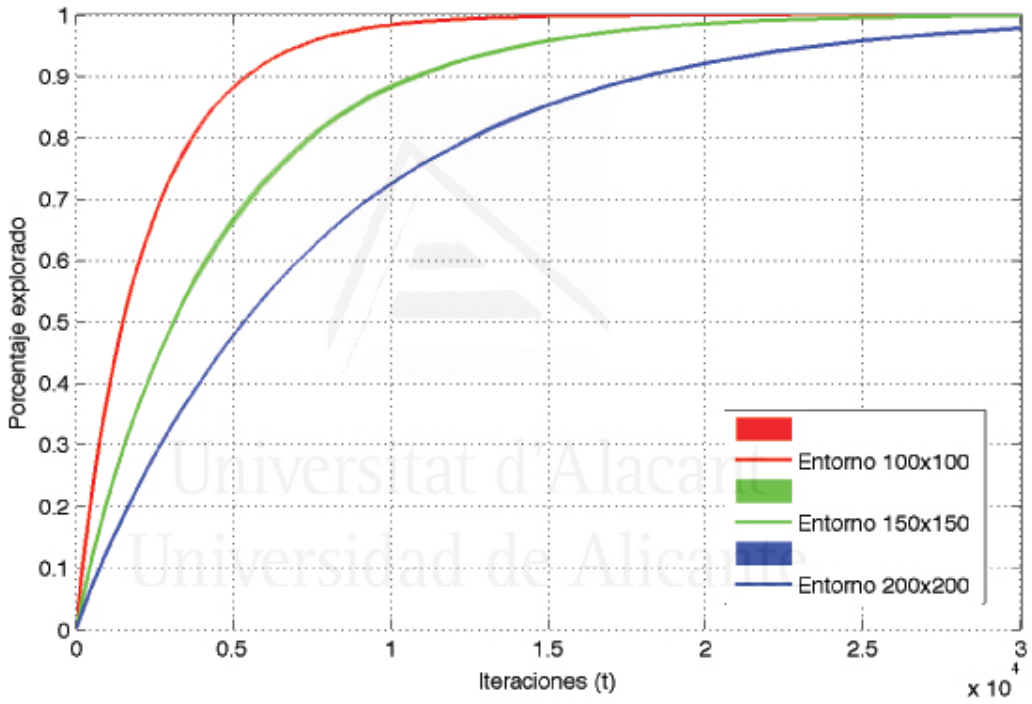


Figura 5.35: Porcentaje explorado para un enjambre de 50 robots en entornos de diferentes tamaños (100×100, 150×150 y 200×200) para el comportamiento de áreas abiertas. Se muestra la media y la desviación típica de 5 ejecuciones para cada tamaño de entorno.

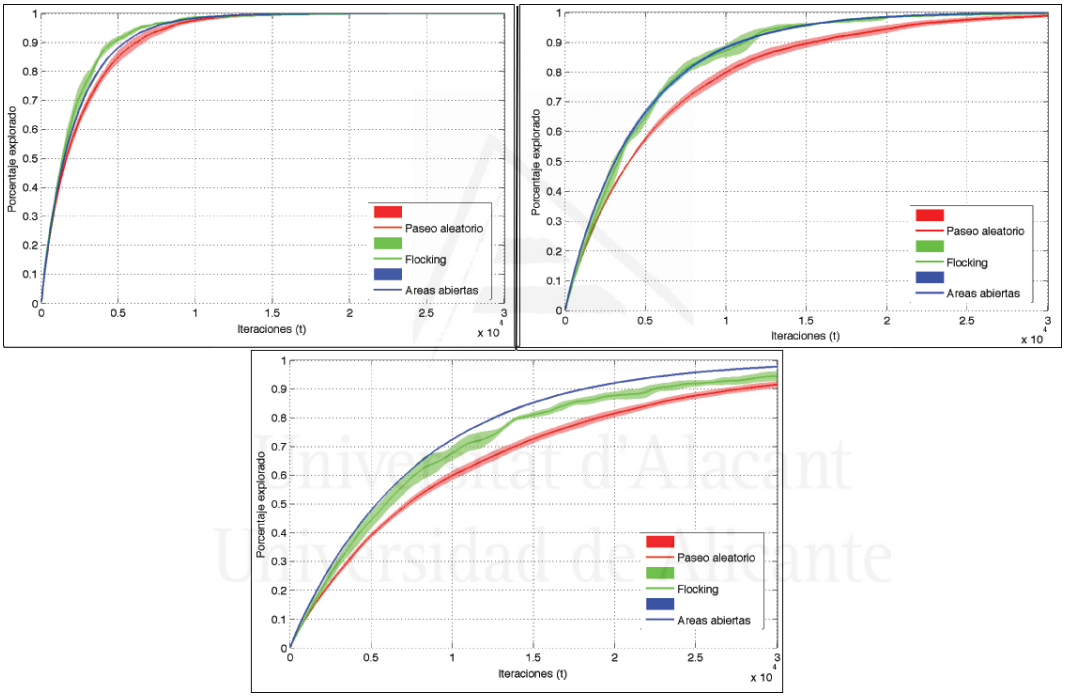


Figura 5.36: Porcentaje explorado para un enjambre de 50 robots para los tres comportamientos en entornos de diferentes tamaños: 100×100 superior izquierda, 150×150 superior derecha y 200×200 inferior. Se muestra la media y la desviación típica de 5 ejecuciones para cada tamaño de entorno.

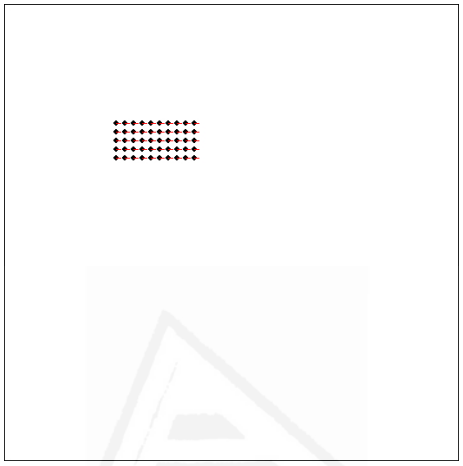


Figura 5.37: Ejemplo de inicialización en un área para un enjambre de 50 robots en un entorno de tamaño 100×100.

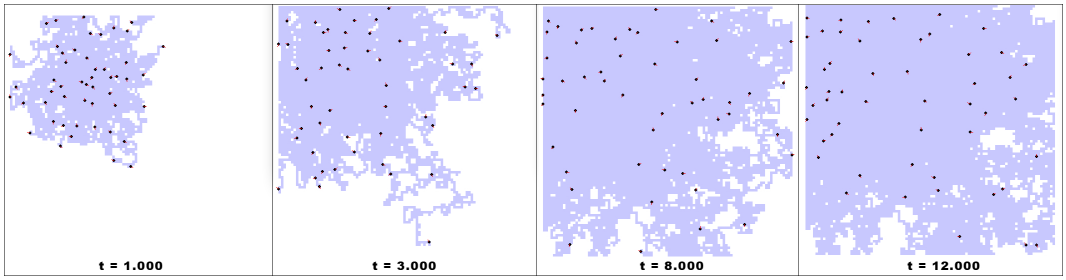


Figura 5.38: Secuencia de ejecución del comportamiento de paseo aleatorio para un enjambre de 50 robots en un entorno de tamaño 100×100 con una inicialización en un área.

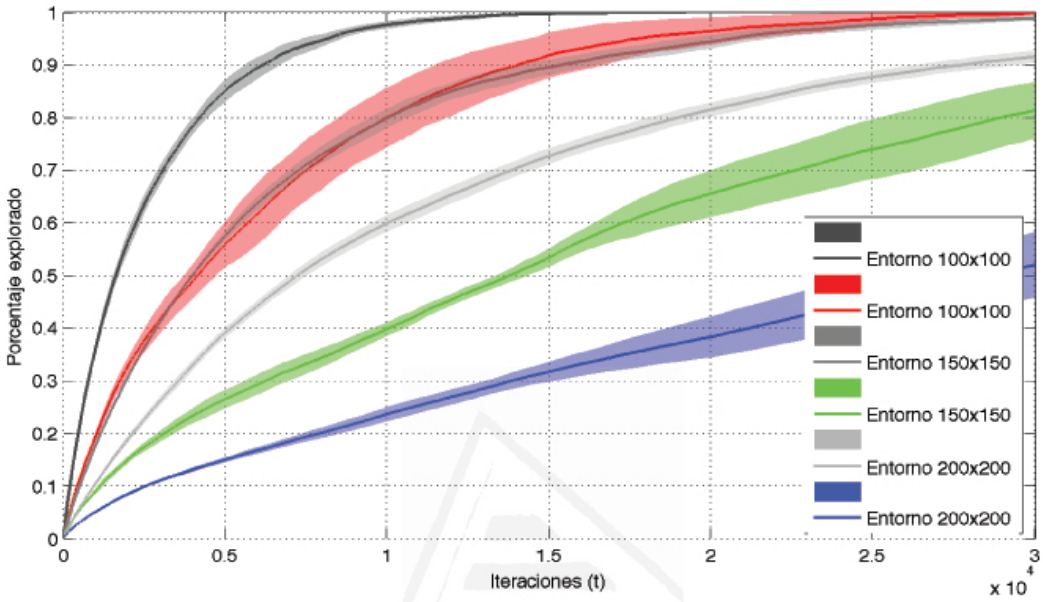


Figura 5.39: Porcentaje explorado para un enjambre de 50 robots en entornos de diferentes tamaños (100 × 100, 150 × 150 y 200 × 200) para el comportamiento de paseo aleatorio con inicialización en un área. Se muestra la media y la desviación típica de 5 ejecuciones para cada tamaño de entorno.

portamiento de paseo aleatorio para tres tamaños de entorno. En color se muestran los resultados del comportamiento para esta inicialización y en escala de grises se muestran los resultados obtenidos en el conjunto de pruebas anteriores para una inicialización aleatoria. Se observa que en este caso el comportamiento tiene un rendimiento más bajo, necesitando más tiempo para realizar la misma tarea. Además, también se observa cómo el funcionamiento de este comportamiento empeora para tamaños de entorno grandes.

En el comportamiento de *flocking* los robots forman grupos capaces de moverse conjuntamente desde el principio de la simulación. En el momento inicial se forman varios grupos para evitar posibles colisiones dado que los robots están demasiado cerca. Posteriormente, forman un único grupo como se puede observar en la secuencia mostrada en el figura 5.40.

Como se puede observar en el gráfico mostrado en la figura 5.41, el comportamiento de *flocking* tiene un funcionamiento similar tanto para la inicialización aleatoria (escala de grises) como para la inicialización en una zona

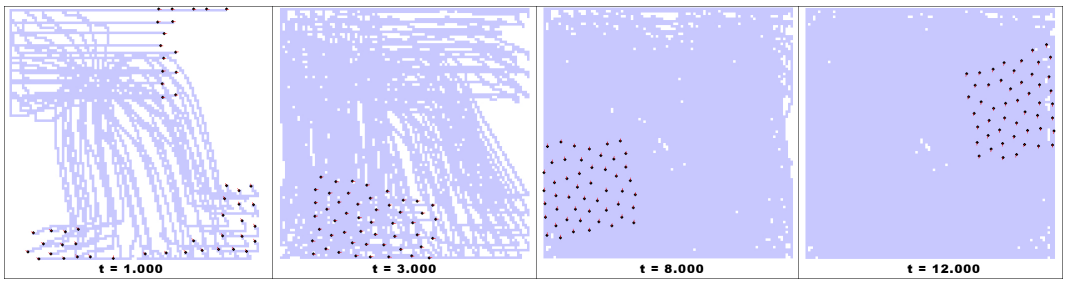


Figura 5.40: Secuencia de ejecución del comportamiento de *flocking* para un enjambre de 50 robots en un entorno de tamaño 100 × 100 con una inicialización en un área.

determinada (gráfico en color) para los entornos de tamaño más pequeño. Para el entorno de mayor tamaño (200×200) la inicialización aleatoria muestra una leve mejora respecto al tiempo necesario para explorar el entorno.

Respecto al comportamiento de áreas abiertas, en la secuencia de imágenes del comportamiento mostrada en la figura 5.42 se puede observar cómo el enjambre se dispersa rápidamente por el entorno, dado que los robots tienden a ocupar áreas donde se encuentran menos robots.

Como ocurre con el comportamiento de *flocking*, el comportamiento de áreas abiertas se comporta de manera similar, tanto para la inicialización aleatoria, como en la inicialización en un área determinada, para los tamaños de entorno más pequeños, como se observa en la figura 5.43. En los entornos de mayor de tamaño el tiempo necesario el levemente mayor.

La figura 5.44 muestra una comparativa de los tres comportamientos para los diferentes tamaños de entorno. En la parte superior izquierda se observa el funcionamiento de los tres comportamientos para un entorno de tamaño 100 × 100. En la parte superior derecha se muestra el resultado de los comportamientos para un entorno de tamaño 150 × 150. Y, por último, en la parte inferior, se muestran los resultados para un entorno de 200 × 200. En los tres gráficos se muestra claramente que el comportamiento de paseo aleatorio es el que obtiene peores resultados en todos los casos, mientras que el comportamiento de *flocking* y el de áreas abiertas muestran un funcionamiento similar en todos los casos.

5.3.3. Discusión

Para estudiar el problema de cobertura de áreas se han definido tres comportamientos: un comportamiento de paseo aleatorio, un comportamiento de

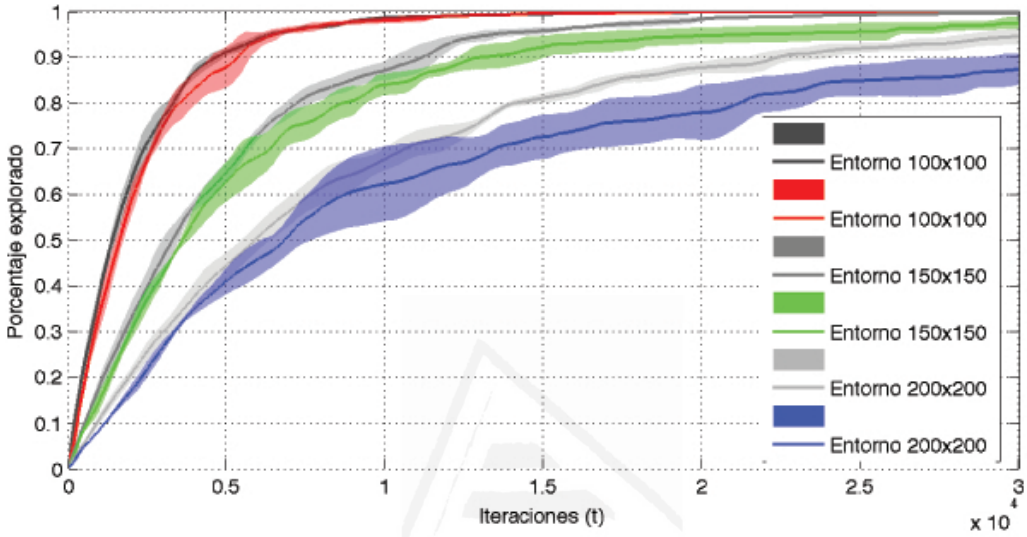


Figura 5.41: Porcentaje explorado para un enjambre de 50 robots en entornos de diferentes tamaños (100×100, 150×150 y 200×200) para el comportamiento de *flocking* con inicialización en un área. Se muestra la media y la desviación típica de 5 ejecuciones para cada tamaño de entorno.

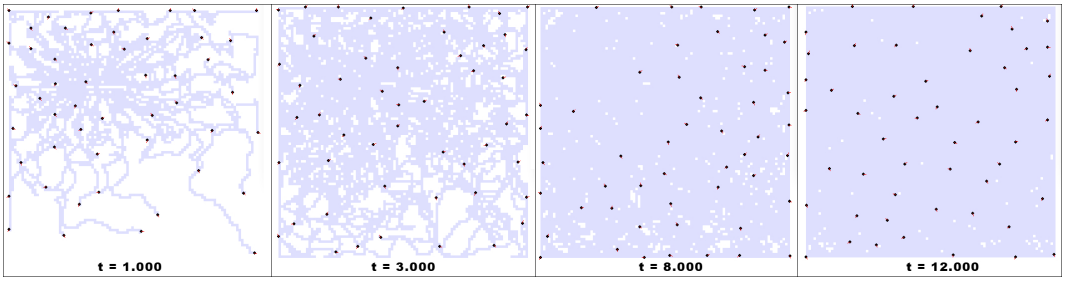


Figura 5.42: Secuencia de ejecución del comportamiento de áreas abiertas para un enjambre de 50 robots en un entorno de tamaño 100×100 con una inicialización en un área.

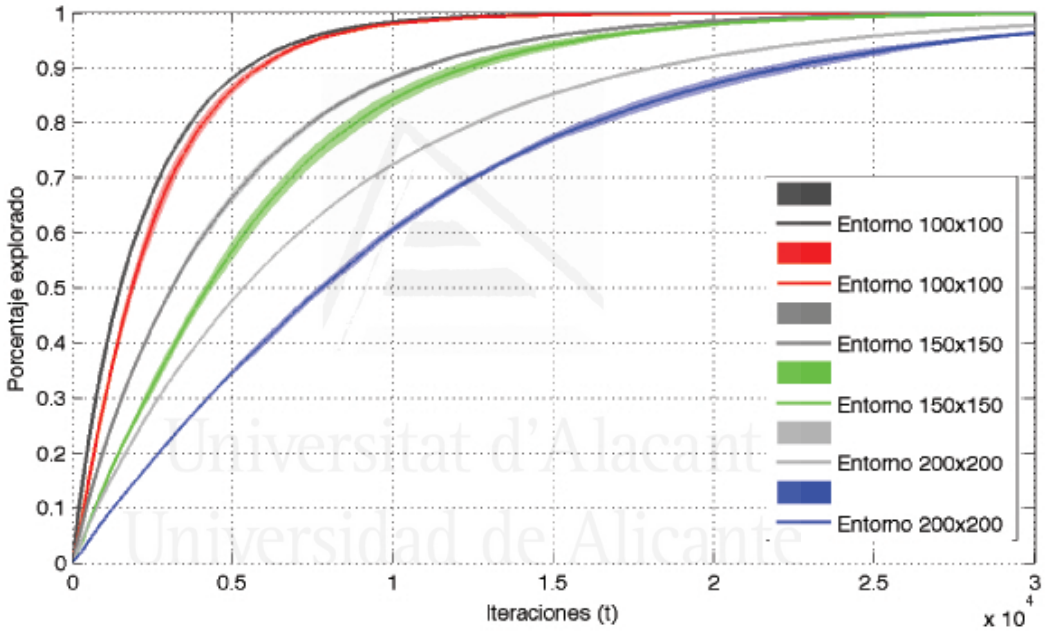


Figura 5.43: Porcentaje explorado para un enjambre de 50 robots en entornos de diferentes tamaños (100 × 100, 150 × 150 y 200 × 200) para el comportamiento de áreas abiertas con inicialización en un área. Se muestra la media y la desviación típica de 5 ejecuciones para cada tamaño de entorno.

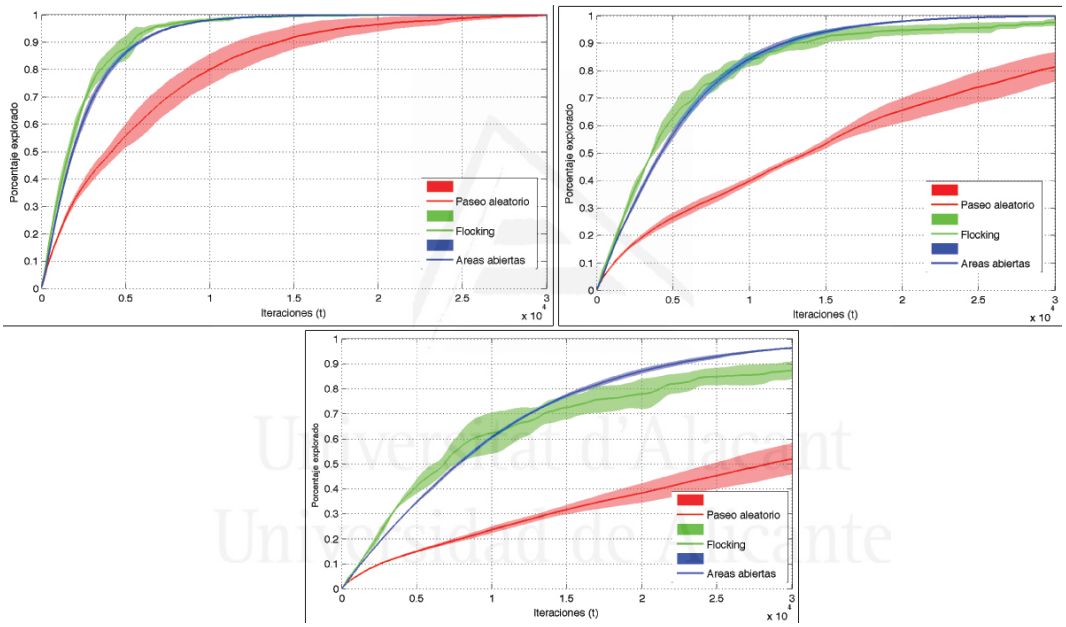


Figura 5.44: Porcentaje explorado para un enjambre de 50 robots para los tres comportamientos en entornos de diferentes tamaños: 100×100 superior izquierda, 150×150 superior derecha y 200×200 inferior, con inicialización en un área. Se muestra la media y la desviación típica de 5 ejecuciones para cada tamaño de entorno.

flocking y un comportamiento de áreas abiertas, donde los robots se dirigen hacia zonas con menos robots.

Las pruebas realizadas se dividen en dos grupos. En el primer conjunto de pruebas se ha estudiado el funcionamiento de estos tres comportamientos ante una inicialización aleatoria de los robots en el entorno. En el segundo conjunto de pruebas se ha estudiado el funcionamiento de los comportamientos teniendo en cuenta que en situaciones reales, de manera general, los robots iniciarán su ejecución desde una misma zona.

Las primeras pruebas demuestran el correcto funcionamiento del enjambre en los tres comportamientos definidos. Todos los comportamientos son capaces de cubrir el 100 % del entorno independientemente del tamaño de éste (siempre que se disponga de tiempo suficiente). Como se ha mostrado en las graficas de la figura 5.36 los tres comportamientos tienen un rendimiento similar, en cuanto al tiempo necesario para explorar todo el enjambre. El comportamiento de paseo aleatorio es el que obtiene peores resultados pero éstos no distan mucho de los resultados obtenidos con los otros comportamientos.

El segundo conjunto de pruebas demuestran el correcto funcionamiento del enjambre para una inicialización en una zona determinada. En este caso, los comportamientos también son capaces de cubrir el 100 % del entorno, siempre que se disponga de tiempo suficiente. En este caso, como se ha mostrado en las gráficas de la figura 5.44 es posible observar una clara diferencia entre el comportamiento de paseo aleatorio y el resto de comportamientos. Mientras que el comportamiento de *flocking* y el de áreas abiertas tienen un funcionamiento similar, el comportamiento de paseo aleatorio obtiene peores resultados, en cuanto al tiempo necesario para cubrir el entorno. Al realizarse la inicialización de los robots en una zona, en este comportamiento, los robots tienden a explorar en mayor medida la zona cercana a la inicialización. Sin embargo, en el comportamiento de áreas abiertas, los robots tienden a dispersarse por el entorno conforme van detectando otros robots.

Para una inicialización en un área determinada del entorno el comportamiento de *flocking* y el comportamiento de áreas abiertas tienen un funcionamiento similar, aunque para los entornos de mayor tamaño, igual que ocurre en la inicialización aleatoria, el comportamiento de áreas abiertas tiene un rendimiento levemente mejor.

5.4. Resumen

En este capítulo se han definido y estudiado tres problemas clásicos de la robótica de enjambre: agregación; movimiento coordinado; y dispersión.

Se ha definido un comportamiento de agregación auto-organizada donde los individuos son capaces de agregarse teniendo en cuenta la información percibida por sus sensores. Para este comportamiento se ha definido una arquitectura de subsunción de dos capas para los agentes del enjambre: en la capa inferior se sitúa un comportamiento básico de evitación de obstáculos; la capa superior se compone de una máquina de estados finita con 3 posibles estados (*repeler*, *esperar*, *aproximarse*). El funcionamiento básico de este comportamiento es que cuando un robot detecta otro robot se acerca a éste y entra en estado *esperar*. Un robot saldrá de este estado *esperar* dependiendo de un cierto valor P. Según el valor de P el enjambre tendrá comportamientos diferentes.

En las pruebas realizadas se ha estudiado el comportamiento del enjambre para tres posibles situaciones de P. En el primer conjunto de pruebas se ha utilizado una probabilidad de transición constante (P constante) que ha permitido comprobar el correcto intercambio entre estados de los agentes, pero sin obtener buenos resultados en cuanto al comportamiento de agregación. En el segundo conjunto de pruebas se ha utilizado un valor de P dependiente del tamaño del grupo en el que se encuentra el agente. En estas pruebas se ha obtenido un mejor funcionamiento del comportamiento, llegando a obtenerse grupos grandes, pero sin conseguir en todas las simulaciones un único grupo agregado. En el último conjunto de pruebas se ha utilizado una probabilidad de transición que tiene en cuenta no únicamente el tamaño del grupo en el que se encuentra el robot, sino también el tamaño del grupo más grande. De esta manera, los robots que pertenecen al grupo más grande tienen menos probabilidad de abandonarlo. El uso de agentes Ψ permite proporcionar al enjambre esta información necesaria para su tarea. Estas pruebas han demostrado un funcionamiento correcto del comportamiento formándose agregaciones con todos los agentes del enjambre.

También ha sido definido un comportamiento de movimiento coordinado (*flocking*). Este comportamiento se basa en tres zonas de actuación: separación, alineación y cohesión. De esta manera, un agente tendrá un comportamiento diferente en función de la zona en la que se encuentre respecto a sus vecinos. Este comportamiento se ha definido mediante una arquitectura de subsunción de dos capas: en la capa inferior se sitúa un comportamiento básico de evitación de obstáculos; y en la capa superior se sitúa una máquina de estados finita con 3 posibles estados (*separación*, *alineación* y *cohesión*)

que hacen referencia a las tres zonas de actuación. Estas zonas se definen en base a tres distancias: R_s, R_a, y R_c, respectivamente.

Las pruebas realizadas para comprobar el funcionamiento de este comportamiento se dividen en tres grupos. En las primeras pruebas se ha comprobado el funcionamiento correcto del comportamiento para entornos vacíos. En las segundas pruebas se ha comprobado su funcionamiento en entornos con obstáculos. En el tercer conjunto de pruebas se ha estudiado cómo afectan cambios en los valores R_s, R_a y R_c. Se ha visto que la distancia R_s establece la separación entre los robots, cuando este valor es pequeño los robots se sitúan muy próximos entre sí, cuando R_s es mayor aumenta la distancia entre éstos. Cambios en este valor no afectan al comportamiento del enjambre. Los cambios más significativos se dan cuando R_s y R_a son valores similares, porque no se establece un área de alineación, lo que conlleva que los robots no sean capaces de coordinar su movimiento con el de sus vecinos y se produzcan separaciones en el grupo.

Por último, para estudiar el problema de cobertura de un área se han definido tres comportamientos diferentes: aleatorio, donde los robots se mueven libremente por el entorno; *flocking*; y áreas abiertas, donde los robots se dispersan por el entorno hacia áreas con menos robots.

Para este problema se han realizado dos conjuntos de pruebas. En el primer conjunto de pruebas el enjambre se inicializa de manera aleatoria en el entorno. En el segundo conjunto de pruebas todos los agentes inician su ejecución desde una misma área del entorno. En las primeras pruebas los tres comportamientos han demostrado un funcionamiento similar, obteniéndose resultados levemente mejores en el comportamiento de áreas abiertas pero todos ellos cubriendo el 100 % del entorno. En el segundo conjunto de pruebas se observa una mayor diferencia en el funcionamiento entre los comportamientos. Mientras que los comportamientos de *flocking* y áreas abiertas obtienen unos resultados similares, el comportamiento aleatorio obtiene claramente peores resultados.

Se ha visto cómo el comportamiento global del enjambre emerge a partir de la definición de un comportamiento sencillo a nivel local.

En el siguiente capítulo se definirá un comportamiento de enjambre para una tarea específica de localización de recursos en el entorno.

Capítulo 6

Sistema de enjambre para localización de recursos

Como se ha visto anteriormente la robótica de enjambre es una aproximación descentralizada a la coordinación de sistemas multi-robóticos, donde el comportamiento colectivo emerge de la interacción local entre los agentes y su entorno. Este tipo de coordinación descentralizada permite que un sistema desarrolle sus objetivos globales embebiendo los patrones de comportamiento simples, posiblemente inspirados por la naturaleza, a nivel de cada agente individual que forma el sistema.

Cualquier tarea en la cual existan una serie de objetos distribuidos físicamente que necesitan ser explorados, inspeccionados, recolectados, completados, rescatados o montados en estructuras es una aplicación potencial de la robótica de enjambre. Por este motivo, la explotación colectiva de recursos es una de las aplicaciones que ha tomado especial relevancia en este tipo de sistemas, pudiéndose encontrar varias aplicaciones que utilizan la robótica de enjambre para solucionar tareas de búsqueda y explotación de recursos. La mayoría de estas aplicaciones se basan en agentes complejos que requieren comunicación explícita entre ellos. Estas características hacen que este tipo de sistemas sean difíciles de implantar en ciertos entornos, donde no siempre se puede establecer comunicación entre agentes y donde sería deseable disponer de un enjambre de gran tamaño. En este capítulo se presenta un sistema de enjambre para la explotación colectiva de recursos donde los agentes que lo forman se caracterizan por su simplicidad y por no comunicarse entre ellos de manera explícita.

La finalidad principal de este capítulo es aportar un sistema de enjambre sin conexión explícita (donde los agentes del enjambre no se pueden comu-

nicar entre sí) basado en comportamientos simples. Este sistema es capaz de localizar de manera emergente la fuente de recursos más prometedora en un entorno desconocido. El objetivo de cada uno de los agentes que forman el enjambre es señalar las fuentes de recursos.

Para este fin, se presentarán las capacidades de un agente individual proponiendo un modelo microscópico que describirá a cada agente. Posteriormente, se presentará un modelo macroscópico orientado a predecir la agregación del enjambre en el entorno. A continuación se describirán las pruebas realizadas orientadas a comprobar tanto el comportamiento microscópico definido como el macroscópico.

6.1. Introducción

Como se ha comentado, la explotación colectiva de recursos es una aplicación que ha tomado especial relevancia en este tipo de sistemas. Esta aplicación requiere, por una parte, la exploración del entorno para buscar un determinado tipo de recurso y, por otra, extraer dichos recursos y utilizarlos o transportarlos a la estación base del enjambre. Existen varias aproximaciones que utilizan la robótica de enjambre para solucionar esta tarea. La mayoría de estos estudios se basan en agentes que son capaces de comunicarse con otros agentes del enjambre (o bien mediante comunicación directa con el resto del enjambre o bien utilizando métodos indirectos como feromonas artificiales). En [LW09] presentan un modelo para un grupo heterogéneo de robots, donde utilizan dos variables globales que los robots deben ajustar, tiempo de búsqueda y tiempo de descanso. En [HW07] presentan un modelo para un enjambre homogéneo de robots que utilizan como medio de comunicación feromonas virtuales. En [SMC06] asumen capacidades de comunicación limitadas, pero los agentes se deben comunicar entre ellos para ir propagando la localización del recurso encontrado.

Además, es común que en este tipo de aplicaciones se utilice un grupo homogéneo de robots, normalmente reducido, con sistemas de percepción complejos (como láser o visión). En [LSPB05] los agentes del enjambre conocen la posición del resto de agentes, donde se indica que se pueden obtener estas posiciones con radares de alto alcance o mecanismos de visión. En [BS03] los robots van dejando en el entorno marcadores equipados con un procesador y una radio de alcance limitado; y los robots son capaces de comunicarse con estos marcadores.

A la hora de implantar este tipo de sistemas podemos encontrarnos con entornos en los cuales las capacidades de comunicación entre agentes pueden

ser limitadas (por ejemplo, en lugares donde la señal a transmitir puede ser obstaculizada por el mismo entorno, como en la exploración planetaria o en entornos donde la emisión de cierto tipo señales no sea posible por diversas causas). Puede ser fundamental analizar sistemas que requieran poca o incluso ninguna comunicación entre agentes, aunque esto sea a expensas de alargar el tiempo requerido para la exploración o la búsqueda de recursos.

A continuación se presenta un mecanismo de agregación basado en agentes simples sin capacidades de comunicación, para la explotación colectiva de recursos. El enjambre será capaz de localizar dinámicamente el área de recursos más prometedora en un entorno desconocido.

6.2. Definición del comportamiento

En el estudio realizado en [GM07] se especifica que con tres comportamientos simples: protector (un agente elegirá otros dos agentes del enjambre y se colocará entre ellos), refugiado (un agente elegirá dos agentes del enjambre y hará que uno de ellos esté en medio de él y el otro) y agresor (un agente perseguirá a otro agente), el enjambre puede desarrollar tres comportamientos diferenciados: expansión, navegación conjunta o agregación (en determinado punto del entorno).

Tomando como base este estudio se ha especificado una arquitectura con dos posibles estados: agresor, en este estado los agentes perseguirán a otro agente y; evasivo, en este estado los agentes huirán de sus agresores. De esta manera, si cada agente persigue a otro agente, el enjambre tendrá un comportamiento de agregación, por el contrario, si todos los agentes tienen un comportamiento evasivo el enjambre tenderá a expandirse.

Cada agente $r_i \in R$ está definido como un vector $r_i = (s_i, p_i, a_i)$ formado por estados $s_i \in S$, sensores $p_i \in P$ y actuadores $a_i \in A$. Cada agente tiene definidos dos actuadores $a_i = (v_{tras}, v_{rot})$ que permiten establecer su velocidad lineal y rotacional. Y cada agente tiene definidos tres sensores $p_i = (m_i, c_i, t_i)$, donde:

- m_i: permite al agente identificar su pareja en el enjambre. Cada robot tiene un compañero, de manera que $\forall m_i \in M, \forall m_j \in M, i \neq j, m_i \neq m_j$.
- $c_i \in [0, 1]$: este sensor indica la cantidad de recurso que existe en la posición actual del robot.
- $t_i \in [0, \infty[$: es el reloj interno del robot.

Cada agente implementará una máquina de estados probabilística con dos posibles estados $s_i \in \{expandir, agrupar\}$: *expandir* corresponde al comportamiento de los agentes evasivos, donde el agente r_i tiene que calcular el vector de repulsión respecto al robot m_i; y *agrupar* corresponde con el comportamiento de los agentes agresores, donde el agente calculará el vector de atracción a m_j. La manera más simple de calcular estos vectores requiere conocer la posición actual del robot pos_i y la de su compañero pos_{mi}, de manera que el vector de repulsión se puede definir como $\boldsymbol{v}_e = pos_i - pos_{m_i}, \boldsymbol{v}_{expand} = \frac{\boldsymbol{v}_e}{\|\boldsymbol{v}_e\|}$ y el vector de atracción se puede definir como $\boldsymbol{v}_g = pos_{m_i} - pos_i, \boldsymbol{v}_{group} = \frac{\boldsymbol{v}_g}{\|\boldsymbol{v}_g\|}$. Sin embargo, es posible calcular estos vectores sin conocer las posiciones del robot, utilizando únicamente métodos de localización indirectos. El cambio de un estado a otro es probabilístico y está guiado por los siguientes tres factores:

Energía del agente e**.** La energía inicial de cada agente es e_{init}. Esta energía decrece con el tiempo. La energía se define como $e = e_{init}\left(-\frac{-e_{init}+\Delta_e}{e_{init}}\right)^t$, siendo Δ_e el incremento de energía en cada unidad de tiempo t.

Probabilidad de agrupamiento p_a**.** Es la probabilidad de que un agente entre en estado *agrupar*. Conforme la energía del agente decrece, la probabilidad de agrupamiento debe ser mayor ya que el objetivo principal es que el enjambre converja en el punto del entorno donde la fuente de recursos sea mayor. Esta probabilidad p_a se define como $p_a = 1 - \frac{e}{2e_{init}}$

Probabilidad de expansión p_e**.** Es la probabilidad de que un agente entre en estado *expandir*. Se define como la inversa de la probabilidad de agrupamiento ($p_e = 1 - p_a$).

Las probabilidades de transición entre estados serán p_a y p_e y el estado inicial será *expandir*, a partir del cual cada agente calculará su estado en el momento t. Cada agente puede cambiar de estado únicamente cada periodo de tiempo t_c para permitir la ejecución de los comportamientos. Este periodo es constante e idéntico para todos los agentes.

Adicionalmente, cuando un agente se encuentra encima de una fuente de recursos, sería conveniente que redujera su velocidad para inspeccionar la fuente y para indicar su posición al resto del enjambre. Con este fin, se propone obtener un nuevo vector $\boldsymbol{v} = \boldsymbol{v}_b \times c^k$ donde $\boldsymbol{v}_b$ es el vector de movimiento obtenido del comportamiento s_i del robot y k es un modificador de la intensidad de atracción de la fuente.

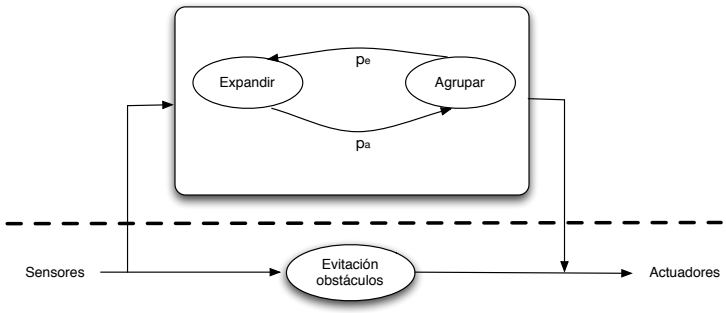


Figura 6.1: Arquitectura para un comportamiento de localización de recursos. Define una arquitectura de subsunción de dos capas. La capa inferior se compone de un comportamiento básico de evitación de obstáculos. La capa superior contiene una máquina de estados finita con dos posibles estados.

La arquitectura dispone de un nivel inferior con el comportamiento de evitación de obstáculos para evitar las colisiones con paredes, elementos del entorno u otros robots. La figura 6.1 muestra un esquema de esta arquitectura de subsunción de dos niveles.

Inicialmente los robots se sitúan formando una matriz de máximo 10 columnas y de un número indeterminado de filas (dependiendo del tamaño del enjambre), de manera que $pos_{init}(r_i) = \{i \mod 10, \lfloor \frac{i}{10} \rfloor\}$ y el compañero de cada agente se define como: $m_i = r_a, a = i + 1 \mod |R|$.

En este comportamiento se utilizan los agentes μ de la arquitectura, ya que el comportamiento global del grupo emerge automáticamente y los robots no necesitan ninguna información adicional.

6.2.1. Modelo macroscópico

Para poder determinar el comportamiento del enjambre a nivel global, se ha definido un modelo macroscópico que permitirá analizar la convergencia del enjambre en un punto.

Los comportamientos de *agrupar* y *expandir* están controlados por la probabilidad de agrupamiento p_a y de expansión p_e respectivamente. Por lo tanto, para determinar si los agentes del enjambre tienen una tendencia a determinado estado, se debe conocer la cantidad de agentes en cada periodo $\Phi = \left\lfloor \frac{t}{t_c} \right\rfloor$ que desarrollan cada uno de los comportamientos. Si asumimos que inicialmente los dos comportamientos tienen la misma probabilidad obtenemos:

$$
\begin{aligned}
&N_e(0) = \tfrac{1}{2}|R| \\
&N_e(\Phi) = p_e \cdot N_a(\Phi - 1) - p_a \cdot N_e(\Phi - 1) \\
&N_a(0) = \tfrac{1}{2}|R| \\
&N_a(\Phi) = p_a \cdot N_e(\Phi - 1) - p_e \cdot N_a(\Phi - 1)
\end{aligned}
$$

Donde $|R|$ es el número de robots del enjambre, N_a es el número de robots en estado *agrupar* en el ciclo Φ y N_e es el número de agentes en estado *expandir* en el ciclo Φ. Teniendo en cuenta el valor de las probabilidades p_a y p_e, calculamos la siguiente función generativa basándonos en las ecuaciones de recurrencia anteriores, que nos permite calcular el número de agentes que se encuentran en determinado estado.

$$
\begin{aligned}
N_e(\Phi) &= \tfrac{1}{2}\frac{|R|(1+\left(-\frac{-e_{init}+\Delta_e}{e_{init}}\right)^{\Phi}\Phi)}{\Phi+1} \\
N_a(\Phi) &= |R| - N_e(\Phi)
\end{aligned}
\tag{6.1}
$$

Utilizando esta ecuación podemos determinar cuándo todos los agentes tienden a desarrollar un comportamiento de agrupamiento y por tanto, como se muestra en [GM07], cuándo el enjambre estará agrupado en cierto punto. Esto ocurrirá cuando $N_e(\Phi)$ sea cercano a 0. Para el modelo de enjambre dado, podemos calcular cuándo el sistema convergerá en un punto, conociendo el número de agentes, la energía inicial de cada agente y su decremento por ciclo.

$$
e_{init} > \Delta_e, \Delta_e > 0, \lim_{\Phi\to\infty} \tfrac{1}{2}\frac{|R|(1 + \left(-\frac{-e_{init}+\Delta_e}{e_{init}}\right)^{\Phi}\Phi)}{\Phi + 1} = 0 \tag{6.2}
$$

6.3. Experimentación

Para comprobar el funcionamiento del comportamiento se han definido diferentes mapas agrupados en tres series de pruebas. En las primeras pruebas se ha utilizado un mapa infinito con ninguna fuente de recursos para comprobar el modelo macroscópico propuesto. El segundo grupo consiste en tres mapas para comprobar el comportamiento del sistema ante ruido en el entorno. Estos mapas se muestran en la figura 6.2. Se ha definido un mapa simple con una única fuente de recursos a) con un área de 100×100 metros donde la fuente de recursos ocupa un área de 30×30 metros, en b) se observa el mismo mapa con un 25 % de ruido Gaussiano y en c) con un 50 % de ruido.

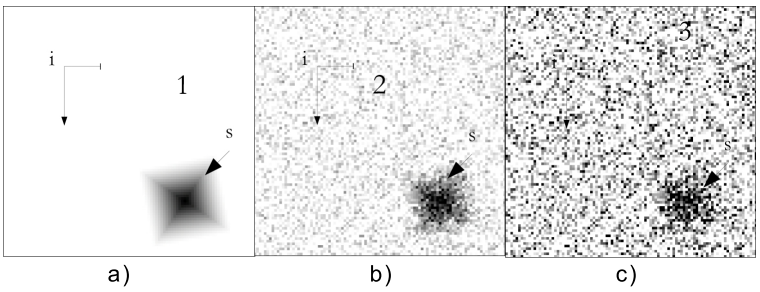


Figura 6.2: Mapas utilizados en el conjunto de pruebas para comprobar el funcionamiento del sistema ante ruido en el entorno. i indica la posición inicial de los robots, s indica la mayor fuente de recursos del entorno.

El tercer grupo consiste en seis mapas para comprobar el comportamiento del sistema ante diferentes fuentes de recursos en el entorno. Estos mapas se pueden observar en la figura 6.3. En la parte superior se observan mapas con varias fuentes de la misma validez. En la parte inferior los mapas contienen múltiples fuentes con diferentes grados de validez.

También se han probado mapas más complejos como el mostrado en la figura 6.4 para mostrar el comportamiento del sistema ante entornos complejos con diferentes fuentes de recurso.

Los entornos definidos pueden contener celdas de recurso, donde cada celda tiene el mismo tamaño que un agente. La cantidad de recurso en una celda se representa de manera progresiva de blanco (ningún recurso) a negro (máxima cantidad de recurso).

Las primeras pruebas, donde se ha utilizado un mapa sin ninguna fuente de recursos, tienen como objetivo comprobar que el modelo macroscópico propuesto predice correctamente el comportamiento del enjambre.

Para determinar si la predicción realizada por el modelo macroscópico coincide con las simulaciones realizadas con el enjambre se han llevado a cabo cinco simulaciones para un mapa sin recursos, utilizando un entorno infinito y los siguientes parámetros $(|R|, t_c, e_{init}, \Delta_e) = (100, 200, 1000, 10)$.

En la figura 6.5 se pueden observar los resultados obtenidos con el modelo macroscópico y con las simulaciones llevadas a cabo. En este gráfico se muestra el número de agentes que se encuentran en estado *expandir* obtenido con el modelo macroscópico (línea azul) y los resultados obtenidos a partir de las simulaciones (zona roja). Se observa cómo ambos resultados coinciden, mostrándose que el número de agentes en estado *expandir* tiende a cero, y por tanto, más agentes se encuentran en estado *agrupar* conforme

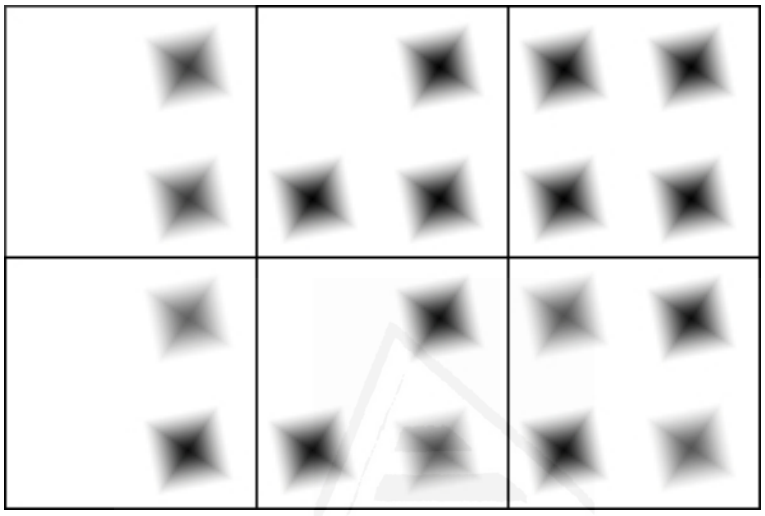


Figura 6.3: Parte superior: Mapas con diferentes fuentes de recursos igual de prometedoras. Parte inferior: Mapas con fuentes de recursos con diferente grado de validez.

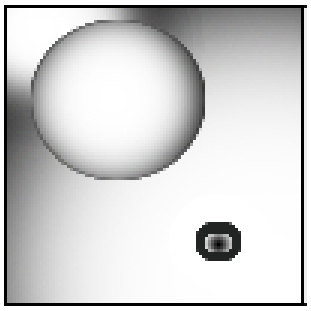


Figura 6.4: Mapa complejo con diferentes áreas de recursos.

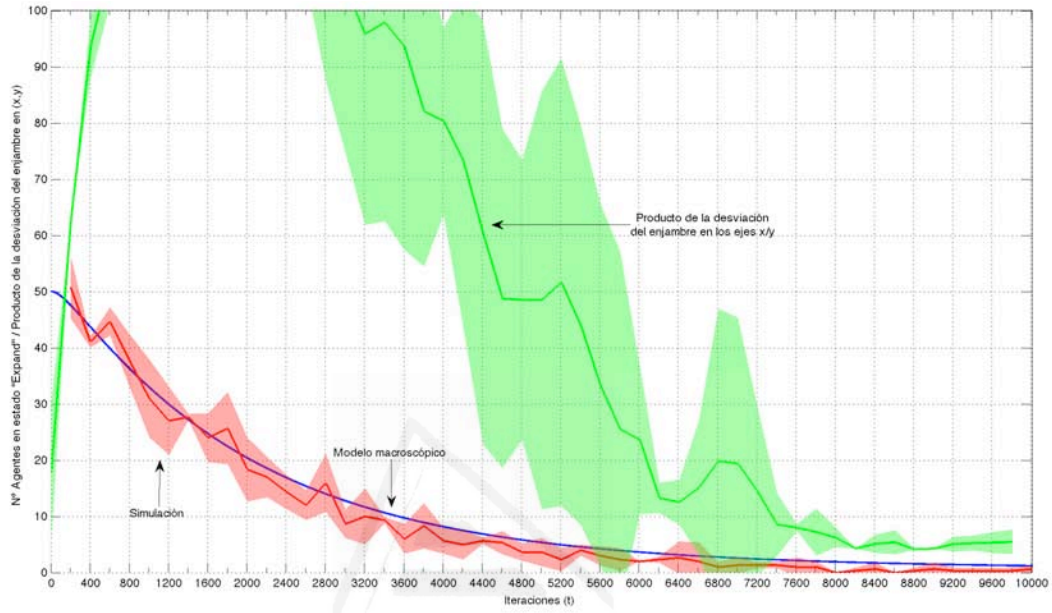


Figura 6.5: Resultados de los experimentos con un mapa sin recursos. Muestra el resultado del modelo macroscópico (línea azul) y de la simulación, donde se muestra la media (línea roja) y la varianza (zona roja).

se incrementa t. Además, la zona verde del gráfico, muestra la desviación del enjambre. Vemos como ésta tiende a cero con el número de iteraciones. Estas dos tendencias nos indican que el enjambre se va agrupando con el tiempo.

El segundo conjunto de pruebas comprueba la robustez del sistema ante ruido en el entorno. Se han llevado a cabo cinco simulaciones para cada mapa mostrado en la figura 6.2 (mapa sin ruido, con 25 % de ruido Gaussiano y con 50 % de ruido). Se han utilizado los siguiente parámetros $(|R|, t_c, e_{init}, \Delta_e) = (100, 200, 1000, 10)$ para un mapa de tamaño 100×100 metros.

La figura 6.6 muestra el resultado de estas simulaciones. Esta figura muestra la distancia a la fuente de recursos para cada una de las simulaciones y la desviación del enjambre. Se observa cómo la distancia a la fuente de recursos disminuye con el tiempo, a su vez, la desviación del enjambre tiende a cero con el número de iteraciones para todas las simulaciones.

Estos resultados demuestran que el enjambre es robusto ante ruido en el entorno. Vemos cómo para el mapa con un 25 % de ruido la convergencia es la misma que para un mapa sin ruido. Para el mapa con un 50 % de ruido

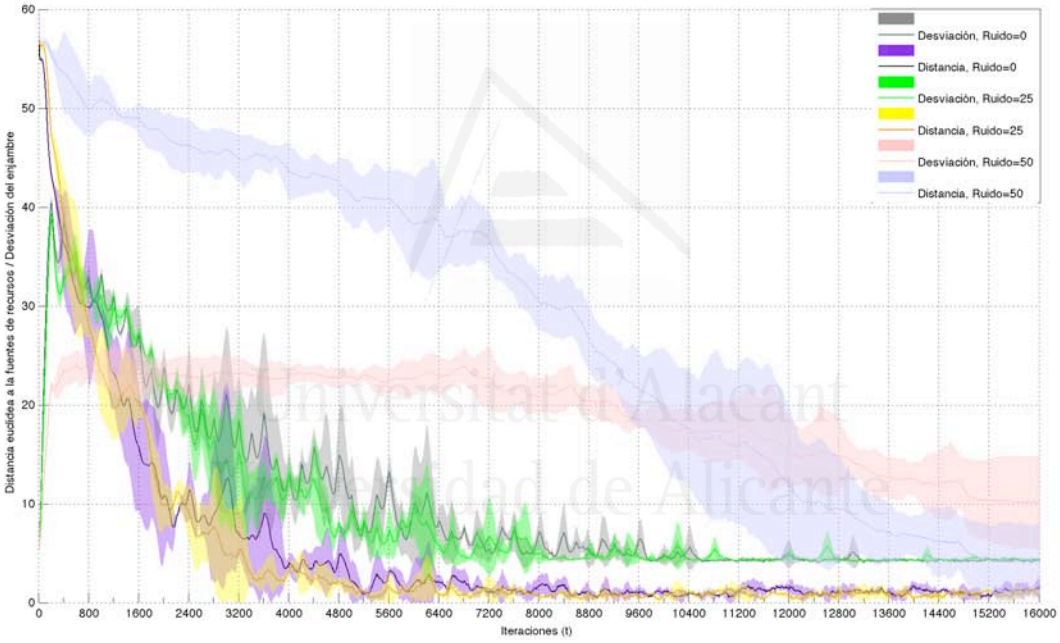


Figura 6.6: Desviación del enjambre y distancia a la fuente de recursos más prometedora con diferentes niveles de ruido Gaussiano.

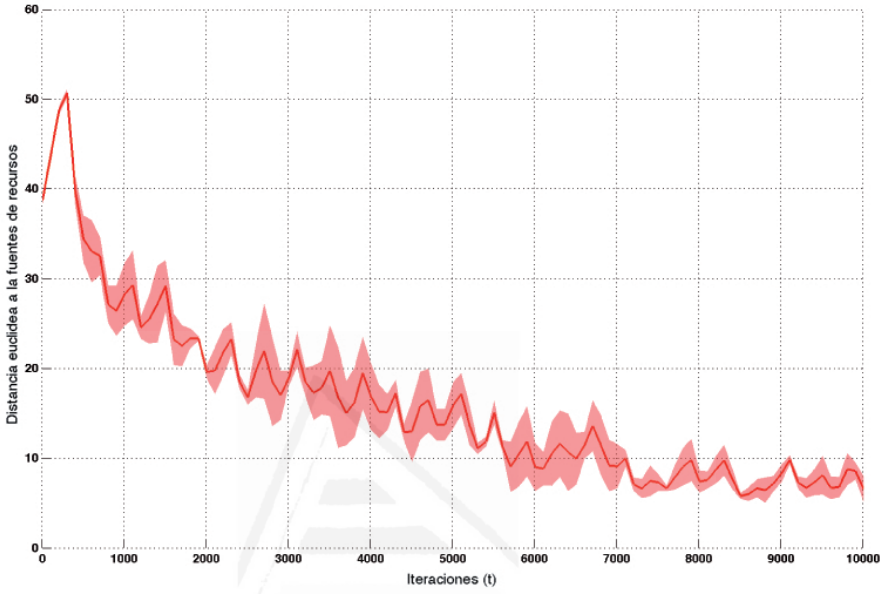


Figura 6.7: Distancia y desviación al área de recursos más cercana en un mapa con tres áreas de recurso igual de prometedoras.

la convergencia es más lenta pero constante.

El tercer grupo de pruebas comprueba el funcionamiento del sistema en entornos con diferentes fuentes de recursos. Para este fin se han utilizado los mapas mostrados en la figura 6.3. Se han realizado 5 simulaciones para cada uno de los mapas con los siguientes parámetros $(|R|, t_c, e_{init}, \Delta_e) = (100, 200, 1000, 10)$ para un mapa de tamaño 100×100 metros.

La figura 6.7 muestra la distancia del enjambre al área de recursos más cercana en un entorno con tres fuentes de recurso de la misma validez. Se observa cómo esta distancia tiende a cero, y por tanto, el enjambre está situado sobre las fuentes de recurso.

La figura 6.8 muestra el comportamiento del enjambre ante entornos con diferentes fuentes de recursos. En a) se observa cómo el enjambre se divide en tres grupos cuando hay tres fuentes de recursos igual de válidas, sin embargo, en b) se observa cómo el enjambre únicamente se divide en las dos zonas de recursos más prometedoras descartando el resto de fuentes.

Por último, para comprobar el funcionamiento del sistema en entornos más complejos y la tendencia del enjambre a la fuente más prome-

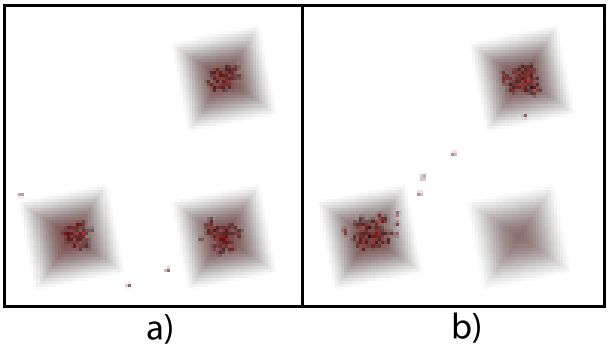


Figura 6.8: Imagen de *MASON* con la situación del enjambre en entornos con diferentes fuentes de recursos. a) Muestra cómo el enjambre se divide cuando hay fuentes de recursos igual de válidas. b) Muestra cómo el enjambre converge únicamente sobre las fuentes de recursos más prometedoras, descartando el resto.

tedora, teniendo en cuenta la importancia del tamaño del enjambre, se han llevado a cabo cinco simulaciones para diferentes tamaños de enjambre $|R| = \{10, 25, 50, 100\}$ con el mapa mostrado en la figura 6.4. El tamaño del entorno se ha limitado a 100×100 metros y se han utilizado los parámetros especificados anteriormente.

La figura 6.9 muestra la distancia a la fuente óptima para cada conjunto de simulaciones con los diferentes tamaños de enjambre. Se observa cómo esta distancia se va reduciendo con el tiempo, tendiendo a cero, para todas las simulaciones, es decir, el enjambre está situado sobre esta fuente.

6.4. Resumen

En este capítulo se ha definido un comportamiento microscópico que define las capacidades de cada agente individual para llevar a cabo una tarea de localización de recursos, y un modelo macroscópico capaz de analizar la convergencia del enjambre.

Para estudiar estos dos modelos se han realizado tres conjuntos de pruebas. En las primeras pruebas se ha utilizado un entorno sin recursos, para comprobar que el modelo macroscópico es capaz de predecir correctamente la convergencia del enjambre. El siguiente conjunto de pruebas está formado por tres entornos con una fuente de recursos y diferentes niveles de ruido Gaussiano. Estas pruebas se han llevado a cabo para comprobar la robustez del comportamiento ante ruido en el entorno. Por último, el tercer conjun-

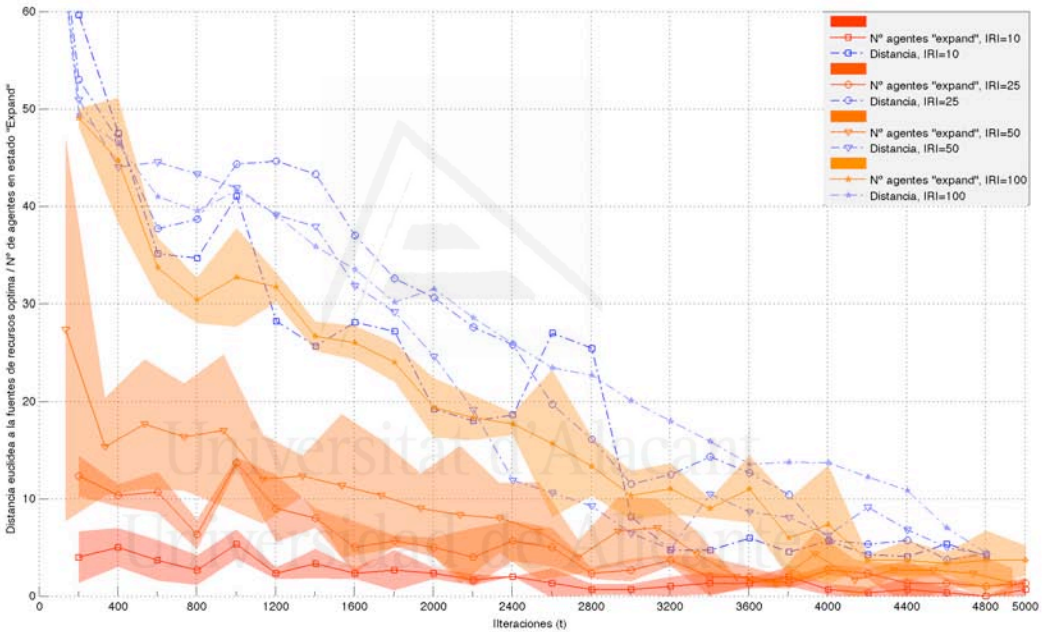


Figura 6.9: Distancia del enjambre al área de recursos más válida, junto con la cantidad de agentes en estado *expandir* para cada tamaño del enjambre $|R| = \{10, 25, 50, 100\}$. Se representa la media de la distancia para cada uno de los tamaños. Para el número de agentes en estado *expandir* se representa la media (línea oscura) y la varianza (zona clara).

to de pruebas ha permitido analizar el comportamiento del enjambre ante entornos con diferentes fuentes de recursos.

Las primeras pruebas demuestran que el modelo macroscópico predice correctamente el número de agentes que se encuentran en un estado determinado, coincidiendo con los resultados obtenidos en las simulaciones. El comportamiento de agregación de los agentes que llevan a cabo el comportamiento *agrupar* estabiliza el sistema en un tiempo limitado, por lo tanto, como el sistema propuesto tiende a que todos los agentes lleven a cabo un comportamiento de *agrupar*, podemos indicar que el sistema tenderá a agruparse cuando el número de agentes en estado $N_e(\Phi) \approx 0$.

El segundo conjunto de pruebas ha demostrado que el enjambre es robusto ante ruido en el entorno. Para un mapa con 25 % de ruido la convergencia del enjambre hacia una fuente de recursos es similar a la de un mapa sin ruido y con un ruido del 50 % la convergencia es más lenta pero constante.

Además, en el tercer conjunto de pruebas también se ha estudiado el comportamiento del enjambre en entornos con diferentes fuentes de recursos. La figura 6.7 muestra cómo la distancia a una de las áreas de recursos más prometedoras tiende a 0, lo que indica que el enjambre tiende a situarse sobre una de estas fuentes más prometedoras. Si en el entorno se encuentran fuentes igual de válidas, el enjambre se dividirá en diferentes grupos y convergerá sobre estas áreas más prometedoras, descartando el resto de recursos.

En todas las pruebas realizadas se obtiene una clara tendencia hacia la agregación sobre las fuentes de recursos.

En este comportamiento se utiliza únicamente la capa de enjambre, es decir, los agentes μ del modelo de arquitectura definido, ya que el comportamiento global del grupo emerge a partir de la interacción de los agentes con el entorno.

En el siguiente capítulo se definirá otro comportamiento específico para un enjambre robótico. Concretamente, un comportamiento para la detección y seguimiento de un vertido de petróleo.

Capítulo 7

Sistema de enjambre para detección de vertidos

Como se ha visto en capítulos anteriores la robótica de enjambre ha acumulado, en los últimos años, una serie de aplicaciones derivadas de sus beneficios de uso (robustez, paralelismo o flexibilidad) y sus características distribuidas. Entre las múltiples aplicaciones de este tipo de sistemas es posible encontrar la exploración de entornos desestructurados, la monitorización de un recurso o su uso como sistemas sensoriales distribuidos móviles. Dos de estas aplicaciones, la monitorización y la detección de un área o perímetro de un recurso dado, tienen varios usos ecológicos. Entre ellos se encuentra la detección y monitorización de contaminantes y otros elementos para delimitar su perímetro y área de manera precisa, como por ejemplo, vertidos de petróleo, nubes tóxicas, bancos de algas o incendios forestales.

Uno de los lugares donde esta detección tiene una relevancia especial es en los mares u océanos. La actividad marítima se ha visto incrementada gradualmente en los últimos años. Muchos buques transportan productos que pueden afectar negativamente el medio ambiente, como petróleo, que puede producir grandes niveles de contaminación en caso de ser vertido al mar.

En este capítulo se presenta un comportamiento de enjambre capaz de monitorizar, cubrir y marcar el perímetro de un recurso utilizando un enjambre homogéneo. Esté comportamiento ha sido pensado para ser llevado a cabo por un enjambre de drones de bajo coste. Estos drones utilizan únicamente su información sensorial y no requieren ninguna comunicación directa con otros agentes del sistema.

El sistema propuesto se basa en el modelo GNOME desarrollado por

National Oceanic & Atmospheric Administration (NOAA) de Estados Unidos que permite modelar un vertido de petróleo. En este modelo es posible utilizar mapas meteorológicos reales para simular las corrientes oceánicas y los vientos. Esto nos permitirá realizar una simulación realista de un vertido de petróleo en las costas españolas.

Para este fin se define un modelo microscópico para un enjambre teniendo en cuenta las propiedades de los vertidos de petróleo. Este sistema permite la monitorización de las manchas de crudo producidas en un vertido marítimo. Posteriormente se define un modelo macroscópico para analizar el funcionamiento del enjambre a nivel global. A continuación, se presentarán los resultados analíticos y experimentales que muestran el funcionamiento del sistema.

7.1. Introducción

La actividad marítima ha crecido de manera gradual en los últimos años. Según la Dirección General de Pesca y Asuntos Marítimos de la Comisión Europea unos 200.000 barcos cruzan el mar Mediterráneo cada año. Además, cerca de 42.000 barcos (excluyendo los pesqueros) cruzan anualmente el Mar del Norte, llevando en sus bodegas productos que pueden afectar el medio ambiente. Hemos podido ver algunos accidentes dramáticos que han conllevado una gran contaminación marítima, como el vertido de petróleo ocasionado por el Prestige en el año 2002 y que afectó a la costa norte española, o el accidente de una plataforma petrolífera en el año 2010 en el golfo de México que vertió más de 779.000 toneladas de crudo al mar. Con el fin de intentar mitigar futuros daños a recursos naturales valiosos causados por accidentes marítimos, la comunidad científica ha llevado a cabo diversas investigaciones para estudiar los procesos que afectan el destino y la distribución de los contaminantes, intentado además, modelar y simular estos procesos. Los sistemas activos, que son capaces de detectar y monitorizar este tipo de vertidos son una herramienta invalorable para ayudar a localizar y limpiar las zonas afectadas.

En [CF07] presentan un sistema cooperativo y descentralizado donde cada agente se compone de una máquina de estados finita y dispone de capacidades de comunicación con un rango limitado. Por lo tanto, cuando un robot localiza el perímetro de un recurso, realiza un *broadcast* de la localización a todos los robots dentro de su rango de comunicación. Cada uno de éstos a su vez realiza la misma operación. Este comportamiento definido únicamente tiene en cuenta perímetros continuos.

En [KBH08] presentan un sistema para un equipo de UAVs (*Unmanned Aerial Vehicles*) donde cada agente es capaz de intercambiar información con sus vecinos para enviar esta información a la estación base. Esta estación base es capaz de detectar la distancia a la que está el agente (y por tanto la longitud del perímetro) por la latencia percibida. En este comportamiento reducen el problema a un perímetro lineal.

En [CKBM06] presentan un sistema formado por UAVs que permite recopilar información de manera colectiva en incendios forestales. Cada agente también envía la información a una estación base, que se encarga de generar, a partir de la información recibida por los diferentes agentes del sistema, un estado general del incendio.

A continuación se presenta un comportamiento definido para un enjambre con el objetivo de cubrir o marcar el perímetro de un vertido petrolífero. En este comportamiento se utilizará GNOME: un sistema de simulación medioambiental realista para contaminaciones marítimas. Con la información obtenida de este simulador, y utilizando información marítima real y predicciones de corrientes de aire, se creará un vertido de petróleo virtual en las costas españolas. Se mostrará el comportamiento del enjambre para detectar y monitorizar el vertido en un periodo de tiempo. Para esto se definirá un modelo microscópico del cual se derivará un modelo macroscópico para predecir el comportamiento global del enjambre.

7.2. Modelo de dispersión de contaminantes

Modelar un vertido de crudo no es una tarea sencilla, principalmente debido a la cantidad de factores que influencian las trayectorias de los contaminantes: las corrientes marítimas, los vientos e incluso la fuerza gravitacional o la tensión superficial del agua. Por todos estos factores, en algunas ocasiones nos preguntamos si la trayectoria calculada por un modelo es precisa, adecuada o correcta. Estas características están asociadas con los datos utilizados para realizar los cálculos y los procesos físicos modelados.

Existen varias aplicaciones para modelar y simular vertidos de contaminantes en el mar. Estas aplicaciones se basan en el modelado conjunto de las fuerzas más importantes que interactúan sobre una mancha de crudo, para poder obtener datos lo más realistas posibles del vertido.

En este trabajo se ha utilizado GNOME, un sistema de simulación medioambiental interactivo para modelar las trayectorias de un vertido en un entorno marítimo. GNOME simula el movimiento del crudo causado por los vientos, las corrientes y las mareas. Este simulador ha sido desarrollado por

Hazardous Materials Response Division (HAZMAT) del *National Oceanic and Atmospheric Administration Office of Response and Restoration (NOAA OR&R)*. Como se comenta en [Woj03], GNOME puede ser utilizado para predecir cómo los vientos, las corrientes y otros procesos pueden mover y dispersar la mancha de crudo en el agua. Este simulador estudia cómo las trayectorias calculadas pueden verse afectadas por la incertidumbre que llevan asociadas las observaciones meteorológicas y las previsiones de viento y corrientes, y son capaces de predecir los cambios físicos y químicos que se pueden producir en el crudo durante el tiempo que permanece en la superficie del mar.

Uno de los beneficios de esta herramienta es la posibilidad de utilizar datos externos con información relativa a mareas y corrientes. Si estos datos son añadidos a la aplicación, GNOME utiliza esta información para realizar las predicciones de la trayectoria del vertido (incluso utilizando incertidumbre durante la simulación). La salida obtenida consiste en gráficos, vídeos y ficheros de datos que se pueden post-procesar en un sistema de información geográfica (SIG) o en *NOAA Emergency Response Division's (ERD1)* de GNOME.

El modelo utilizado en esta herramienta es general y es aplicado a problemas de trayectorias. Es un modelo de Euler/Lagrange de dos dimensiones (2D). Los mapas de la costa se pueden utilizar como entradas del modelo, de esta manera, es posible modelar cualquier área.

De manera más específica, la propagación aleatoria se calcula por un comportamiento de movimiento aleatorio gobernado por una probabilidad. El movimiento aleatorio está basado en un valor de difusión D, que representa las turbulencias horizontales de la difusión en el agua. Durante un vertido, el valor es calibrado basándose en la información del modelo. De esta manera, la difusión y la propagación son tratadas como procesos estocásticos. Los efectos producidos por las tensiones gravitacionales y superficiales son ignorados ya que éstos únicamente son importantes en los primeros momentos del vertido.

En [oRR12] se presenta la ecuación de difusión principal utilizada en GNOME, donde D_x y D_y son los coeficientes de difusión escalares en las direcciones x e y, y C es la concentración de contaminante:

$$\frac{\partial C}{\partial t} = D_x \cdot \frac{\partial^2 C}{\partial x^2} + D_y \cdot \frac{\partial^2 C}{\partial y^2}$$

Como se ha comentado, GNOME simula esta difusión con un movimiento aleatorio, obteniendo como coeficiente de difusión:

$$D_x = \frac{1}{2} \cdot \frac{\sigma^2}{\Delta t}$$

donde σ^2 es la varianza de la distribución de los puntos propagados y Δt es el tiempo transcurrido entre pasos.

GNOME modela la evaporación mediante un algoritmo de evaporación simple de tres fases donde el contaminante es tratado como una sustancia de tres componentes con periodo de desintegración independiente [BFMP82]:

$$X_{prob} = \frac{P_1 \cdot \left(2^{\frac{-t_i}{H_1}} - 2^{\frac{t_{i-1}-2t_i}{H_1}}\right) + P_2 \cdot \left(2^{\frac{-t_i}{H_2}} - 2^{\frac{t_{i-1}-2t_i}{H_2}}\right) + P_3 \cdot \left(2^{\frac{-t_i}{H_3}} - 2^{\frac{t_{i-1}-2t_i}{H_3}}\right)}{P_1 \cdot 2^{\frac{-t_i}{H_1}} + P_2 \cdot 2^{\frac{-t_i}{H_2}} + P_3 \cdot 2^{\frac{-t_i}{H_3}}}$$

Donde t_i y $t_i - 1$ es el tiempo transcurrido (en horas) en el paso i y en el paso previo $i - 1$, respectivamente, desde el lanzamiento de los LE (Langrarian elements). H_1, H_2, H_3 son los periodos de desintegración de cada componente del contaminante (en horas); y P_1, P_2, P_3 son los porcentajes de cada componente del contaminante.

Las sustancias vertidas son modeladas como masas de puntos (hasta 10,000[4]) llamados LEs (*Lagrangian elements*) o "*splots*" (derivado de "*spill dots*"). Los vertidos pueden ser modelados de diferentes maneras: como un único vertido o como varios lanzamientos continuos; como puntos o líneas de recursos; o incluso situarlos en una cuadrícula en un mapa con fines de diagnóstico.

Una vez que GNOME ejecuta una simulación, la solución se produce en forma de trayectoria. GNOME proporciona dos soluciones para un escenario: la trayectoria estimada más exacta; o una trayectoria con incertidumbre. La primera solución muestra el resultado del modelo asumiendo todos los datos de entrada como correctos. La solución con incertidumbre permite al modelo predecir otras posibles trayectorias que pueden ser menos probables, pero que pueden tener asociado un riesgo mayor. En este comportamiento utilizaremos la solución con incertidumbre para las partículas de contaminante (representadas por sus LEs) para generar un mapa de contaminante continuo. Además, para generar las simulaciones se han utilizado tanto datos de corrientes marítimas como datos de viento reales.

Es posible encontrar más detalles de este modelo matemático en [oRR12].

7.3. Definición del comportamiento

Como se ha comentado anteriormente, el modelado de este tipo de contaminantes es complejo ya que interactúan multitud de factores. A continuación, se extraerán un conjunto de características que debe cumplir un sistema para localizar este tipo de vertidos.

Por una parte, el enjambre ha de ser capaz de detectar y seguir los elementos contaminantes, que pueden desplazarse en el tiempo, principalmente por la advección y difusión. Dependiendo de la aplicación se puede asumir que el origen del vertido es conocido o suponer que será el mismo enjambre el que deberá realizar inicialmente el proceso de detección (en esta experimentación se asume que la detección forma parte de las tareas a desarrollar). Por otra parte es muy probable que surjan varias manchas de crudo de diferentes tamaños debido, entre otros factores, a la dispersión y evaporación.

El comportamiento de enjambre debe ser altamente robusto a fallos y debe llevar a cabo su comportamiento de manera totalmente distribuida. Además, el comportamiento debe ser altamente escalable, permitiendo la agregación y eliminación de nuevos agentes en el sistema, tanto por razones de robustez como por el desarrollo del mismo comportamiento, ya que en una primera fase es posible que sea beneficioso utilizar un número reducido de agentes hasta encontrar realmente evidencias de un determinado vertido. El objetivo principal de este comportamiento es determinar la capacidad del enjambre para localizar, converger y seguir un vertido petrolífero.

Antes de definir el comportamiento se debe tener en cuenta que éste es diseñado para agentes basados directamente en robots voladores. Estos drones dispondrán de una cámara de vídeo que utilizará un algoritmo de visión encargado de determinar si existen manchas de petróleo en alguna posición del espacio visible. Por razones de seguridad asumimos que los drones vuelan en distintas alturas, por lo que no se producirán colisiones en una misma posición (x, y). Asumimos también que debido a la altura de vuelo (unos 500m sobre el nivel del mar) las diferencias que se pueden producir en el campo de visión derivadas de las diferencias en altura de los agentes son despreciables.

7.3.1. Modelado microscópico

En este apartado se propone un comportamiento microscópico homogéneo, que ejecutarán todos los agentes del enjambre. Este comportamiento consta de tres estados: *Deambular*, *Recurso* y *EnRecurso*. A grandes rasgos, inicialmente los agentes buscarán por el entorno intentando encontrar

algún rastro del vertido. Una vez detectado el vertido el agente se dirigirá hacia él. Por último intentará mantenerse o bien dentro de él (para cubrir la mancha) o bien en su perímetro (para marcarla) dependiendo del comportamiento deseado.

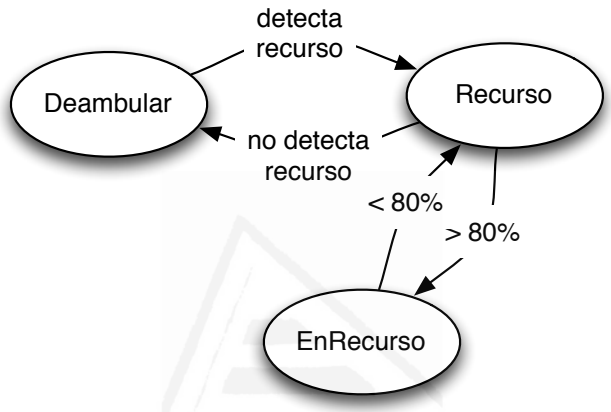


Figura 7.1: Máquina de estados finita que gobierna el funcionamiento de cada agente del enjambre. Un agente comienza en el estado *Deambular*. La transición al estado *Recurso* ocurre cuando el sensor visual del agente detecta una mancha de petróleo. Desde este estado la transición a *EnRecurso* se da cuando la cantidad de petróleo detectada es $> 80\,\%$ de la imagen. Cuando la cantidad de petróleo es $\leq 80\,\%$ de la imagen vuelve al estado *Recurso*. Por último, si el sensor del agente no detecta petróleo vuelve al estado *Deambular*.

En la figura 7.1 se muestra la máquina de estados finita que gobierna el comportamiento de cada robot. El estado inicial es *Deambular*, ya que inicialmente asumimos que los agentes desconocen la posición inicial del vertido. La transición del estado *Deambular* al estado *Recurso* se realizará cuando el sensor visual del agente detecte una mancha de petróleo. Si estando en el estado *Recurso* se deja de detectar dicho recurso el enjambre pasará de nuevo al estado *Deambular* donde seguirá buscando recurso. Si estando en estado *Recurso* la cantidad de petróleo detectada es mayor que el $80\,\%$ de la imagen, el agente pasaría al estado *EnRecurso*. El agente se mantendrá en este estado siempre que el porcentaje de petróleo detectado sea mayor que el $80\,\%$, en caso contrario volvería al estado *Recurso*.

A continuación se describe con más detalle el funcionamiento de cada

uno de los estados especificados.

Estado Deambular

Al inicio de la ejecución cada agente comienza en este estado. En ese momento los agentes no tienen conocimiento de la posición del vertido, por lo tanto, se mueven al azar por el entorno para intentar detectar un recurso.

La velocidad en el instante t queda definida tal que:

$$\boldsymbol{v}_w(t) = \boldsymbol{v}_w(t-1) + \boldsymbol{rand} \cdot \mu_1$$

siendo $\boldsymbol{rand}$ un vector aleatorio uniforme definido dentro de los intervalos de velocidad mínimos y máximos del agente y μ_1 el coeficiente de variabilidad sobre la velocidad actual. Con valores cercanos a 1 el robot se desplazará de manera totalmente aleatoria.

Estado Recurso

Una vez el agente ha detectado el recurso se dirige hacia a él para situarse en su perímetro o encima, dependiendo del comportamiento deseado. La velocidad que gobernará al agente queda definida a partir de tres factores:

$$\boldsymbol{v} = \alpha_1 \times \boldsymbol{v}_c + \alpha_2 \times \boldsymbol{v}_o + \alpha_3 \times \boldsymbol{v}_r$$

donde $\alpha_1 + \alpha_2 + \alpha_3 = 1$ y definen la intensidad de cada término. $\boldsymbol{v}_c$ especifica la dirección a tomar por el robot, que estará determinada por la zona con el mayor promedio de intensidad de recurso:

$$\boldsymbol{v}_c = \gamma(S) \times \left\| \sum_{s \in S} ((\boldsymbol{pos}(s) - \boldsymbol{pos}(rob)) \cdot s) \right\|$$

siendo S el conjunto de las lecturas que detecta el sensor de recurso en un momento dado, $pos(s)$ el vector posición de una lectura dada y $\boldsymbol{pos}(rob)$ la posición actual del robot. Se asume que la intensidad de recurso detectada se encuentra en el intervalo $[0, 1]$, siendo 0 la falta completa de recurso y 1 la detección inequívoca del mismo. γ determina el sentido del vector velocidad, dependiendo de si el robot se encuentra fuera o dentro del recurso. Su finalidad es por tanto mantener a los robots en el perímetro del recurso a detectar:

$$\gamma(S) = \begin{cases} 1 & \frac{\sum_{s \in S} s}{|S|} \leq \eta \\ -1 & \frac{\sum_{s \in S} s}{|S|} > \eta \end{cases}$$

Donde η es un umbral que determina a partir de la cantidad de recurso detectado (0 significa que no se ha detectado recurso y 1 que todo lo detectado es recurso) si el agente se encuentra en el perímetro. Si lo que se desea es que los agentes cubran el entorno contaminado, en lugar de marcar el perímetro, entonces $\gamma(S)$ se definirá como 1 para cualquier conjunto de lecturas S.

$\boldsymbol{v}_o$ especifica un vector de evitación respecto a todos los robots detectados en un momento dado:

$$\boldsymbol{v_o} = \left\| \sum_{i=1}^{|R|} (\boldsymbol{pos}(r_i) - \boldsymbol{pos}(rob)) \right\|$$

donde R es el conjunto de robots detectados, $\boldsymbol{pos}(r_i)$ la posición del robot detectado i y $\boldsymbol{pos}(rob)$ la posición del robot actual.

Por otra parte, para tener en cuenta la precisión de las localizaciones transmitidas, se incluye un componente aleatorio para modelar esta incertidumbre en el movimiento del robot: $\boldsymbol{v_r}(t) = \boldsymbol{v}_r(t-1) + \boldsymbol{rand} \cdot \mu_2$, donde μ_2 es el coeficiente de variabilidad sobre la velocidad.

Estado EnRecurso

Por último, cuando los agentes se encuentran dentro del vertido y por tanto no perciben su límite, asumimos que desarrollarán una conducta de desplazamiento aleatorio hasta encontrarse de nuevo con el agua:

$$\boldsymbol{v}_s(t) = \boldsymbol{v}_s(t-1) + \boldsymbol{rand} \cdot \mu_3$$

donde μ_3 es el coeficiente de variabilidad sobre la velocidad.

7.3.2. Modelado macroscópico

Una vez establecido el comportamiento microscópico de los agentes es interesante analizar el comportamiento global del enjambre. Existen varias técnicas para analizar dicho comportamiento, como el uso de ecuaciones de recurrencia generadas a partir de un comportamiento microscópico definido mediante una máquina de estados finita o la definición de ecuaciones diferenciales. No obstante, la mayoría de estos métodos únicamente permiten analizar a nivel global la evolución de las transiciones entre estados.

La plataforma propuesta en [HW08] permite obtener la distribución de probabilidad de la posición del enjambre para cualquier instante t. Esto nos permitirá predecir, detalladamente, el comportamiento global del sistema.

Tal y como se define en [HW08], una vez definido el comportamiento microscópico, es posible calcular el comportamiento global del sistema utilizando la ecuación de Fokker-Planck:

$$\frac{\partial \rho(\boldsymbol{r},t)}{\partial t} = -\nabla\left(\boldsymbol{A}(\boldsymbol{r},t)\rho(\boldsymbol{r},t)\right) + \frac{1}{2}Q\nabla^2\left(B^2(\boldsymbol{r},t)\rho(\boldsymbol{r},t)\right) \qquad (7.1)$$

donde Q es el desplazamiento producido por una colisión. $\rho\left(\boldsymbol{r},t\right)dr_x dr_y$ es la probabilidad de encontrar un robot en la posición r dentro del rectángulo definido por dr_x y dr_y en el instante t. r se utiliza para posiciones de puntos del espacio que no tienen por qué estar directamente conectados con un robot explícito.

Esta ecuación proporciona un método para modelar estadísticamente un enjambre de robots, basándose en técnicas de modelado de sistemas multipartícula provenientes del campo de la física cuántica. A partir de una ecuación de Langevin, que representa el comportamiento de una única partícula, se deriva la ecuación de Fokker-Planck para todo el conjunto.

Como se describe en [Ham10b], la ecuación de Fokker-Planck implementa las abstracciones necesarias respecto a detalles microscópicos y trata los parámetros que cambian rápidamente como ruido. La ecuación continúa siendo correcta si este ruido es generado por un proceso Gaussiano, es decir, si está totalmente determinado por los primeros dos momentos. Tiene como resultado la evolución temporal de la densidad de probabilidad de las posiciones de los agentes.

Inicialmente, para diseñar el enjambre se deben especificar las funciones $\boldsymbol{A}$ y B de la ecuación 7.1, conforme al comportamiento microscópico deseado. La función $\boldsymbol{A}$ es una dirección y describe el movimiento determinístico basándose en la información proporcionada por el entorno y la información proporcionada indirectamente por otros robots a través del entorno. La función B describe el componente aleatorio del movimiento. $\boldsymbol{A}$ y B se caracterizan por el algoritmo de control subyacente. B normalmente incorpora las influencias de otros robots que alteran el movimiento del agente, por ejemplo, por la necesidad de realizar un comportamiento de evitación de obstáculos. $\boldsymbol{A}$ debe incorporar una influencia externa, como un gradiente suave.

Definir estas funciones es la parte más complicada de estas ecuaciones. Este diseño no es único y requiere encontrar dos funciones que describan correctamente el comportamiento del enjambre. A continuación se presentarán de manera analítica dichas funciones.

Función *A*

Esta función determina el desplazamiento del enjambre. $\boldsymbol{A}$ depende en primer lugar de un vector que representa la información direccional. Para definirla es habitual utilizar un campo de potencial P. En nuestro caso, basándonos en el modelo microscópico propuesto, necesitamos establecer una función que tenga en cuenta los estados de movimiento aleatorio de los robots, las probabilidades de que un movimiento no se realice correctamente por parte de un agente (por ejemplo debido a una colisión) y el campo de potencial sobre el que navegan los robots. Aunque se podría modelar una distribución de probabilidad para cada estado, dado que nuestro modelo microscópico no tiene interacción entre los agentes (excepto la meramente física, como las colisiones) y que el comportamiento de los estados es relativamente simple, es posible agrupar el comportamiento macroscópico del enjambre en una única distribución. En la ecuación 7.2 se propone una función $\boldsymbol{A}$ que tiene en cuenta estos aspectos.

$$\boldsymbol{A}(\boldsymbol{r},t) = (1-b_r)\cdot(1-\rho(\boldsymbol{r},t))^{\mu_4}\cdot(\Gamma(P(\boldsymbol{r},t))\cdot K\left(\nabla P(\boldsymbol{r},t)\right)) \tag{7.2}$$

donde P está directamente relacionada con las lecturas de los sensores en la posición r en el instante t y μ_5 es un término de normalización:

$$P(\boldsymbol{r},t) = \mu_5 \cdot s(\boldsymbol{r},t)$$

Cuando la densidad de los agentes aumenta, se incrementa la probabilidad de colisión y por tanto se reduce la tasa en la cual los robots se dirigen con el vector especificado por $\boldsymbol{A}$. De esta manera proponemos el término $1-\rho(\boldsymbol{r},t)$, que penaliza el movimiento de las zonas con mucha cantidad de robots. μ_4 permite ajustar la intensidad de dicha penalización. K obtiene una versión promediada del campo de potencial mediante la aplicación de un operador de convolución (en este caso, un filtro Gaussiano con $\sigma = 4|S|$). Esto nos permite introducir un componente que tiene en cuenta la naturaleza probabilística del sistema, uniformizando la elección entre zonas cercanas con un potencial alto. Además necesitamos obtener a partir del campo de potencial proveniente de los recursos detectados que dirección tomará un robot. Γ es una función que se aplica al campo de potencial definido por el sensor de recurso y que permite obtener para cada posición de P el sentido a tomar por los agentes, tal y como lo haría un agente individual utilizando la función γ[1]. $(1-b_r)$ se trata de un término normalizador.

[1] En nuestro caso hemos utilizado un filtro "sliding-neighborhood" para realizar el mismo

Función B

La función B describe el movimiento no determinístico y por tanto tiene en cuenta el movimiento aleatorio de los agentes. En nuestro comportamiento microscópico intervienen principalmente dos fuerzas a tener en cuenta en este término. Por una parte tenemos influencias derivadas de los agentes que se encuentren en los estados *Deambular* y *EnRecurso*. Estos estados se mueven de manera aleatoria, dependiendo de la intensidad de los parámetros μ_1 y μ_3. Por otra parte, el propio comportamiento provocará que el espacio tenga zonas de mayor densidad de agentes. En estas zonas aumentará la probabilidad de colisión dependiendo de la densidad de agentes en un momento dado.

$$B(\boldsymbol{r}, t) = b_r \cdot \rho(\boldsymbol{r}, t)^{\mu_4} \tag{7.3}$$

De esta manera, en la ecuación 7.3 se observan dos términos. b_r engloba las influencias de aquellos estados donde los agentes desarrollan un movimiento aleatorio o pseudo-aleatorio. Y $\rho(\boldsymbol{r}, t)^{\mu_4}$ se trata de un término que permite definir la relación de la densidad de robots respecto a su probabilidad de colisión.

7.4. Experimentación

Para realizar la experimentación se ha realizado una simulación de un vertido de crudo en la zona costera de "Sant Antoni de Portmany", en la isla de Ibiza, utilizando un área de aproximadamente $350km^2$. Se ha elegido esta área debido a las diversas corrientes que afectan a la zona y por su cercanía a la península. Se han utilizado datos meteorológicos reales, tanto para las corrientes de agua como para las de aire, tomando como fecha de inicio del vertido el día 10 de abril de 2013 a las 0:00h. La simulación se prolonga hasta siete días después desde el inicio del vertido, simulando un barco cargado con 100.000 barriles de petróleo. En la imagen 7.3 se observa el mapa de la zona principal afectada y el lugar inicial del vertido.

Para trabajar con los datos generados con GNOME se han promediado los puntos de vertido generados por la aplicación (*splots* en GNOME), uniformizando de esta manera la continuidad entre zonas contaminadas cercanas. Además, esto permite simplificar la simulación reduciendo el número

cálculo sobre P que en el modelo microscópico, cambiando el signo del desplazamiento al alcanzar una proporción superior al 80 % de contaminante detectado

$$\begin{cases} Robots & = 200 \\ \mu_1 & = 0{,}3 \\ \mu_2 & = 0{,}3 \\ \mu_3 & = 0{,}3 \\ \eta & = 0{,}8 \\ \alpha_1 & = 0{,}6 \\ \alpha_2 & = 0{,}2 \\ \alpha_3 & = 0{,}2 \end{cases}$$

Figura 7.2: Parámetros por defecto a utilizar en las simulaciones. Estos parámetros han sido ajustados mediante experimentación a partir de las definiciones del comportamiento microscópico base.

de *splots* necesarios: $M(t)' = G\left(M(t) \oplus s_{shape}\right)$, siendo $M(t)$ el mapa obtenido en GNOME en el instante t, donde G es un filtro Gaussiano con $\sigma = 10|S|$, y $\oplus$ es un operador morfológico binario.

En 7.2 se muestran los parámetros microscópicos utilizados para realizar las simulaciones del sistema:

7.4.1. Pruebas del modelo microscópico

A continuación se presentan varias pruebas para verificar el funcionamiento del modelo a nivel local. En las primeras pruebas se analizará la convergencia del enjambre para una única mancha estática. Sobre este mismo ejemplo se modificará γ para que el enjambre cubra la mancha en lugar de marcar su perímetro. En las siguientes pruebas se verificará la convergencia del enjambre pero en este caso seleccionando un instante en el cual existan varios focos de vertido activos. En las últimas pruebas se verificará el seguimiento de la mancha en movimiento.

Para la simulación se ha utilizado un enjambre de 200 agentes distribuidos inicialmente de manera aleatoria en el entorno. Estos agentes circularán de manera uniforme a 60km/h. Se simularán drones de reducido tamaño ($< 3m^2$), capaces de detectar de manera visual un área de $1km^2$.

Detección de una mancha de vertido

Las primeras pruebas realizadas consisten en la detección de una mancha estática. En la parte izquierda de la figura 7.3 se presenta un mapa de la posición del vertido y la distribución inicial de los agentes. En la parte

derecha de esta misma figura se muestra la posición de los agentes en el instante $t = 15{,}000s$ y $t = 30{,}000s$, arriba y abajo, respectivamente. En la imagen se puede observar como en el instante 30,000s el enjambre rodea totalmente la mancha de petróleo.

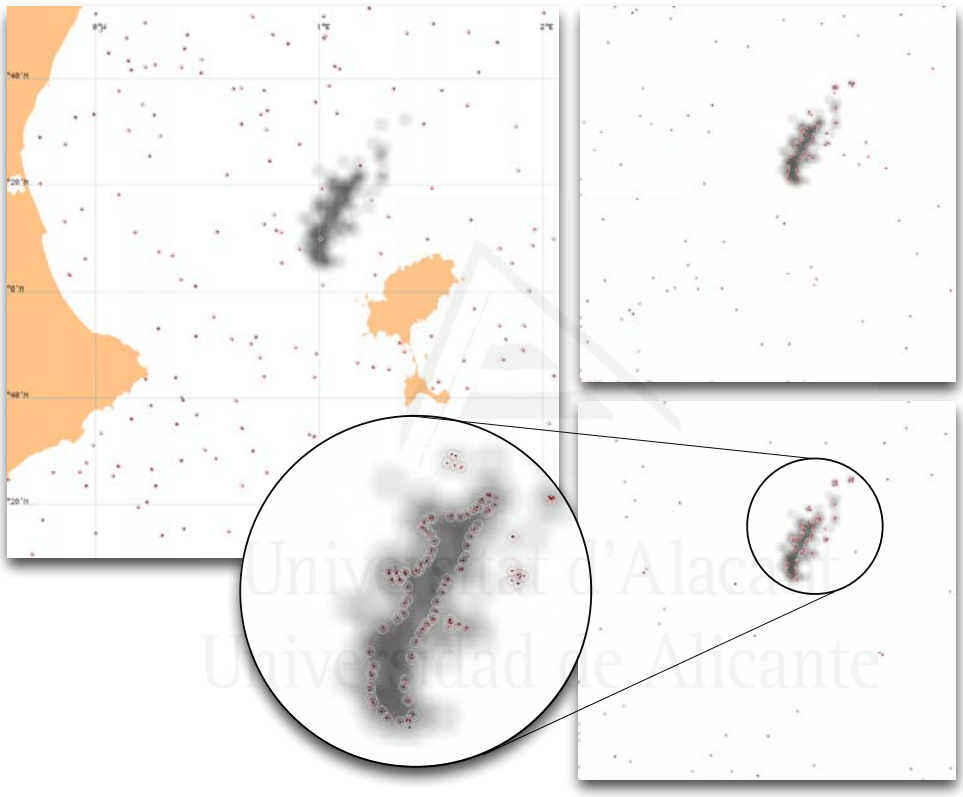


Figura 7.3: Distribución de los agentes con respecto al tiempo (en segundos) sobre un vertido de petróleo para la tarea de detectar su perímetro. Izquierda: situación geográfica de la mancha y posición inicial de los agentes. Derecha: posición de los agentes en el instante $t = 15{,}000s$. Abajo: posición de los agentes en el instante $t = 30{,}000s$

La figura 7.4 muestra el porcentaje de agentes sobre una mancha de petróleo respecto al tiempo para 5 ejecuciones independientes. Como se puede observar, el número de agentes que están situados sobre la mancha aumenta de manera progresiva con el tiempo.

Por otra parte, como se comentó al definir el modelo microscópico del enjambre, en caso de que $\gamma(S) = 1$ el enjambre cubrirá la mancha en lugar

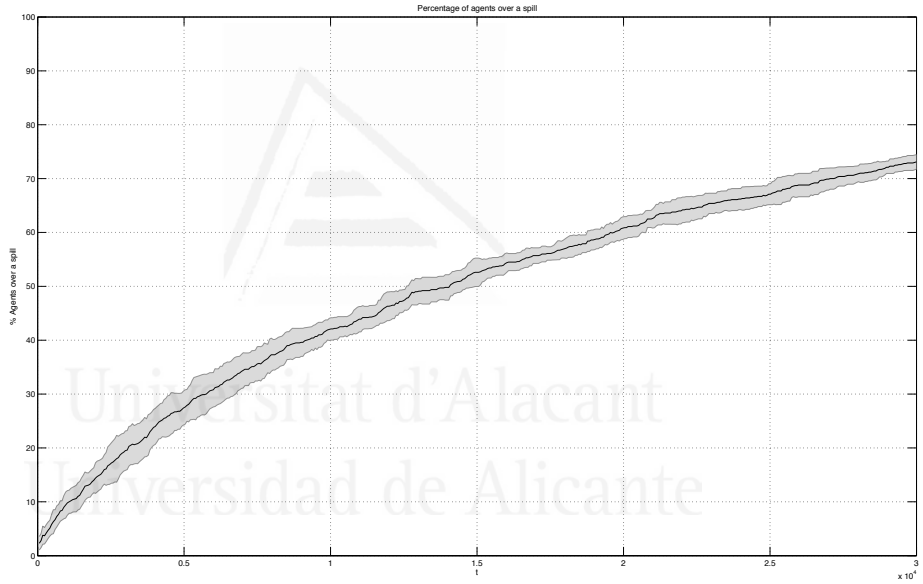


Figura 7.4: Porcentaje de agentes sobre una mancha de petróleo respecto al tiempo (en segundos). Se representan 5 simulaciones diferentes, mostrando su media y varianza

de definir su perímetro. En la 7.5 se observa la evolución del enjambre para este caso. Como se muestra en la imagen, mientras que en el experimento anterior los agentes se situaban en el perímetro de la mancha, en este caso los agentes están situados sobre la mancha.

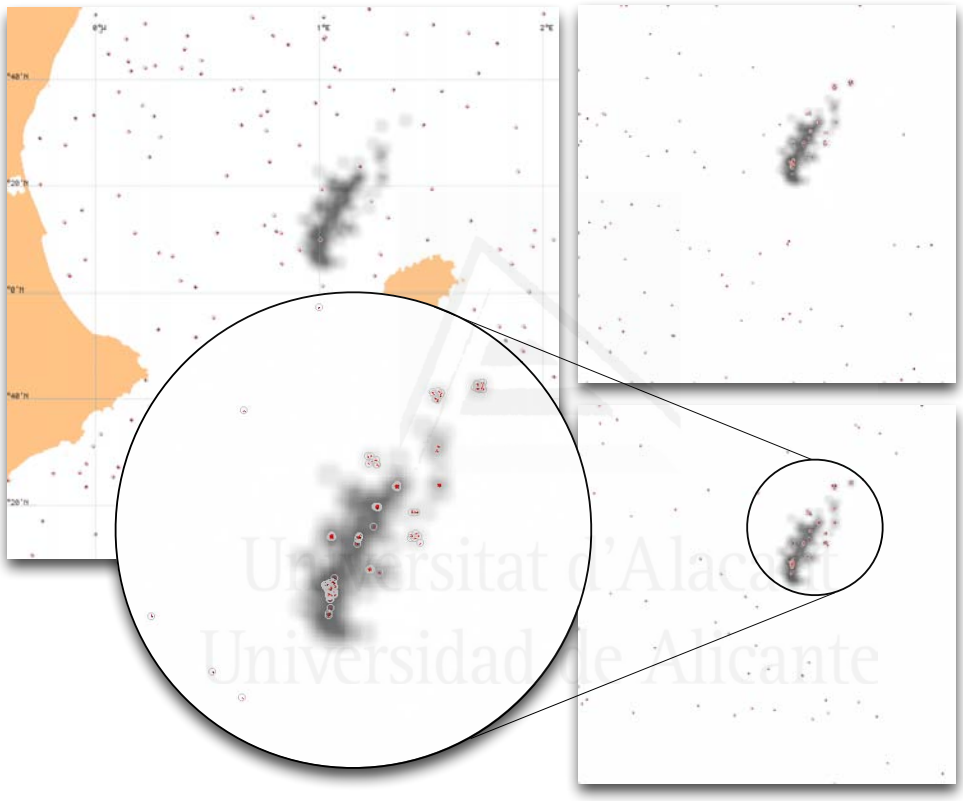


Figura 7.5: Distribución de los agentes con respecto a un vertido de petróleo para la tarea de cubrir dicho vertido ($\gamma(S) = 1$). Izquierda: situación geográfica de la mancha y posición inicial de los agentes. Derecha: posición de los agentes en el instante $t = 15{,}000s$. Abajo: posición de los agentes en el instante $t = 30{,}000s$

Detección de varias manchas de vertido

En un vertido petrolífero es muy común que se produzcan varias manchas de crudo. Por ejemplo cuando las corrientes submarinas acercan rápidamente a la costa el contaminante, mientras el navío causante del vertido continúa

vertiendo petróleo al mar. Es por ello que es de gran importancia verificar el funcionamiento del enjambre en situaciones donde se producen varias zonas de vertido distantes entre sí. En la simulación realizada, ya en las primeras horas se producen varias manchas que contienen elevadas proporciones de contaminante.

Para la realización de estas pruebas se ha seleccionado uno de los instantes de la simulación con una estructura de mancha compleja para comprobar el funcionamiento del modelo microscópico propuesto. En la figura 7.6 se muestra la evolución de la distribución de los robots. Como se observa en la figura, aún en este caso complejo el modelo es capaz de localizar y marcar el perímetro.

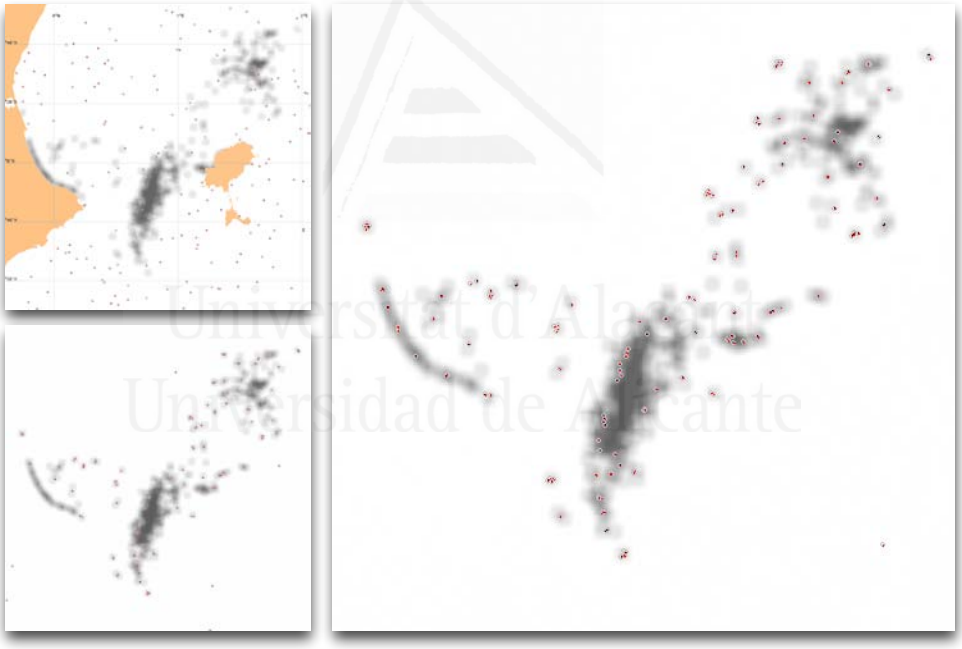


Figura 7.6: Distribución de los agentes con respecto a un vertido de petróleo para la tarea de detectar su perímetro en un mapa con varias manchas de contaminante. Izquierda arriba: situación geográfica de la mancha y posición inicial de los agentes. Izquierda abajo: posición de los agentes en el instante $t = 15{,}000s$. Derecha: posición de los agentes en el instante $t = 30{,}000s$

Más concretamente, en la figura 7.7 se observa la evolución del porcentaje de agentes que están situados encima de una zona contaminada. Se puede observar cómo aumenta la cantidad de agentes que están situados en algu-

na mancha de crudo conforme pasa el tiempo, distribuyéndose por ésta de manera uniforme.

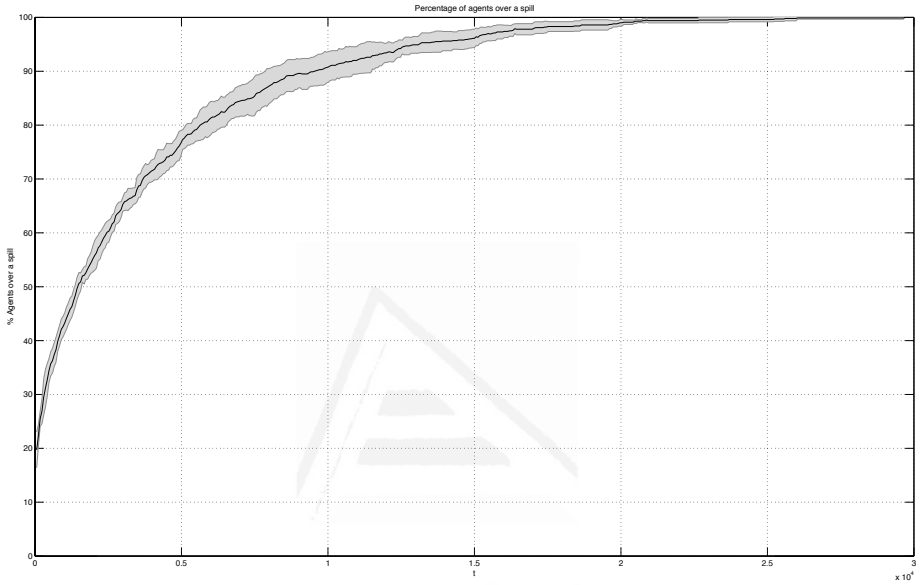


Figura 7.7: Porcentaje de agentes sobre manchas de petróleo respecto al tiempo (en segundos) para cubrir una mancha compleja. Se muestran 5 simulaciones diferentes, mostrando su media y varianza.

Detección de una secuencia del vertido

En los experimentos anteriores se ha mostrado cómo el enjambre es capaz de detectar y marcar el perímetro/área de manchas complejas. No obstante, en un vertido real es tan importante la localización del vertido como un seguimiento eficaz. En este apartado se analiza el comportamiento del enjambre teniendo en cuenta la evolución del vertido en el tiempo.

Utilizando GNOME se ha realizado una simulación de 168 horas sobre la evolución del vertido, comenzando el 10 de abril de 2013 a las 0:00h. Para simplificar la simulación, se ha optado por capturar una instantánea del estado de la mancha cada 4,5 horas. Aunque durante este tiempo la mancha habrá sufrido cambios estos serán asumibles por el enjambre. En la figura 7.8 se muestra el punto de origen del vertido y algunas instantáneas de su

evolución temporal.

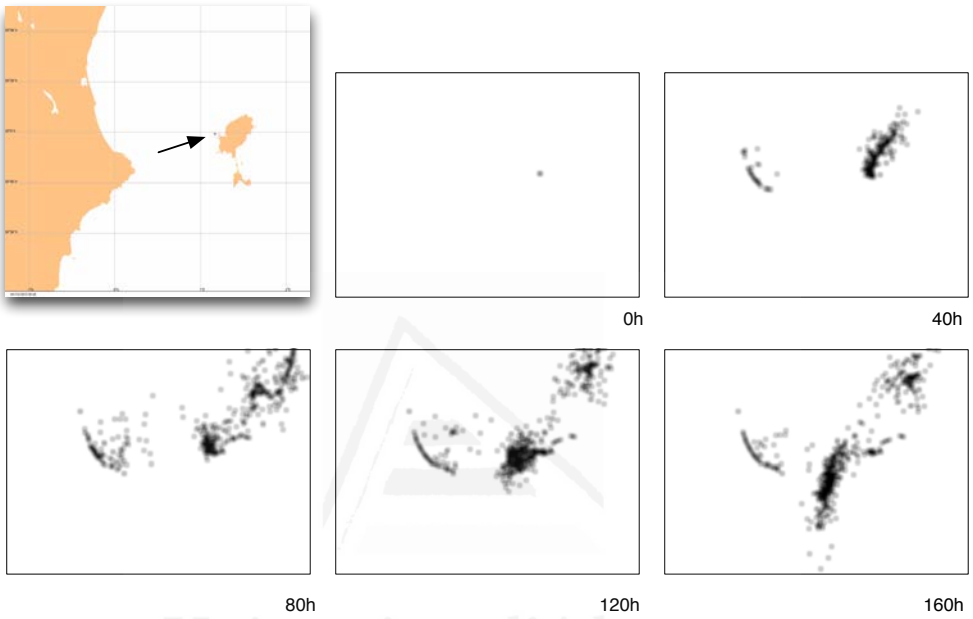


Figura 7.8: Origen del vertido y evolución temporal. Se muestran varias capturas que indican la posición del vertido en varios instantes de tiempo, medidos en horas desde el origen del vertido.

Al igual que en los experimentos anteriores se ha distribuido el enjambre de manera aleatoria por el entorno. Se ha continuado el proceso de simulación con la salvedad de que cada 4,5 horas se actualiza la posición del recurso obtenida mediante GNOME. En la figura 7.9 se muestra la evolución del enjambre respecto al estado de la mancha de crudo. Se observa cómo los agentes se van situando en el perímetro de las diferentes manchas.

La figura 7.10 muestra el porcentaje de agentes que están situados sobre un recurso en un momento dado. En este gráfico se muestran los resultados de 5 simulaciones diferentes, promediando y mostrando las varianza de todas ellas para cada instante de tiempo. Como se observa en el gráfico, aparecen una serie de escalones producidos artificialmente por el simulador al cambiar de un estado de la mancha al siguiente de manera repentina. Aún así se puede observar cómo la evolución del enjambre es correcta, incrementándose en el tiempo el número de agentes que están situados sobre la mancha.

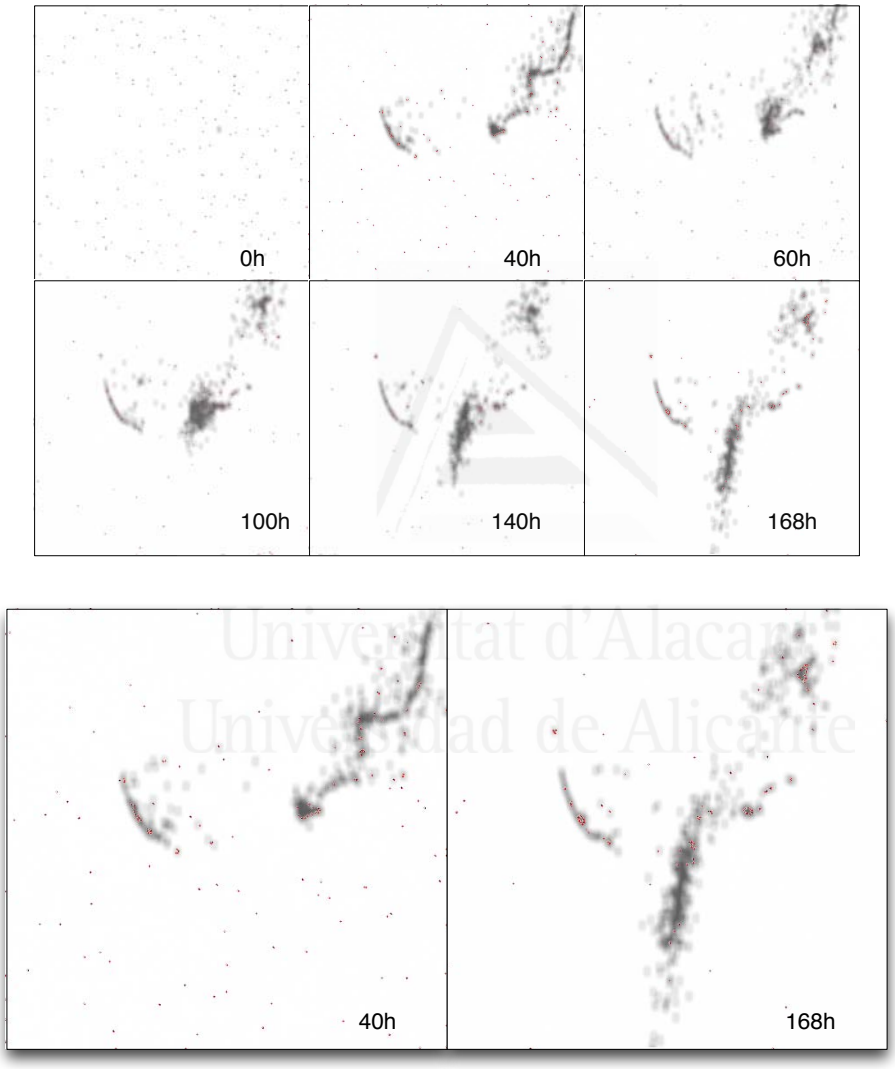


Figura 7.9: Evolución del enjambre respecto a la mancha de crudo. Se presentan varias capturas en distintos instantes de tiempo, medidos en horas desde el inicio del vertido.

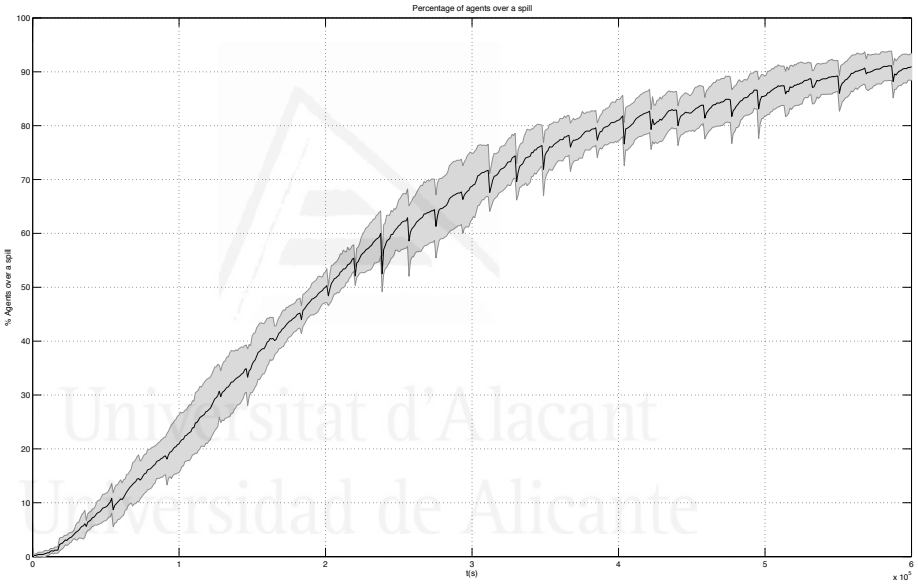


Figura 7.10: Porcentaje de agentes sobre una mancha de petróleo respecto al tiempo (en segundos). El vertido y su evolución temporal se han obtenido utilizado la herramienta de simulación GNOME. Se muestran 5 simulaciones diferentes, mostrando su media y varianza.

$$\begin{array}{ll} Robots & = 200 \\ \mu_4 & = 5 \\ \mu_5 & = 0{,}95 \\ br & = 0{,}5 \end{array}$$

Figura 7.11: Parámetros por defecto utilizados en las simulaciones para el modelo macroscópico. Estos parámetros han sido ajustados a partir de los experimentos realizados.

7.4.2. Pruebas del modelo macroscópico

En el punto anterior se ha mostrado el funcionamiento del comportamiento microscópico definido. Se ha visto cómo con reglas relativamente simples el enjambre es capaz de localizar y marcar el perímetro de un vertido petrolífero. Se han realizado diversas pruebas para verificar la validez del modelo presentado en diversos casos.

Además de las pruebas anteriores, utilizando un modelo macroscópico, es posible establecer para un determinado vertido, la zonas del mapa con mayor probabilidad de contener un robot, independientemente del número de agentes a utilizar (siempre y cuando se utilice un número suficiente de ellos). Esto permite visualizar el comportamiento de manera más exacta para enjambres de gran tamaño sin estar limitados por el número de agentes a simular.

En esta sección, utilizando la definición macroscópica presentada previamente, se comprobará cómo el enjambre localiza y marca en un determinado instante una mancha de crudo. En la figura 7.11 se muestran los parámetros por defecto a utilizar para las pruebas de este modelo. Presentaremos el análisis del comportamiento macroscópico del enjambre para dos instantes de tiempo $t = (140h, 168h)$.

El proceso de simulación, una vez definidas las funciones $\boldsymbol{A}$ y B de la ecuación de Fokker-Planck es sencillo. Inicialmente se debe establecer un punto de origen del enjambre. Aunque en las simulaciones microscópicas es posible establecer distintos puntos de origen (como establecer de manera aleatoria la posición de los agentes), la ecuación Fokker-Planck requiere que se establezca un único punto. Se han realizado diversas simulaciones con distintos puntos de origen, observando cómo el resultado para un valor de t elevado es muy parecido; sólo se observan ligeras variaciones si el punto de inicio se sitúa encima de una mancha de crudo, aumentando en este caso la probabilidad de esta mancha con respecto al resto que puedan existir en el

entorno.

En las pruebas aquí mostradas se ha utilizado el mismo punto de origen, discretizando el área de simulación en 100×100 unidades. De esta manera, se establece el origen en la ecuación de Fokker-Planck como $(x, y) = (40, 25)$.

Una vez definida la ecuación de Fokker-Planck, podemos obtener de manera iterativa la distribución de probabilidad de que un determinado agente se encuentre en una posición del entorno en un momento dado. Esto se consigue iterando para cada instante de tiempo t todas las posiciones del mapa actualizando para cada una de ellas la nueva probabilidad descrita por la ecuación.

En la figura 7.12 se presentan los resultados del estado macroscópico del enjambre para el instante del vertido $t = 140h$. Se observa una clara convergencia en las zonas de perímetro de la mancha. En la figura 7.13 se presenta de manera tridimensional la distribución de probabilidad de que un agente esté en una posición determinada del entorno en el instante del vertido $t = 168h$.

Como se puede observar el modelo macroscópico predice correctamente el funcionamiento observado a nivel microscópico del enjambre.

Verificación de los modelos

En los dos apartados anteriores se ha mostrado el funcionamiento del modelo tanto a nivel microscópico (local) como a nivel macroscópico (global). A continuación se va a comparar las predicciones del modelo microscópico con el modelo macroscópico para un caso concreto.

En la figura *a)* de 7.14 se muestra una distribución de probabilidad obtenida a partir del modelo microscópico. Para ello se ha discretizado el entorno de simulación y con los mismos parámetros utilizados en secciones anteriores se ha desarrollado la simulación de 200 agentes para el instante de vertido $t = 140h$. Se ha ido almacenando a lo largo de la simulación cuántos agentes pasaban por cada celda discreta (para hacerlo comparable con el modelo macroscópico se ha discretizado también el entorno en 100×100 celdas) y calculado posteriormente la probabilidad de que un agente se encuentre en dicha celda en un momento dado.

De la misma manera se ha procedido a obtener la distribución de probabilidad que predice el modelo macroscópico obteniendo la imagen mostrada en la figura *b)* de 7.14. Ya a simple vista se observa cómo el área de interés es cubierta por ambos modelos. Existen pequeñas diferencias relativas a que la simulación microscópica tiene deficiencias, ya que entre otras cosas, depende directamente del número de agentes utilizados en la simulación. Sin

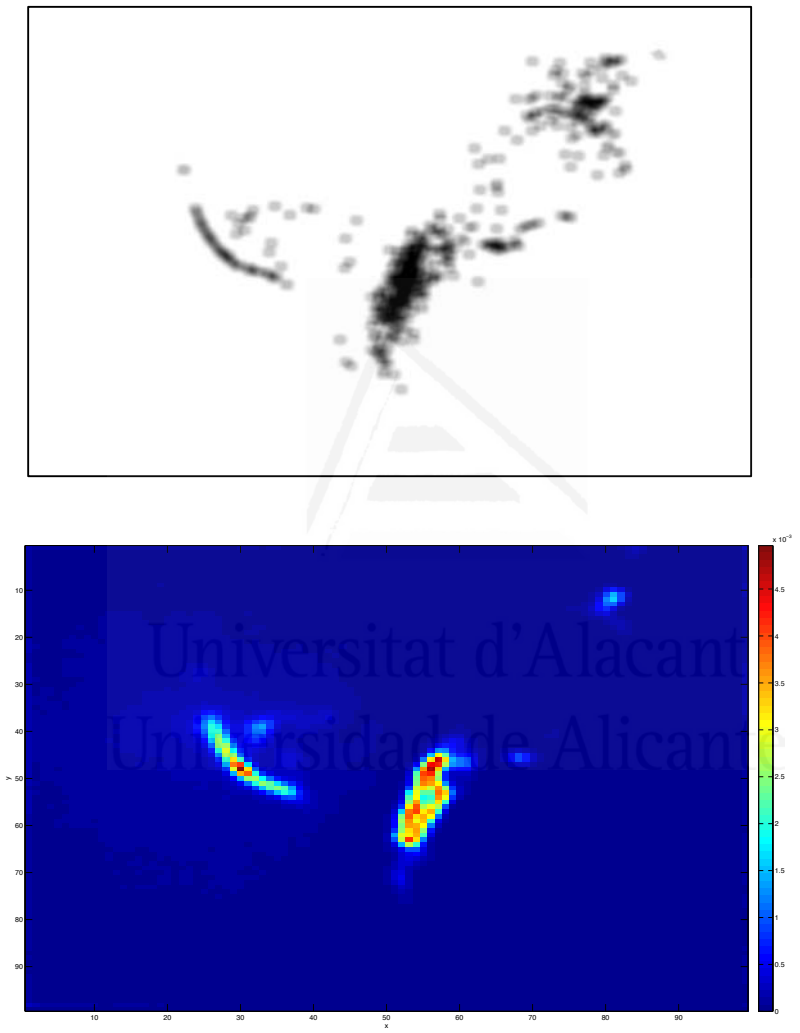


Figura 7.12: Arriba: mapa (M) generado a partir de los datos de GNOME para $t = 140h$. Abajo: probabilidad de que un robot se encuentre en una determinada posición del espacio con $t = 140h$. Muestreo utilizando el modelo macroscópico del enjambre, mediante la ecuación Fokker-Planck.

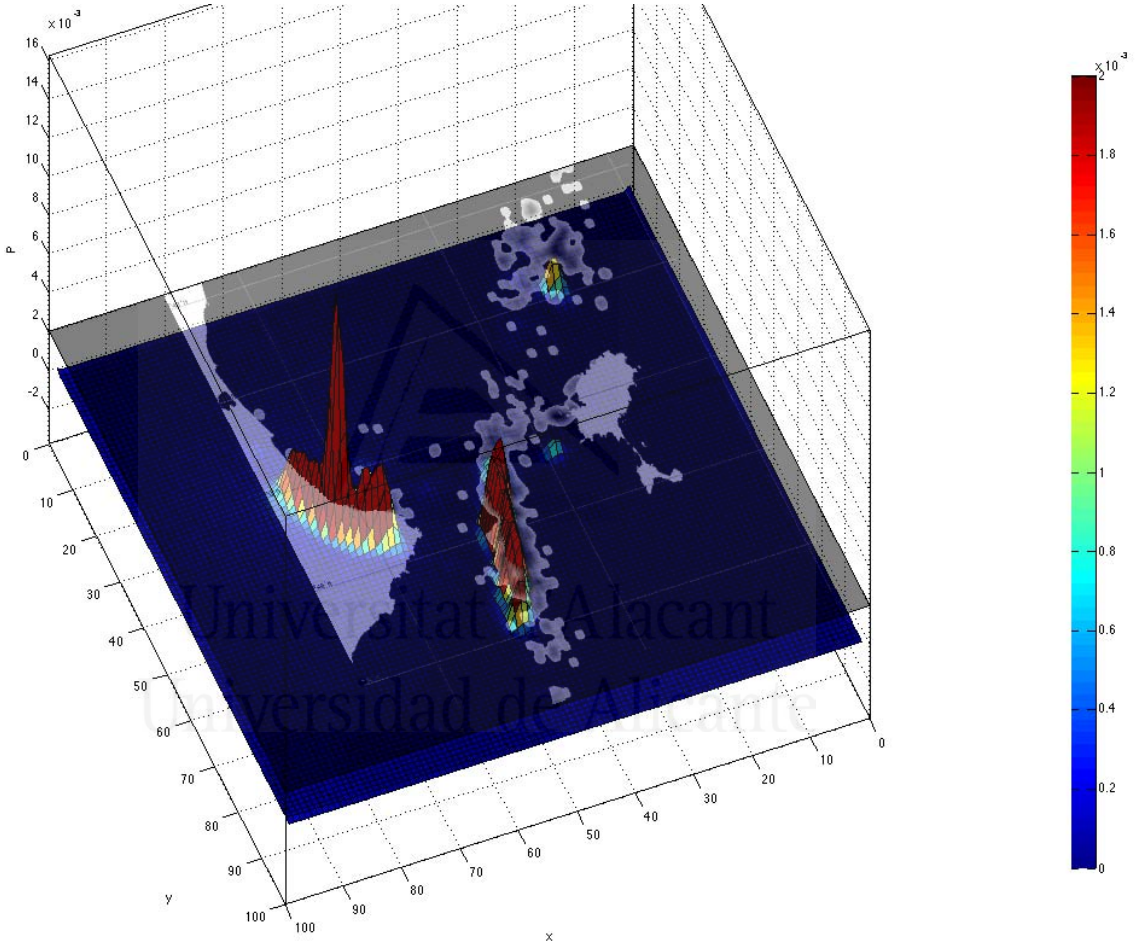


Figura 7.13: Represetación tridimensional de la probabilidad de que un robot se encuentre en una determinada posición del espacio con $t = 168h$. Muestreo utilizando el modelo macroscópico del enjambre, mediante la ecuación Fokker-Planck. Se añade un plano para la probabilidad 0.002 donde se representa el mapa del entorno y el estado de la mancha en $t = 168h$

embargo, podemos intentar comparar ambas aproximaciones multiplicando ambas distribuciones, de manera que sólo las probabilidades más elevadas permanezcan. De esta manera es posible observar si las zonas del vertido quedan identificadas correctamente en ambos modelos. Para ello se ha rectificado levemente la distribución microscópica, ya que al realizar la simulación utilizando un número limitado de agentes nos encontramos con que las probabilidades más altas muchas veces ocultan información importante en la distribución. Es por ello que se ha utilizado para la comparación la raíz cuadrada de la distribución.

El producto de la distribución macroscópica junto con la raíz de la distribución microscópica se presenta en la figura *c)* de 7.14. En ella se observa cómo las partes fundamentales del vertido se detectan en ambas distribuciones, prediciendo ambas, de manera general, el mismo comportamiento.

7.5. Resumen

En este capítulo se ha presentado un modelo microscópico capaz de localizar, marcar y seguir un vertido de petróleo, y un modelo macroscópico capaz de predecir el comportamiento del enjambre en su totalidad y de verificar su funcionamiento.

El comportamiento microscópico definido permite la detección y seguimiento de manchas de petróleo ocasionadas por un vertido marítimo. Este comportamiento será ejecutado por cada uno de los agentes que forman el enjambre de manera individual. Se trata de un comportamiento simple, capaz de ejecutarse en sistemas con baja capacidad de procesamiento, que además no requiere comunicación entre agentes ni sistemas de localización globales. Esto puede provocar que el enjambre tienda más lentamente hacia el vertido, dependiendo del terreno a cubrir y el número de agentes. Sin embargo, se trata de un comportamiento más versátil y fácil de implantar incluso en zonas sin cobertura de GPS.

Se ha verificado cómo el proceso de marcado sin comunicación directa es robusto y eficiente, haciendo un seguimiento del enjambre continuado. El enjambre es capaz de delimitar completamente el vertido, si el número de agentes es suficiente. Para esta última parte un agente ha de ser capaz de detectar aquellos agentes que estén cerca de él. Existen varias maneras de conseguirlo: utilizar una cámara, transmitir la posición GPS de los agentes, etc.

En muchos países se requiere que los drones emitan de manera continua una baliza indicando su posición. En Europa se utilizan los 433Mhz para

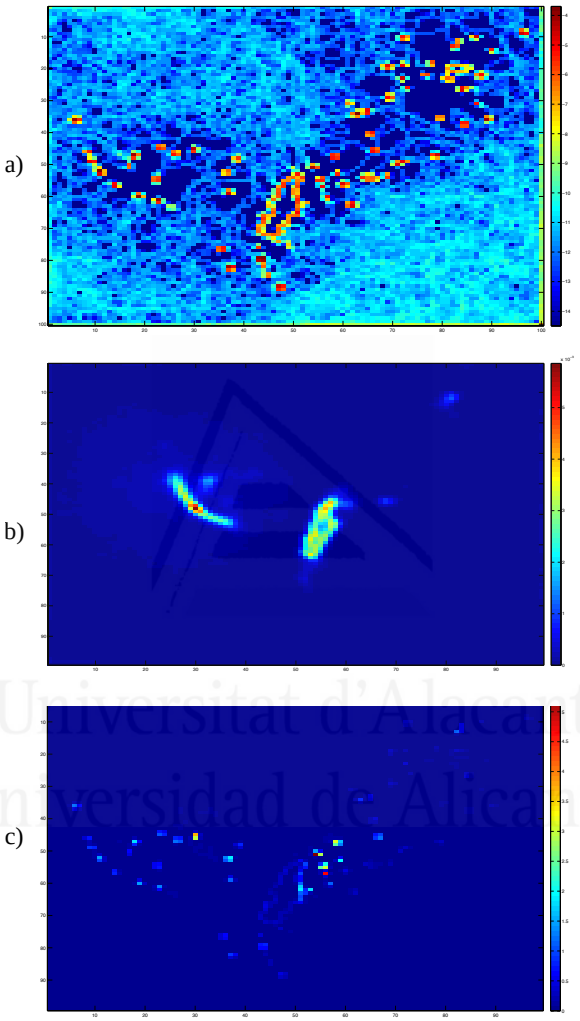


Figura 7.14: Comparación de la predicción microscópica junto con el modelo macroscópico para el mapa M en el instante $t = 140h$. a) $(log(P_{micro}))$: se muestra el logaritmo de la distribución de probabilidades obtenida al simular 200 agentes durante 30.000 segundos. Se utiliza una distribución logarítmica para resaltar el estado de probabilidad baja. b) (P_{macro}): distribución de probabilidad obtenida mediante la ecuación Fokker-Planck. c) producto de ambas distribuciones, decrementando la importancia de los valores altos en la simulación microscópica (concretamente $P_{macro} \times P_{micro}^{1/2}$)

tal efecto. Por este motivo, sería posible utilizar esta infraestructura, ya que se puede detectar la intensidad de señal en una determinada zona. Si esta intensidad crece con el movimiento del agente, éste debe tomar una ruta alternativa. Como aproximación de enjambre, no se trata de ningún tipo de comunicación entre agentes, sino de una simple baliza, requerida para poder conocer, si fuera necesario, la posición de los drones.

Por estas características, este comportamiento puede ser implementado utilizando únicamente los agentes μ de la arquitectura de enjambre propuesta ya que los agentes no necesitan ninguna información, más que aquella percibida por sus sensores, ni ningún tipo de coordinación o comunicación con otros miembros del enjambre. Aunque cabe destacar que sería posible utilizar los agentes Ψ para tareas más complejas como la recopilación de datos del enjambre para aportar información de más alto nivel, como un mapa de localización del vertido.

Para probar este comportamiento microscópico se han realizado varios conjuntos de pruebas: la detección de una mancha de vertido; la detección de varias manchas; y el seguimiento de un vertido en movimiento. Todas las pruebas con resultados satisfactorios.

Una vez comprobado el modelo microscópico, se ha propuesto un modelo macroscópico que demuestra que la tendencia del enjambre, para un número suficiente de agentes, es la que se aprecia en el modelo microscópico. La conexión de ambos modelos se ha probado para una mancha compleja, generada con la aplicación GNOME.

Es importante tener en mente las diferencias entre ambos modelos. El modelo microscópico es el que define el comportamiento individual, y como tal es sencillo de entender a nivel local. Sin embargo, el comportamiento de todo el enjambre no queda definido por este modelo. Se pueden realizar pruebas para un elevado número de agentes para poder analizar el comportamiento global. Sin embargo estas pruebas son muy costosas y no están exentas de problemas. El modelo macroscópico establece el comportamiento global del enjambre. Permite por tanto verificar el comportamiento emergente de la interacción de todos los agentes que ejecutan el modelo microscópico. El modelo macroscópico permite demostrar la tendencia del enjambre para un número suficiente de agentes. Su análisis es también complejo, al tratar con ecuaciones diferenciales que, por ejemplo, obligan a elegir un único punto para empezar la simulación. Aún así las ventajas son destacables, como el análisis continuo para cualquier punto del espacio y tiempo de la probabilidad de que se encuentre un determinado agente, o los tiempos de simulación despreciables si comparamos con el muestreo del modelo microscópico.

Las pruebas realizadas demuestran resultados satisfactorios tanto a nivel

local, con el modelo microscópico, como a nivel global, con el modelo macroscópico, quedando reflejados en ambos modelos las características fundamentales del comportamiento definido para la detección y seguimiento de un vertido de petróleo.

Capítulo 8

Conclusiones y líneas futuras

Este capítulo resume el trabajo realizado en el desarrollo de esta tesis. En primer lugar se presentan las conclusiones del trabajo realizado revisando los contenidos descritos en los diferentes capítulos, se describen las aportaciones realizadas y se indican las publicaciones obtenidas. Finalmente, se indican las líneas de trabajo futuras de esta investigación.

8.1. Conclusiones

La robótica de enjambre estudia el uso de grandes grupos de robots simples, de manera que estos grupos puedan llevar a cabo tareas colectivas que están fuera de las capacidades de un único robot. Este comportamiento colectivo emerge de forma auto-organizada a partir de las interacciones entre los robots y entre los robots y el entorno.

A la hora de alcanzar estos comportamientos, los sistemas robóticos de enjambre deben cumplir una serie de requerimientos que los diferencian de otros sistemas multi-robóticos. Los robots que forman el enjambre deben ser autónomos, relativamente sencillos e ineficientes, con sensores locales y mecanismos de comunicación limitados. Además, los sistemas deben ser robustos, flexibles y escalables y no deben disponer de ningún módulo de control centralizado.

Teniendo en cuenta estas propiedades de la robótica de enjambre, es posible ver que comparten muchas características con los sistemas multi-agente como la autonomía de los agentes, el trabajo conjunto o la descentralización de los datos. Por lo tanto, el uso de un sistema multi-agente para desarrollar sistemas robóticos de enjambre puede ser una buena alternativa, aunque esta asociación no se puede realizar de manera directa por las exigencias de

control distribuido y tolerancia a fallos requeridas por la robótica de enjambre.

Se ha visto cómo con el uso de una plataforma de agentes como *JADE* es posible solventar estas limitaciones. *JADE* proporciona dos herramientas para asegurar que el sistema permanece en funcionamiento ante errores en el contenedor principal, la replicación del AMS y la persistencia del DF. Con la replicación del contenedor principal se crea un anillo lógico de contenedores, de tal manera que, si uno de los contenedores falla, el resto son capaces de detectar el error y llevar a cabo las operaciones de reestructuración necesarias.

Por otro lado, *JADE* no tiene en cuenta posibles problemas de pérdida de cobertura o desconexiones. Se ha propuesto una solución a este problema, la incorporación de una red *MANET* que proporciona las funcionalidades de red necesarias sin la existencia de ningún administrador central y sin utilizar infraestructuras de red fija.

Con el objetivo de definir comportamientos para el control de un sistema robótico de enjambre y ante la posibilidad de utilizar los sistemas multi-agente para el desarrollo de este tipo de sistemas, se ha definido un modelo de arquitectura de agentes para enjambres de robots. En primer lugar se ha realizado una revisión de las arquitecturas robóticas existentes, tanto individuales como multi-robóticas. Estas arquitecturas, aunque adecuadas para determinadas tareas, no cumplen los requisitos impuestos en robótica de enjambre. Por este motivo, se ha definido un modelo de arquitectura híbrida basada en un sistema multi-agente para el control de un sistema robótico de enjambre. Esta arquitectura dispone de dos niveles. El nivel inferior define el enjambre puro. Este nivel está formado por los robots que cumplen de manera estricta todas las restricciones que señala la robótica de enjambre. Sin embargo, en algunas ocasiones y, para determinadas tareas, es complicado alcanzar los objetivos marcados utilizando únicamente un grupo de robots que cumple las restricciones de la robótica de enjambre. Por este motivo, se define una capa superior que complementa las funcionalidades del enjambre.

Esta división en dos niveles permite que la capa inferior sea utilizada por separado para tareas donde el comportamiento colectivo deseado puede emerger únicamente a partir de la interacción local entre los agentes y entre los agentes y el entorno.

Teniendo como marco la definición anteriormente propuesta, se han desarrollado tres comportamientos básicos de los sistemas de enjambre: agregación, *flocking* y dispersión.

En el comportamiento de agregación los individuos son capaces de agre-

garse (formar un único grupo) de manera auto-organizada. La conducta de los agentes viene determinada por una máquina de estados finita que depende de un valor P. Según este valor los agentes tendrán un comportamiento diferente. En primer lugar se ha llevado a cabo este comportamiento utilizando únicamente el nivel inferior de la arquitectura, es decir, el enjambre puro. En este caso, se ha visto cómo los agentes eran capaces de agregarse pero no siempre formaban un único grupo. Haciendo uso de los agentes de más alto nivel, es posible modificar el valor de P, para utilizar información global y por tanto que tenga en cuenta el tamaño del grupo más grande. Con el uso del segundo nivel de la arquitectura los agentes son capaces de agregarse formando un único grupo.

En el comportamiento de *flocking* los agentes son capaces de formar un grupo y mantener un movimiento coordinado, suave, evitando los objetos del entorno. Este comportamiento se ha llevado a cabo únicamente con la parte reactiva de la arquitectura, es decir, con el enjambre puro, ya que el comportamiento emerge a partir de las lecturas de los sensores de cada robot. Se ha visto cómo el enjambre va formando un grupo y es capaz de mantenerse en movimiento de manera coordinada, incluso con la existencia de obstáculos en el entorno.

En el comportamiento de dispersión se intenta maximizar la cobertura de un área. Es un comportamiento básico para llevar a cabo tareas más complejas como búsquedas, rescates, exploraciones, etc. Para este comportamiento se han definido tres conductas. La primera de ellas es un comportamiento de paseo aleatorio donde los robots se mueven libremente por el entorno evitando los obstáculos que puedan encontrarse. En la segunda conducta se ha utilizado el comportamiento de *flocking* definido anteriormente para una tarea de cobertura de un área. En la última conducta los agentes son capaces de detectar los robots que tienen alrededor y de dirigirse a zonas abiertas (con menos robots). La segunda y tercera conducta obtienen mejores resultados que el paseo aleatorio, en cuanto al tiempo necesario para cubrir el entorno, pero se ha visto cómo todas las conductas son capaces de alcanzar el objetivo y explorar todo el entorno.

A continuación se han realizado dos casos de estudio definiendo dos comportamientos de enjambre. El primer caso de estudio define un comportamiento para la localización de recursos en el entorno, el segundo define un comportamiento para la detección y seguimiento de un vertido petrolífero.

Le explotación colectiva de recursos es una aplicación que ha tomado especial relevancia en este campo, ya que cualquier tarea donde existan una serie de objetos o recursos distribuidos físicamente que necesitan ser explorados, inspeccionados o recolectados es una aplicación potencial de la robótica

de enjambre. En el primer caso de estudio se define un sistema de enjambre para la explotación colectiva de recursos sin conexión explícita entre los agentes. Para este comportamiento se han definido, por un lado, las capacidades individuales de cada agente mediante un modelo microscópico y, por otro lado, un modelo macroscópico orientado a predecir el comportamiento global del enjambre en el entorno. Se ha visto cómo el enjambre es capaz de agregarse sobre las fuentes de recursos en todos los casos estudiados y cómo el modelo macroscópico predice de manera correcta el número de agentes que se encuentran en cada estado, coincidiendo éstos con las simulaciones realizadas.

Otra de las aplicaciones donde este tipo de sistemas es cada vez más utilizado es en la exploración de entornos y su monitorización. Estas aplicaciones tienen varios usos ecológicos como por ejemplo la detección y seguimiento de vertidos de petróleo, nubes tóxicas o bancos de algas. En el segundo caso de estudio se ha presentado un comportamiento de enjambre capaz de detectar, monitorizar, cubrir y marcar el perímetro de un vertido de petróleo marítimo. Teniendo en cuenta las propiedades de los vertidos de petróleo se ha definido, por un lado, un modelo microscópico que describe el comportamiento de cada agente individual, donde los agentes buscarán un vertido y una vez encontrado se situarán dentro de él para cubrirlo o en su perímetro para marcarlo según el comportamiento deseado. Por otro lado, se define un comportamiento macroscópico para analizar el comportamiento global del enjambre. Este modelo macroscópico permite analizar a nivel global la evolución del enjambre y obtener una distribución de la posición del enjambre en cualquier instante. Mediante las pruebas realizadas se ha demostrado que ambos modelos identifican correctamente las zonas de vertido.

A continuación, en el contexto de los objetivos fijados inicialmente, se detallarán las aportaciones realizadas en esta tesis:

- **Uso de sistemas multi-agente en robótica de enjambre.** Una vez se ha realizado un estudio del estado del arte de la robótica de enjambre donde se han descrito el tipo de problemas apropiados para ser resueltos por este campo y las propiedades de los sistemas robóticos de enjambre, se ha visto que estos sistemas comparten muchas características con los sistemas multi-agente tanto por las características de los propios sistemas en sí, como por el tipo de tareas que pueden realizar.

  Sin embargo, la aplicación de los sistemas multi-agente a la robótica de enjambre no se puede realizar de manera directa por los requerimientos de tolerancia a fallos y robustez de los sistemas robóticos de

enjambre. Por este motivo se han propuesto los siguientes mecanismos para proporcionar un sistema multi-agente robusto, adecuado a la robótica de enjambre: replicación del AMS, persistencia del DF, y uso de redes *MANET*.

Se ha aportado un estudio del funcionamiento de las opciones de tolerancia a fallos proporcionadas por *JADE* indicándose los pasos seguidos internamente por los mismos.

- **Modelo de arquitectura para la robótica de enjambre.** Se ha realizado un estudio de las arquitecturas robóticas existentes tanto de las arquitecturas individuales como multi-robóticas. En este estudio se ha visto que estas arquitecturas no cumplen, de una u otra forma, las características establecidas por la robótica de enjambre. Por este motivo, se propone un modelo de arquitectura híbrida para el control de un sistema de enjambre basada en un sistema multi-agente. Este modelo tiene en cuenta tanto los requerimientos de la robótica de enjambre, como las necesidades de alto nivel.

  El modelo aportado cumple las características de robustez, flexibilidad y escalabilidad requeridas por la robótica de enjambre. Además, permite la definición de sistemas descentralizados y distribuidos que pueden trabajar sin comunicación explícita y que están formados por agentes con capacidades limitadas.

- **Comportamientos básicos de robótica de enjambre.** Se han definido tres comportamientos básicos de este campo: agregación, *flocking* y dispersión. Estos comportamientos pueden ser considerados como fundamentales ya que son precursores de otros comportamientos más complejos. En los tres comportamientos el enjambre ha sido capaz de alcanzar los objetivos marcados.

  En el comportamiento de *flocking* y dispersión se han alcanzado los objetivos deseados mediante el nivel inferior de la arquitectura, el enjambre. En el comportamiento de *flocking* el enjambre forma un único grupo y es capaz de coordinar su movimiento, evitando los obstáculos del entorno. En el comportamiento de dispersión se han estudiado tres estrategias: una estrategia aleatoria, el comportamiento de *flocking* anterior y una conducta donde los robots se dirigen hacia áreas con menos robots. En todas las estrategias el enjambre es capaz de cubrir el entorno.

  En el comportamiento de agregación, haciendo uso únicamente del en-

jambre puro, no se obtenía en todas las ocasiones un resultado óptimo. Para poder alcanzar siempre la agregación de la totalidad del enjambre en un único grupo se ha utilizado el nivel superior de la arquitectura para proporcionar información necesaria al enjambre. En este caso el enjambre es capaz de agregarse en un único grupo.

- **Comportamiento para la localización de recursos.** Se ha aportado un comportamiento para la localización colectiva de recursos. Este comportamiento ha sido desarrollado con el nivel inferior de la arquitectura, nivel de enjambre. Se basa en agentes simples, sin conexión explícita. Se ha definido tanto un nivel microscópico que especifica las capacidades de cada agente individual, como un nivel macroscópico capaz de analizar la convergencia de todo el enjambre. Se ha visto cómo el modelo macroscópico predice correctamente el estado de los agentes, coincidiendo con las simulaciones realizadas. También, se ha visto cómo el sistema es capaz de localizar de manera emergente la fuente de recursos más prometedora en un entorno desconocido, con ruido y con varias fuentes de recursos.

- **Comportamiento para la detección y seguimiento de un vertido petrolífero.** Se ha aportado un comportamiento de enjambre capaz de detectar, monitorizar, cubrir y marcar el perímetro de un recurso. Concretamente, de un vertido petrolífero marítimo. Para esta tarea se ha utilizado el enjambre puro, ya que los agentes utilizan únicamente su información sensorial. Para este estudio se han modelado vertidos de petróleo utilizando *GNOME*. Este sistema permite el modelado realista de vertidos haciendo uso de mapas meteorológicos reales, corrientes marítimas y vientos. Teniendo en cuenta las propiedades de los vertidos de petróleo, se ha definido un modelo microscópico que especifica el comportamiento de cada agente. Posteriormente, se ha definido un modelo macroscópico que predice el funcionamiento del sistema a nivel global. Las pruebas realizadas demuestran que ambos modelos reflejan las características fundamentales del comportamiento y que el sistema es capaz de detectar una mancha de vertido, varias manchas producidas por la dispersión de éste y realizar el seguimiento de estas manchas en movimiento.

A continuación se presentan las publicaciones relacionadas con el trabajo de investigación realizado:

- Modelling Oil-Spill detection with swarm drones.Abstract and Applied Analysis. Sometido a revisión 2013. Aznar, F.; Sempere, M.; Pujol, M.; Rizo, R. y Pujol, M.J.
- A macroscopic model for high intensity radiofrequency signal detection in swarm robotics systems. International Journal of Computer Mathematics. 2013. Aznar, F.; Pujol, M.; Sempere, M. y Rizo, R.
- A Macroscopic Model for Signal Detection in Swarm Robotics. Mathematical Modelling in Engineering and Human Behaviour. 2013. Aznar, F.; Pujol, M.; Pujol, F.; Sempere, M.; Pujol, M.J. y Rizo, R.
- Distributed column formation in swarm robotics for simple agents. Conferencia de la Asociación Española para la Inteligencia Artificial. 2013. Arques, P.; Aznar, F. y Sempere, M.
- Implementación de una arquitectura de agentes para robótica de enjambre utilizando JADE. XIV Conferencia de la Asociación Española para la Inteligencia Artificial. 2011. Sempere, M.; Aznar, F.; Pujol, M. y Rizo R.
- Swarm System for Resource Location with Non Explicitly Connected Agents. International Journal of Artificial Intelligence. 2011. Sempere, M.; Aznar, F.; Pujol, M. y Rizo, R.
- Agents for Swarm Robotics: Architecture and Implementation. Advances in Intelligent and Soft Computing. 2011. Aznar, F.; Sempere, M.; Mora, F.J.; Arques, P.; Puchol, J.; Pujol, M. y Rizo, R.
- On Cooperative Swarm Foraging for simple, Non Explicitly Connected, Agents. 2010. Advances in Intelligent and Soft Computing. Sempere, M.; Aznar, F.; Pujol, M. y Rizo, R.
- On Intelligent Interface Agents for Human Based Computation. Lecture Notes in Computer Science. 2007. Aznar, F.; Sempere, M.; Pujol, M. y Rizo R.
- Building a Robotic Eye and Eyelid for Human-Robot Interaction. Portuguese Conference on Artificial Intelligence. 2007. Mateo, J.; Aznar, F.; Sempere, M.; Pujol, M. y Rizo, R.
- Learning Discrete Probability Distribution with a Multi-resolution Binary Tree. Lecture Notes in Computer Science. 2007. Sanchis, F.A.; Aznar, F.; Sempere, M.; Pujol, M. y Rizo, R.

- 3D Robot Mapping: Combining Active and Non Active Sensors in a Probabilistic Framework. Lecture Notes in Computer Science. 2006. F.; Sempere, M.; Pujol, M., Rizo R. y Molina, R.

- Bayesian Emotions: Developing an Interface for Robot/Human Communication. Lecture Notes in Computer Science. 2005. Aznar, F.; Sempere, M.; Pujol, M. y Rizo R.

- A Cognitive Model for Autonomous Agents Based on Bayesian Programming. ecture Notes in Computer Science. 2005. znar, F.; Sempere, M.; Pujol, M. y Rizo R.

8.2. Líneas futuras

En este último apartado y como finalización de este trabajo, realizaremos una aproximación a las líneas futuras que se pueden trazar en el horizonte teniendo como punto de partida las aportaciones de esta investigación.

El trabajo futuro a desarrollar se enmarca principalmente en dos líneas de investigación detalladas a continuación.

Por una parte, el avance en el estudio y diseño de comportamientos. La robótica de enjambre tiene un amplio rango de aplicaciones y existen muchas tareas susceptibles de llevarse a cabo mediante este tipo de sistemas. Esta línea de investigación futura se centra en el desarrollo de nuevos comportamientos para la robótica de enjambre. Concretamente queremos centrarnos en comportamientos enfocados a resolver problemáticas reales, como la monitorización del entorno de un lago o embalse, la detección y seguimiento de incendios o nubes tóxicas, el control de plagas en agricultura, etc. En definitiva, aspectos relacionados con la seguridad y vigilancia en diferentes ámbitos.

Por otra parte, la implantación de este tipo de comportamientos en sistemas reales. La robótica de enjambre es un campo de investigación actualmente activo, donde han sido desarrollados muchos comportamientos adecuados a determinadas tareas. La mayoría de estas investigaciones han sido realizadas en entornos simulados o bien, en caso de realizarse con sistemas físicos reales, han sido realizadas en entornos acotados, específicamente diseñados y preparados para estas pruebas. A pesar del potencial de los sistemas robóticos de enjambre, éstos aun no han sido ampliamente extendidos a entornos reales.

Dentro de esta segunda línea de investigación se trabajará principalmente en dos aspectos: diseño y creación de un enjambre de drones y la

implantación de los comportamientos en éstos.

Actualmente, nos encontramos trabajando en el diseño y construcción de un enjambre de hexacópteros como el mostrado en la figura 8.1.

Figura 8.1: Prototipo de hexacóptero para formar un enjambre de drones.

El núcleo central de este dron es una placa de control de vuelo ArduPilot que se compone de *GPS*, barómetro, acelerómetro y giroscopio. Esta placa se conecta mediante el protocolo *Mavlink* a una placa de control que permite la realización de tareas más complejas. Cada uno de los drones dispondrá de un módulo *XBee* para conectarse entre ellos.

Una vez desarrollado el enjambre de drones, se trabajará en el ajuste de los comportamientos definidos para poder implantarlos en sistemas reales.

Bibliografía

[AAC10] Gianluca Antonelli, Filippo Arrichiello, and Stefano Chiaverini. Flocking for multi-robot systems via the null-space-based behavioral control. *Swarm Intelligence*, 4:37–56, 2010.

[AB97] Ronald C. Arkin and Tucker Balch. Aura: Principles and practice in review. *Journal of Experimental and Theoretical Artificial Intelligence*, 9:175–189, 1997.

[AC87] MarkV. Abrahams and PatrickW. Colgan. Fish schools and their hydrodynamic function: a reanalysis. *Environmental Biology of Fishes*, 20(1):79–80, 1987.

[AML89] J. Albus, H. McCain, and R. Lumia. Nasa/nbs standard reference model for telerobot control system architecture (nasrem). Technical report, National Institute of Standards and Technology, 1989.

[AOP10] Saleh Ali K. Al-Omari and Sumari Putra. An overview of mobile ad hoc networks for the existing protocols and applications. *Journal on Applications of Graph Theory in Wireless Ad hoc Networks and Sensor Networks*, 2(1):87–110, 2010.

[Ark89] R.C. Arkin. Motor schema-based mobile robot navigation. *International Journal of Robotics research*, 8(4):92–112, 1989.

[Ark90] Ronald C. Arkin. Integrating behavioral, perceptual, and world knowledge in reactive navigation. *Robotics and Autonomous Systems*, 6(1-2):105 – 122, 1990. Designing Autonomous Agents.

[Azn06] F. Aznar. *FSR-BAY: Modelo probabilístico para la Fusión Sensorial Robótica*. PhD thesis, Universidad de Alicante, 2006.

[BCG07] F. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent Systems with JADE*. John Wiley & Sons, Ltd, 2007.

[Bel03] Xavier Belles. *Insecticidas biorracionales*. Consejo Superior de Investigaciones Científicas, 2003.

[Ben05] G. Beni. From swarm intelligence to swarm robotics. *Swarm Robotics. Lecture Notes in Computer Science*, 3342/2005:1–9, 2005.

[BFMP82] Paul D Boehm, David L Fiest, Donald Mackay, and Sally Paterson. Physical-chemical weathering of petroleum hydrocarbons from the ixtoc i blowout: Chemical measurements and a weathering model. *Environmental Science & Technology*, 16(8):498–505, 1982.

[BKJJ98] R.P. Bonasso, R. Kerri, K. Jenks, and G. Johnson. Using the 3t architecture for tracking shuttle rms procedures. In *Intelligence and Systems, 1998. Proceedings., IEEE International Joint Symposia on*, pages 180 –187, may 1998.

[Bro86] R. Brooks. A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of*, 2(1):14–23, mar 1986.

[BS03] M.A. Batalin and G. Sukhatme. Efficient exploration without localization. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 2, pages 2714–2719, 2003.

[BS07] L. Bayindir and E. Sahin. A review of studies in swarm robotics. *Turkish Journal Electrical Engineering and Computer Sciences*, 2:115–147, 2007.

[BS09] L. Bayindir and E. Sahin. Modeling self-organized aggregation in swarm robotic systems. In *Swarm Intelligence Symposium, 2009. SIS '09. IEEE*, pages 88 –95, 2009.

[BSdM03] Didac Busquets, Carles Sierra, and Ramon Lopez de Mantaras. A multiagent approach to qualitative landmark-based navigation. *Autonomous Robots*, 15(2):129–154, 2003.

[BWH03] M. Beger, M. Watzke, and H. Helin. Toward a fipa approach for mobile ad hoc environment. In *Intelligence in Next Generation Networks (ICIC'03)*, 2003.

[CB08] Justin Collins and Rajive Bagrodia. Programming in mobile ad hoc networks. In *Proceedings of the 4th Annual International Conference on Wireless Internet*, WICON '08, pages 73:1–73:9, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[CD07] Alexandre Campo and Marco Dorigo. Efficient multi-foraging in swarm robotics. In *Proceedings of the 9th European conference on Advances in artificial life*, ECAL'07, pages 696–705, Berlin, Heidelberg, 2007. Springer-Verlag.

[CF07] Justin Clark and Rafael Fierro. Mobile robotic sensors for perimeter detection and tracking. {*ISA*} *Transactions*, 46(1):3–13, 2007.

[CFK97] Y. Uny Cao, Alex S. Fukunaga, and A. B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4:226–234, 1997.

[CKBM06] David W. Casbeer, Derek B. Kingston, Randal W. Beard, and Timothy W. McLain. Cooperative forest fire surveillance using a team of small unmanned air vehicles. *Int. J. Systems Science*, 37(6):351–360, 2006.

[CM07] N. Correll and A. Martinoli. Robust distributed coverage using a swarm of miniature robots. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 379 –384, april 2007.

[CM10] Ibrahim Cil and Murat Mala. A multi-agent architecture for modelling and simulation of small military unit combat in asymmetric warfare. *Expert Syst. Appl.*, 37(2):1331–1343, 2010.

[CM11] Nikolaus Correll and Alcherio Martinoli. Modeling and designing self-organized aggregation in a swarm of miniature robots. *The International Journal of Robotics Research*, 30(5):615–626, 2011.

[COD08] Anders Christensen, Rehan O,ÄôGrady, and Marco Dorigo. Swarmorph-script: a language for arbitrary morphology generation in self-assembling robots. *Swarm Intelligence*, 2:143–165, 2008.

[Con92] J.H. Connell. Sss: a hybrid architecture applied to robot navigation. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, volume 3, pages 2719 – 2724, may 1992.

[DDPG11] Frederick Ducatelle, GianniA. DiCaro, Carlo Pinciroli, and LucaM. Gambardella. Self-organized cooperation between robotic swarms. *Swarm Intelligence*, 5(2):73–96, 2011.

[dLAABM06] Alessandro de Luna Almeida, Samir Aknine, Jean-Pierre Briot, and Jacques Malenfant. Plan-based replication for fault-tolerant multi-agent systems. In *Proceedings of the 20th international conference on Parallel and distributed processing*, IPDPS'06, pages 347–347, Washington, DC, USA, 2006. IEEE Computer Society.

[DLABA+07] Alessandro De Luna Almeida, Jean-Pierre Briot, Samir Aknine, Zahia Guessoum, and Olivier Marin. Towards autonomic fault-tolerant multi-agent systems. In *2nd Latin American Autonomic Computing Symposium (LAACS'07)*, 9 2007.

[DS04] M Dorigo and E. Sahin. Swarm robotics- special issue editorial. *Autonomous Robots*, 17:2–3, 2004.

[DTG+05] M. Dorigo, E. Tuci, R. Grob, V. Trianni, T.H. Labella, and S. Nouyan. The swarm-bots project. *Swarm Robotics. Lecture Notes in Computer Science*, 3342, 2005.

[DTUV08] Christophe Dony, Chouki Tibermacine, Christelle Urtado, and Sylvain Vauttier. Specification of an exception handling system for a replicated agent environment. In *Proceedings of the 4th international workshop on Exception handling*, WEH '08, pages 24–31, New York, NY, USA, 2008. ACM.

[GBFM10] Zahia Guessoum, Jean-Pierre Briot, Nora Faci, and Olivier Marin. Towards reliable multi-agent systems - an adaptive replication mechanism. *Multiagent and Grid Systems - An International Journal*, 6(1):1–24, 3 2010.

[GBMD06] R. Gross, M. Bonani, F. Mondada, and M. Dorigo. Autonomous self-assembly in swarm-bots. *Robotics, IEEE Transactions on*, 22(6):1115 –1130, dec. 2006.

[GCD+03] Dani Goldberg, Vincent Cicirello, M Bernardine Dias, Reid Simmons, Stephen Smith , and Anthony (Tony) Stentz. Market-based multi-robot planning in a distributed layered architecture. In Frank E. Schneider Alan C. Shultz, Lynne E. Parker, editor, *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2003 International Workshop on Multi-Robot Systems*, volume 2, pages 27–38. Kluwer Academic Publishers, 2003.

[GD09] Roderich Gross and Marco Dorigo. Towards group transport by swarms of robots. *Int. J. Bio-Inspired Comput.*, 1(1/2):1–13, 2009.

[GGT07] S. Gernier, J. Gautrais, and G. Theraulaz. The biological principles of swarm intelligence. *Swarm Intelligence*, 1:3–31, 2007.

[GM07] Ian A. Gravagne and Robert J. Marks. Emergent behaviors of protector, refugee, and aggressor swarms. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(2):471–476, 2007.

[Ham10a] Heiko Hamann. A framework of models for swarm robotic systems. In *Space-Time Continuous Models of Swarm Robotic Systems*, volume 9 of *Cognitive Systems Monographs*, pages 41–76. Springer Berlin Heidelberg, 2010.

[Ham10b] Heiko Hamann. *Space-Time Continuous Models of Swarm Robotic Systems: Supporting Global-to-Local Programming*, volume 9. Springer, 2010.

[HLW09] Jingwen He, Hokyin Lai, and Huaiqing Wang. A commonsense knowledge base supported multi-agent architecture. *Expert Syst. Appl.*, 36(3):5051–5057, 2009.

[HMDD04] Jeroen Hoebeke, Ingrid Moerman, Bart Dhoedt, and Piet Demeester. An overview of mobile ad hoc networks: Applications and challenges. *Journal of the Communications Network*, 3:60–66, 2004.

[Hor12] Cay S. Horstmann. *Scala for the impatient*. Addison-Wesley, 2012.

[HSWN10] N.R. Hoff, A. Sagoff, R.J. Wood, and R. Nagpal. Two foraging algorithms for robot swarms using only local communication. In *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*, pages 123–130, 2010.

[HTM09] Fiona Higgins, Allan Tomlinson, and Keith M. Martin. Survey on security challenges for swarm robotics. In *ICAS '09: Proceedings of the 2009 Fifth International Conference on Autonomic and Autonomous Systems*, pages 307–312, Washington, DC, USA, 2009. IEEE Computer Society.

[HW07] Heiko Hamann and Heinz Wörn, H.rn. An analytical and spatial model of foraging in a swarm of robots. In Erol Sahin, William Spears, and Alan Winfield, editors, *Swarm Robotics*, volume 4433 of *Lecture Notes in Computer Science*, pages 43–55. Springer Berlin / Heidelberg, 2007.

[HW08] Heiko Hamann and Heinz Wörn. A framework of space–time continuous models for algorithm design in swarm robotics. *Swarm Intelligence*, 2(2-4):209–239, 2008.

[ILS07] Bianca Innocenti, Beatriz Lopez, and Joaquim Salvi. A multi-agent architecture with cooperative fuzzy control for a mobile robot. *Robotics and Autonomous Systems*, 55(12):881 – 891, 2007.

[JaGAJ10] A. Jevtić and, P. Gazi, D. Andina, and M. Jamshidi. Building a swarm of robotic bees. In *World Automation Congress (WAC), 2010*, pages 1 –6, sept. 2010.

[KB00] C. Ronald Kube and Eric Bonabeau. Cooperative transport by ants and robots. *Robotics and Autonomous Systems*, 30:85–101, 2000.

[KBH08] D. Kingston, R.W. Beard, and R.S. Holt. Decentralized perimeter surveillance using a team of uavs. *Robotics, IEEE Transactions on*, 24(6):1394–1404, 2008.

[KC00] Sanjeev Kumar and Philip R. Cohen. Towards a fault-tolerant multi-agent system architecture. In *Proceedings of the fourth*

international conference on Autonomous agents, AGENTS '00, pages 459–466, New York, NY, USA, 2000. ACM.

[KC06] G. Kim and W. Chung. Tripodal schematic control architecture for integration of multi-functional indoor service robots. *IEEE Transactions on Industrial Electronics*, 53(5):1723–1736, 2006.

[Kla07] E. Klavins. Programmable self-assembly. *Control Systems, IEEE*, 27(4):43 –56, aug. 2007.

[KMRS97] Kurt Konolige, Karen Myers, Enrique Ruspini, and Alessandro Saffiotti. The saphira architecture: A design for autonomy. *Journal of Experimental and Theoretical Artificial Intelligence*, 9:215–235, 1997.

[KSA+05] Z.A. Khan, S. Shahid, H.F. Ahmad, A. Ali, and H. Suguri. Decentralized architecture for fault tolerant multi agent system. In *Autonomous Decentralized Systems, 2005. ISADS 2005. Proceedings*, pages 167 – 174, april 2005.

[KTI+11] Yael Katz, Kolbjørn Tunstrøm, Christos C. Ioannou, Cristián Huepe, and Iain D. Couzin. Inferring the structure and dynamics of interactions in schooling fish. *Proceedings of the National Academy of Sciences*, 108(46):18720–18725, 2011.

[LA11] Jong-Hyun Lee and Chang Wook Ahn. Improving energy efficiency in cooperative foraging swarm robots using behavioral model. In *Bio-Inspired Computing: Theories and Applications (BIC-TA), 2011 Sixth International Conference on*, pages 39 –44, sept. 2011.

[LCRPS04] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, and Keith Sullivan. Mason: A new multi-agent simulation toolkit. In *Proceedings of the 2004 SwarmFest Workshop*, volume 8, 2004.

[LEF09] Xiang Li, M.Fikret Ercan, and YuFai Fung. A triangular formation strategy for collective behaviors of robot swarm. In Osvaldo Gervasi, David Taniar, Beniamino Murgante, Antonio Lagan, Youngsong Mun, and MarinaL. Gavrilova, editors, *Computational Science and Its Applications ,ICCSA 2009*, volume 5592 of *Lecture Notes in Computer Science*, pages 897–911. Springer Berlin Heidelberg, 2009.

[LG02] Kristina Lerman and Aram Galstyan. Mathematical model of foraging in a group of robots: Effect of interference. *Autonomous Robots*, 13:127–141, 2002.

[LG06] Luke Ludwig and Maria Gini. Robotic swarm dispersion using wireless intensity signals. In *in Proc. Int'l Symp. on Distributed Autonomous Robotic Systems*, pages 135–144, 2006.

[Liu08] W. Liu. *Design and Modelling of Adaptive Foraging in Swarm Robotic Systems*. PhD thesis, University of the West of England, 2008.

[LJYL12] Joo-Baek Leem, Wonju Jeon, Chi-Young Yun, and Sang-Hee Lee. Quantitative analysis of fish schooling behavior with different numbers of medaka (oryzias latipes) and goldfish (carassius auratus). *Ocean Science Journal*, 47(4):445–451, 2012.

[LL10] Ting Lan and Wenlei Li. A macroscopic state model of swarm robot system. In *Intelligent Systems (GCIS), 2010 Second WRI Global Congress on*, volume 2, pages 258 –261, 2010.

[LLC11] Suwantaweekul Lalitta, Geunho Lee, and Nak Young Chong. Coverage control of a robotic swarm for pollution monitoring. In *Ubiquitous Robots and Ambient Intelligence (URAI), 2011 8th International Conference on*, pages 188 –192, nov. 2011.

[LMG05] K. Lerman, A. Martinoli, and A. Galstyan. A review of probabilistic macroscopic models for swarm robotic systems. *Swarm Robotics Workshop: State of the art Suervey.*, 3342:143–152, 2005.

[LOC00] M. Lindstrom, A. Oreback, and H.I. Christensen. Berra: a research architecture for service robots. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 4, pages 3278 –3283, 2000.

[LSPB05] Sean Luke, Keith Sullivan, Liviu Panait, and Gabriel Balan. Tunably decentralized algorithms for cooperative target observation. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, AAMAS '05, pages 911–917. ACM, 2005.

[LW09] W. Liu and A. Winfield. A macroscopic probabilistic model of adaptive foraging in swarm robotics systems. In *6th Vienna International Conference on Mathematical Modelling*, 2009.

[LW10] Wenguo Liu and Alan F. T. Winfield. Modeling and optimization of adaptive foraging in swarm robotic systems. *Int. J. Rob. Res.*, 29(14):1743–1760, 2010.

[LZF10] Yanmin Lei, Qidan Zhu, and Zhibin Feng. Research on architecture of multiple robots system based on the dynamic networks. In *Information and Automation (ICIA), 2010 IEEE International Conference on*, pages 93–98, 2010.

[MA92] R.R. Murphy and R.C. Arkin. Sfx: An architecture for action-oriented sensor fusion. In *Intelligent Robots and Systems, 1992., Proceedings of the 1992 lEEE/RSJ International Conference on*, volume 2, pages 1079 –1086, jul 1992.

[MC08] Leandro Marcolino and Luiz Chaimowicz. Experiments in the coordination of large groups of robots. In Gerson Zaverucha and Augusto da Costa, editors, *Advances in Artificial Intelligence - SBIA 2008*, volume 5249 of *Lecture Notes in Computer Science*, pages 268–277. Springer Berlin / Heidelberg, 2008.

[ME10] M Mtshali and A. Engelbrecht. Robotic architectures. *Defence Science Journal*, 60(1):15–22, 2010.

[MG04] Ryan Morlok and Maria Gini. Dispersing robots in an unknown environment. In *in 7th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, pages 253–262, 2004.

[MPAC12] Alessandro Marino, LynneE. Parker, Gianluca Antonelli, and Fabrizio Caccavale. A decentralized architecture for multi-robot systems based on the null-space-behavioral control with application to multi-robot border patrolling. *Journal of Intelligent & Robotic Systems*, pages 1–22, 2012.

[MSC11] Christoph Moeslinger, Thomas Schmickl, and Karl Crailsheim. A minimalist flocking algorithm for swarm robots. In *Proceedings of the 10th European conference on Advances in artificial life: Darwin meets von Neumann - Volume Part II*,

ECAL'09, pages 375–382, Berlin, Heidelberg, 2011. Springer-Verlag.

[Mur00] R. Murphy. *An introduction to AI robotics*. The MIT Press, 2000.

[NGMMH08] Iñaki Navarro, Alvaro Gutierrez, Fernando Matia, and Felix Monasterio-Huelin. An approach to flocking of robots using minimal local sensing and common orientation. In Emilio Corchado, Ajith Abraham, and Witold Pedrycz, editors, *Hybrid Artificial Intelligence Systems*, volume 5271 of *Lecture Notes in Computer Science*, pages 616–624. Springer Berlin / Heidelberg, 2008.

[NTMNM11] D. Nakhaeinia, S.H. Tang, S.B. Mohd Noor, and O. Motlagh. A review of control architectures for autonomous navigation of mobile robots. *International Journal of the Physical Sciences*, 6(2):169–174, 2011.

[OC03] Anders Oreback and HenrikI. Christensen. Evaluation of architectures for mobile robotics. *Autonomous Robots*, 14(1):33–49, 2003.

[oRR12] United States. National Ocean Service. Office of Response and Restoration. General noaa operational modeling environment (gnome) technical documentation. *U.S. Dept. of Commerce, National Oceanic and Atmospheric Administration, National Ocean Service, Office of Response & Restoration*, 2012.

[Pap08] White Paper. Wireless mesh networking. zigbee vs digimesh, 2008.

[Par98] L.E. Parker. Alliance: an architecture for fault tolerant multirobot cooperation. *Robotics and Automation, IEEE Transactions on*, 14(2):220–240, 1998.

[PBW04] Michael Pirker, Michael Berger, and Michael Watzke. An approach for fipa agent service discovery. In *in Mobile Ad Hoc Environments , Workshop on Agents for Ubiquitous Computing., Ubiagent04*, 2004.

[PEH05] D. Payton, R. Estkowski, and M. Howrd. Pheromone robotics and the logic of virtual pheromones. *Swarm Robotics. LNCS*, 3342:45–57, 2005.

[Pia99] Maurizio Piaggio. Heir - a non-hierarchical hybrid architecture for intelligent robots. In Jorg P. Muller, Anand S. Rao, and Munindar P. Singh, editors, *Intelligent Agents V: Agents Theories, Architectures, and Languages*, volume 1555 of *Lecture Notes in Computer Science*, pages 243–259. Springer Berlin Heidelberg, 1999.

[PPS+08] J. L. Posadas, J. L. Poza, J. E. Simo, G. Benet, and F. Blanes. Agent-based distributed architecture for mobile robot control. *Engineering Applications of Artificial Intelligence*, 21(6):805–823, 2008.

[PR07] Anies Hannawati Purnamadjaja and R. Andrew Russell. Guiding robots'behaviors using pheromone communication. *Auton. Robots*, 23(2):113–130, 2007.

[PY09] J.L. Posadas Yague, J.L.; Poza Lujan. Revisión de las arquitecturas de control distribuido. Technical report, Universidad Politécnica de Valencia, 2009.

[Rey87] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.*, 21(4):25–34, 1987.

[Sah05] E. Sahin. Swarm robotics: From sources of inspiration to domains of application. *Swarm Robotics. LNCS*, 3342:10–20, 2005.

[SAPR11] M. Sempere, F. Aznar, M. Pujol, and R. Rizo. Swarm system for resource location with non explicitly connected agents. *International Journal of Artificial Intelligence*, 6:203–214, 2011.

[SBS07] Onur Soysal, Erkin Bahceci, and Sahin. Aggregation in swarm robotic systems: Evolution and probabilistic control. *Turkish Journal of Electrical Engineering & Computer Sciences*, 15(2):199–225, 2007.

[SFT+11] A Stranieri, E Ferrante, A E Turgut, V Trianni, C Pinciroli, M Birattari, and M Dorigo. Self-organized flocking with a heterogeneous mobile robot swarm. *Advances in Artificial Life ECAL 2011*, 33(1):789–796, 2011.

[SGBT08] Erol Sahin, Sertan Girgin, Levent Bayindir, and Ali Emre Turgut. Swarm robotics. In Christian Blum and Daniel Merkle, editors, *Swarm Intelligence*, Natural Computing Series, pages 87–100. Springer Berlin Heidelberg, 2008.

[SH08] M. Siebold and J. Hereford. Easily scalable algorithms for dispersing autonomous robots. In *Southeastcon, 2008. IEEE*, pages 545 –550, april 2008.

[Sim94] R.G. Simmons. Structured control for autonomous robots. *Robotics and Automation, IEEE Transactions on*, 10(1):34–43, feb 1994.

[SMC06] Thomas Schmickl, Christoph Möslinger, and Karl Crailsheim. Collective perception in a robot swarm. In *in Swarm Robotics - Second SAB 2006 International Workshop*, page 2007, 2006.

[SS07] Onur Soysal and Erol Sahin. A macroscopic model for self-organized aggregation in swarm robotic systems. In Erol Sahin, William Spears, and Alan Winfield, editors, *Swarm Robotics*, volume 4433 of *Lecture Notes in Computer Science*, pages 27–42. Springer Berlin / Heidelberg, 2007.

[SSB+05] J. Seyfried, M. Szymansky, N. Bender, R. Estaña, M. Thiel, and H. Worn. The i-swarm project: Intelligent small world autonomous robots for micro-manipulation. *Swarm Robotics. LNCS*, pages 70–83, 2005.

[TCGS08a] Ali E. Tuci, E.rgut, Hande Celikkanat, Fatih Gokce, and Erol Sahin. Self-organized flocking with a mobile robot swarm. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 1*, AAMAS '08, pages 39–46, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[TCGS08b] Ali E. Turgut, Hande Celikkanat, Fatih Gokce, and Erol Sahin. Self-organized flocking in mobile robot swarms. *Swarm Intelligence*, 2(2-4):97–120, 2008.

[UTS07] E. Ugur, A.E. Turgut, and E. Sahin. Dispersion of a swarm of robots based on realistic wireless intensity signals. In *Computer and information sciences, 2007. iscis 2007. 22nd international symposium on*, pages 1 –6, nov. 2007.

[VDS04] H. Van Dyke and Brueckner S.S. Engineering swarming systems. *Methodologies and Software Engineering for Agent Systems*, 2004.

[Wei99] Gerhard Weiss, editor. *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT Press, Cambridge, MA, USA, 1999.

[WJ95] Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10:115–152, 1995.

[WLGW05] Lin Wang, Hon Li, Dhrubajyoti Goswami, and Zunce Wei. A fault-tolerant multi-agent development framework. In Jiannong Cao, Laurence Yang, Minyi Guo, and Francis Lau, editors, *Parallel and Distributed Processing and Applications*, volume 3358 of *Lecture Notes in Computer Science*, pages 126–135. Springer Berlin / Heidelberg, 2005.

[WLTW10] Hongxing Wei, Dezhong Li, Jindong Tan, and Tianmiao Wang. The distributed control and experiments of directional self-assembly for modular swarm robots. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4169 –4174, oct. 2010.

[Woj03] Katarzyna Wojtaszek. *Application of transport model for building contingency maps of oil spills on the North Sea*. PhD thesis, MS Thesis, Delft University of Technology, 71 pp., 2003.

[Woo02] Michael Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, 2002.

[WZLZ10] Xiangke Wang, Hui Zhang, Huimin Lu, and Zhiqiang Zheng. A new triple-based multi-robot system architecture and application in soccer robots. In Honghai Liu, Han Ding, Zhenhua Xiong, and Xiangyang Zhu, editors, *Intelligent Robotics and Applications*, volume 6425 of *Lecture Notes in Computer Science*, pages 105–115. Springer Berlin Heidelberg, 2010.

[XC08] WangBao Xu and XueBo Chen. Artificial moment method for swarm robot formation control. *Science in China Series F: Information Sciences*, 51(10):1521–1531, 2008.

[XD05] Peng Xu and Ralph Deters. Fault-management for multi-agent systems. In *Proceedings of the The 2005 Symposium on Applications and the Internet*, SAINT '05, pages 287–293, Washington, DC, USA, 2005. IEEE Computer Society.

[XWM11] Ma Jiachen Xie Wei and Yang Mingli. *Research on Building Mechanism of System for Intelligent Service Mobile Robot*. Dr. Janusz Bedkowski, 2011.

[YB02] H. Yavuz and A. Bradshaw. A new conceptual approach to the design of hybrid control architecture for autonomous mobile robots. *Journal of Intelligent and Robotic Systems*, 34(1):1–26, 2002.

[YLC+06] Kwang Hyun Yoo, Yuro Lee, Hun Jin Choi, Bo Hyun Yoo, and Dong Hwan Kim. Swarm robotics: Self assembly, physical configuration, and its control. In *SICE-ICASE, 2006. International Joint Conference*, pages 4276–4279, 2006.

[YT07] Yongming Yang and Yantao Tian. Swarm robots aggregation formation control inspired by fish school. In *Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on*, pages 805 –809, dec. 2007.