Universitat de les Illes Balears
Departament de Ciències Matemàtiques i Informàtica

Tesis Doctoral

Nuevas metodologías para la asignación de tareas y formación de coaliciones en sistemas multi-robot

---

por

José Guerrero Sastre

Director: Dr. Gabriel Oliver Codina

Memoria para la obtención del grado de
DOCTOR EN INFORMÁTICA

Palma de Mallorca, Enero 2011

Per als meus pares

Resumen

Los sistemas formados por múltiples robots móviles, también conocidos como sistemas multi-robot, permiten llevar a cabo tareas que con un único robot serían imposibles de realizar o requerirían mucho más tiempo. Además, presentan otras ventajas como una mayor robustez y flexibilidad. Para poder garantizar todas estas ventajas, se han de abordar múltiples problemas, muchos de los cuales son, a día de hoy, objeto de numerosos trabajos de investigación. De entre todos estos problemas, esta tesis se centra en la asignación de tareas (*Multi-Robot Task Allocation*-MRTA), esto es, en los métodos que permiten decidir qué robot o conjunto de robots ha de ejecutar cada tarea. Este trabajo analiza la idoneidad de dos de los principales métodos de MRTA, los basados en *swarm intelligence* y los basados en mecanismos de subastas, para tratar tareas con restricciones temporales. La principal característica de estas tareas es que cada una de ellas se ha de ejecutar antes de un determinado instante de tiempo, comúnmente referido como *deadline*. Se pondrá de manifiesto que ambos tipos de mecanismos presentan carencias para tratar tareas con *deadlines*. Estas carencias son especialmente importantes cuando dos o más robots, formando una coalición, pueden ser asignados a una misma tarea. Uno de los aspectos a los que esta tesis dedica mayor atención es la predicción del tiempo de ejecución, que depende, entre otros factores, de la interferencia física entre robots. La interferencia es el fenómeno producido cuando dos o más robots quieren acceder a una misma posición al mismo tiempo. Diversos autores han descrito y tratado el fenómeno de la interferencia, pero, hasta ahora, no se ha tenido en cuenta su efecto en los mecanismos de MRTA basados en subastas. Por su parte, existen diferentes tipos de técnicas que, utilizadas junto a métodos de

swarm, permiten reducir los efectos nocivos de la interferencia. Sin embargo, estas técnicas no cuantifican el nivel de interferencia ni predicen qué efectos va a tener sobre el futuro desarrollo de una tarea.

Esta tesis presenta el primer mecanismo de subastas para la creación de coaliciones que tiene en cuenta de manera explícita la interferencia física entre robots. Para ello, entre otras aportaciones, se ha desarrollado un modelo de predicción del tiempo de ejecución, basado en *Support Vector Regression* (SVR). Además, se han utilizado conceptos provenientes de los métodos de *swarm* y se ha propuesto un nuevo paradigma llamado subasta doble. Todas estas contribuciones permiten mejorar los resultados respecto a los sistemas clásicos, sobre todo en entornos con restricciones temporales. Además, este trabajo propone un nuevo método de *swarm*, llamado *swarm* pseudo-aleatorio (*Pseudo-random Swarm*-PSW), y un nuevo método de subastas llamado *Earliest Deadline First Best Pair* (EDFBP) que permite asignar un único robot por tarea. Los resultados experimentales muestran como ambos métodos permiten incrementar el número de tareas finalizadas antes de su límite temporal, con respecto a los mecanismos actuales de MRTA.

Agradecimientos

Quisiera aprovechar estas líneas para agradecer a todas aquellas personas sin cuyo apoyo y ayuda esta tesis no se hubiese podido llevar a cabo.

En primer lugar, mi más sincero agradecimiento a mi director de tesis, Gabriel Oliver, por haberme dado la oportunidad de realizar este trabajo. Gracias a sus consejos, a su ayuda, a su confianza y al tiempo dedicado esta tesis ha podido ver la luz.

También quisiera agradecer su ayuda al resto de miembros del grupo de Sistemas, Robótica y Visión (SRV): a Antoni Burguera, Javier Antich y Guillermo Rodriguez-Navas con quienes inicié esta aventura. A Manuel Barranco por su apoyo moral y técnico, además de por haber aguantado tantos gélidos días en nuestro despacho. Y por supuesto al resto de miembros del grupo: Alberto Ortiz, Julián Proenza, Yolanda González, Francesc Bonin y Bartomlomé Garau. Además de a todos aquellos otros que se han ido uniendo al grupo a lo largo de estos últimos años. También tengo que agradecer a Oscar Valero sus valiosas indicaciones respecto a los formalismos matemáticos utilizados, así como a Ginés Valverde y Mateu Hernández la ayuda técnica prestada. De igual manera, dar las gracias al resto de miembros del Departamento de Ciencias Matemáticas e Informática que me han apoyado.

Finalmente, agradezco de manera muy especial a mi familia y sobre todo a mis padres, Andrés y María, su paciencia e incondicional apoyo durante todos estos años. A ellos va dedicada esta tesis.

Contenidos

Lista de figuras

Lista de tablas

Lista de algoritmos

Capítulo 1

Introducción

El presente trabajo se centra en la asignación de tareas en sistemas multi-robot, es decir, se profundiza en el estudio del reparto del trabajo a realizar por un colectivo de robots móviles autónomos, especialmente cuando estas tareas han de ser ejecutadas por una coalición de robots. En este capítulo se explican las principales características que presentan los sistemas formados por múltiples robots, poniendo de manifiesto sus ventajas e inconvenientes. Además, se presentan los objetivos de esta tesis y sus principales contribuciones.

1.1. Motivación

Los sistemas formados por múltiples robots móviles autónomos que tienen un objetivo común, a partir de ahora llamados sistemas multi-robot, presentan múltiples ventajas respecto a los sistemas con un único robot. Algunas de estas ventajas son:

- Robustez: tener múltiples robots permite incrementar la tolerancia del sistema ante posibles errores individuales. Si en un sistema multi-robot uno de sus miembros se avería, el resto de robots operativos puede seguir ejecutando la tarea.
- Incremento de la capacidad de trabajo: la unión de varios robots puede permitir

llevar a cabo tareas que con uno solo serían muy difíciles de ejecutar. Un ejemplo claro de este tipo de situaciones es el transporte de objetos de gran tamaño o peso. Debido al peso o longitud de un determinado objeto puede ser necesario que varios robots deban cooperar para su transporte.

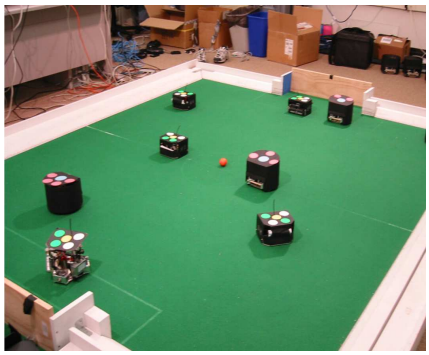
- Rapidez: los sistemas multi-robot permiten llevar a cabo una misma tarea de manera paralela en múltiples lugares y, de esta manera, reducir su tiempo de ejecución. La tarea de limpieza de superficies es un claro ejemplo de esta característica, ya que cada robot puede limpiar un área diferente de manera simultánea.

- Reducción de costes: tener múltiples robots simples suele resultar más económico que tener un único robot complejo.

- Flexibilidad: los sistemas multi-robot pueden reconfigurarse y adaptarse muy fácilmente a múltiples tipos de entorno y tareas. Al tener múltiples robots, se incrementan también las posibles maneras de combinar las habilidades de cada uno de ellos para llevar a cabo una misma tarea.

Como consecuencia de todas estas ventajas, los sistemas multi-robot pueden ayudar a la ejecución de una amplia variedad de tareas. Estas tareas se pueden dividir en tareas débilmente acopladas y tareas fuertemente acopladas, en función del grado necesario de cooperación entre robots para llevarlas a cabo. Las tareas débilmente acopladas no necesitan que los robots colaboren entre ellos, aunque dicha cooperación puede mejorar el rendimiento del sistema, por ejemplo reduciendo el tiempo de ejecución o disminuyendo la energía total consumida. El ejemplo más claro de este tipo de tareas es la recolección de objetos o *foraging*. La tarea de *foraging* consiste en la localización de una serie de objetos por parte de un conjunto de robots y su posterior transporte hasta un punto de descarga. Cada objeto puede ser transportado por un único robot. Otro ejemplo de este tipo de tareas es la limpieza de superficies, en la que cada robot ha de limpiar un área determinada. En cambio, en las tareas fuertemente acopladas es imprescindible que los robots colaboren entre ellos para llevarlas a cabo. Un ejemplo de este tipo de tareas es la de 'empujar una caja' (*box pushing*), en la que

varios robots han de empujar un objeto voluminoso hasta una determinada posición. En este caso, los robots han de coordinar sus movimientos para mover el objeto en la dirección correcta, por tanto, intentar llevar a cabo esta tarea con un único robot o sin coordinación entre ellos sería una misión prácticamente imposible. Otro ejemplo de tareas fuertemente acopladas son las llevadas a cabo en la RoboCup Soccer [9] y en la RoboCup Rescue [82]. En la competición RoboCup Soccer dos equipos de robots se enfrentan en un partido de fútbol, tal como se puede observar en la figura 1.1(a). En la RoboCup Rescue se ha de coordinar la actuación de varios equipos de emergencia para afrontar una catástrofe. La figura 1.1(b) muestra una captura de pantalla del simulador utilizado durante esta competición.



(a) Imagen de la liga de robots, RoboCup Soccer.

(b) Captura de pantalla del simulador utilizado durante la competición RoboCup Rescue.

Figura 1.1: Ejemplos de tareas fuertemente acopladas.

Las ventajas de los sistemas multi-robot únicamente se podrán poner de manifiesto si se implementan mecanismos de cooperación apropiados. Para ello se han de resolver, entre otros, tres problemas básicos:

- Planificación: consiste en dividir la tarea inicial, que puede llegar a ser muy compleja, en una serie de subtareas más simples que puedan ser ejecutadas por un único robot o subconjunto de ellos. Además, durante la planificación se decide en qué orden el conjunto de robots ha de llevar a cabo estas tareas, es decir, se genera un plan.

- Asignación de tareas (*task allocation*): durante la asignación de tareas se decidirá qué robot o conjunto de robots ha de ejecutar cada una de las tareas. A partir de ahora este problema se denominará MRTA (*Multi-Robot Task Allocation*). El problema MRTA es especialmente complejo cuando varios robots pueden ser asignados a una misma tarea, en este caso se dirá que los robots forman coaliciones. Los mecanismos de asignación de tareas más comunes hoy en día son los basados en *swarm* y los métodos de subastas. En los métodos de *swarm* no existen mecanismos explícitos de coordinación ni se necesitan protocolos de comunicación entre los miembros del sistema, sino que el comportamiento global surge de la interacción entre comportamientos muy simples implementados en cada robot. Se trata de mecanismos inspirados en el comportamiento de colonias de insectos como las hormigas o las abejas. Los sistemas basados en *swarm* se pueden dividir a su vez en no deterministas, en los que cada robot tiene una determinada probabilidad de ejecutar una tarea, y deterministas, en los que cada robot siempre selecciona la mejor tarea desde su punto de vista. En cambio, en los mecanismos de subastas es necesario implementar algoritmos y protocolos de comunicación que permitan a los robots seleccionar la tarea a ejecutar. Gracias a la comunicación entre los robots, los métodos basados en subastas pueden llevar a cabo tareas más complejas que las que pueden realizar los métodos de *swarm*. Además, las soluciones obtenidas suelen ser mejores en términos de tareas finalizadas o consumo de energía con respecto a las soluciones alcanzadas mediante métodos de *swarm*. En el capítulo 2 se explicarán más en detalle ambos mecanismos.

- Coordinación del movimiento: cada robot ha de decidir qué movimientos realizar para llevar la tarea que tiene asignada asignada de la mejor manera posible sin entorpecer la ejecución del resto de robots.

Tal como han demostrado otros autores, como por ejemplo Zlot y Stentz en [141], a pesar de que estos problemas puedan plantearse de forma independiente unos de otros, están estrechamente interrelacionados. Además, solucionarlos puede resultar

muy complejo, entre otras razones, debido a que cada robot puede tener características diferentes, el entorno suele ser muy cambiante y dinámico y puede haber errores en las comunicaciones entre los robots o en la lectura de sus sensores.

1.2. Objetivos

Esta tesis se centrará en el estudio y desarrollo de nuevos métodos de MRTA en general y, de forma particular, en aquellos que abordan tareas con restricciones temporales. Este tipo de tareas se han de llevar a cabo antes de un determinado instante de tiempo (*deadline*), en caso contrario, la tarea no se podrá llevar a cabo, o bien su utilidad decrecerá con el tiempo. De entre las diferentes estrategias existentes para afrontar el problema de asignación de tareas, este trabajo se basa en los mecanismos más utilizados actualmente: los basados en *swarm* y los métodos de subastas. Así, esta tesis intentará, en primer lugar, dar respuesta a la siguiente pregunta:

- ¿Pueden los sistemas de asignación de tareas basados en *swarm* o subastas afrontar tareas con restricciones temporales?.

Tal como se demostrará, los sistemas actuales basados en estos dos paradigmas muestran importantes carencias para llevar a cabo tareas con restricciones temporales. De todos los problemas mostrados por ambos mecanismos, el más importante, desde nuestro punto de vista, es el de la predicción del tiempo de ejecución de una tarea, cuando ésta ha de ser llevada a cabo por una coalición de robots. El tiempo de ejecución depende de múltiples factores: tipo de robots utilizados, características de la tarea, características del entorno, etc. Esta tesis se centrará en el estudio y modelización de uno de estos factores: la interferencia física entre robots, producida cuando dos o más de ellos intentan acceder a una misma posición al mismo tiempo. Es bien conocido que la interferencia física afecta de manera crítica al funcionamiento del sistema, tal como ya han demostrado diferentes trabajos [109, 92, 91]. A pesar de ello, actualmente no existe ningún mecanismo de subastas que incluya los efectos de

la interferencia física en el proceso de asignación de robots a las distintas tareas. Por tanto, otro objetivo fundamental de esta tesis será el siguiente:

- Desarrollo de un mecanismo de predicción del tiempo de ejecución que tenga en cuenta la interferencia física entre robots y que se pueda incluir en un proceso de subastas para la creación de coaliciones.

Finalmente, el tercer objetivo de este trabajo es la realización de un análisis comparativo entre los mecanismos de subasta y *swarm*, de nuevo en entornos con restricciones temporales. Este estudio amplia el trabajo realizado por otros autores [81], donde no se utilizan simulaciones realistas ni hay *deadlines* asociados a las tareas.

Para validar los algoritmos y métodos propuestos en esta tesis, se han utilizado mayormente tareas de recogida, donde varios robots, formando una coalición, pueden colaborar en el transporte de un mismo objeto. Los objetos pueden tener restricciones temporales, de manera que han de ser recogidos o transportados antes de un determinado instante de tiempo. Este tipo de tareas ha sido ya utilizado en otros trabajos, aunque sin restricciones temporales. Además, gracias a su simplicidad, este entorno experimental permite estudiar la interferencia física entre robots de manera aislada e independiente de otro tipo de factores, que quedan fuera del alcance de este trabajo. Tampoco se han tenido en cuenta aspectos relacionados con la planificación temporal de las tareas. No obstante, los mecanismos de predicción de los tiempos de ejecución, que aquí se propondrán, podrán ser utilizados en trabajos futuros para realizar esta planificación.

1.3. Contribuciones

Las principales aportaciones de esta tesis en el ámbito de la asignación de tareas en sistemas multi-robot son las siguientes:

- Mecanismo para la predicción del tiempo de ejecución en tareas con restricciones temporales. Se ha propuesto un nuevo mecanismo basado en *Support Vector*

Regression (SVR) capaz de predecir el tiempo que necesitará un conjunto de robots para finalizar una tarea, teniendo en cuenta la interferencia física entre ellos. Mediante este sistema, se puede determinar cuántos robots serán necesarios para llevar a cabo una tarea y, por tanto, si en el sistema hay recursos suficiente para finalizarla antes de su *deadline*. Se trata del primer mecanismo de este tipo capaz de determinar el tamaño de una coalición de robots en tiempo de ejecución. Este trabajo ha sido publicado en [70, 69] y se explicará en el capítulo 5 de esta memoria.

- Mecanismo de subastas que tiene en cuenta la interferencia física: esta tesis propone el primer mecanismo basado en subastas que tiene en cuenta la interferencia física en problemas que admiten múltiples robots en cada tarea. Este mecanismo utiliza conceptos heredados de los métodos basados en *swarm* para determinar, en tiempo de ejecución, el número de robots necesarios para llevar a cabo cada tarea. El método propuesto se puede integrar con una nueva estrategia de coordinación del movimiento, llamada *lane keeping*, basada en campos de potencial. Además, en el momento de su publicación se trató de uno de los primeros mecanismos de subastas que permite múltiples robots por tarea. Los resultados de estos trabajos han sido publicados en [66, 65, 64, 63] y se explicarán en el capítulo 4 de esta memoria.

- Nuevo mecanismo para la asignación de tareas en entornos con restricciones temporales (subasta doble): se ha propuesto un nuevo mecanismo de subastas, llamado 'subasta doble', que utiliza la predicción del tiempo de ejecución, obtenido a partir del modelo SVR, para determinar la cantidad de robots necesarios para finalizar una tarea antes de su límite temporal. Se trata del primer mecanismo de subastas capaz de calcular en tiempo de ejecución el número de robots necesarios para realizar una tarea con restricciones temporales en función de la interferencia física. Además, este nuevo sistema de predicción del tiempo de ejecución, evita que se tengan que utilizar mecanismos para monitorizar el progreso de la tarea, mecanismos que pueden llegar a ser muy complejos y costosos. Los

resultados de este trabajo han sido publicados en [70, 69, 68, 67] y se explicarán en el capítulo 5 de esta memoria.

- Análisis comparativo entre mecanismos de *swarm* y subastas: esta tesis realiza por primera vez un análisis comparativo entre los principales algoritmos basados en *swarm* y los sistemas de subastas en entornos con restricciones temporales. De esta manera, se amplían los trabajos realizados por autores, como Kalra y Martinoli en [81], con tareas sin *deadlines*. Como resultado de este estudio se ha propuesto un nuevo algoritmo de subastas llamado *Earliest Deadline First Best Pair* (EDFBP) que reduce el tiempo de ejecución respecto a otros sistemas ya existentes sin que disminuya el número de tareas finalizadas antes de su *deadline*. Estos resultados han sido parcialmente publicados en [71] y se explicarán en el capítulo 3.

- Algoritmos de *swarm* pseudo-aleatorio: se ha propuesto una nueva estrategia basada en *swarm*, híbrida entre un sistema determinista y uno no determinista, llamada *Pseudo-Random Swarm* (PSW). Los experimentos realizados muestran como las diferentes variantes propuestas del sistema PSW, *Pseudo-random Swarm Robots* (PSW-R) y *Pseudo-random Swarm Distance* (PSW-D), mejoran los resultados de los actuales métodos de *swarm*. En el capítulo 3 se explicarán más detalladamente estos métodos.

1.4. Estructura del documento

El resto de esta memoria se ha estructurado en 5 capítulos, con el siguiente contenido:

- ***Capítulo 2**. Asignación de Tareas en Sistemas multi-robot: definición y trabajos relacionados*: se revisarán los mecanismos actuales de asignación de tareas relacionados con los métodos propuestos en esta tesis. También se analizará la

complejidad del problema MRTA y se verá como, en la mayoría de los casos, es de tipo NP-hard.

- ***Capítulo 3***. *Configuraciones con un único robot por tarea*: en este capítulo se presentarán los métodos que permiten asignar un único robot por tarea, tanto en entornos con restricciones temporales como sin ellas. A partir de los resultados experimentales se han propuesto tres nuevos métodos: dos basados en *swarm* (PSW-R y PSW-D) y uno en subastas (EDFBP).

- ***Capítulo 4***. *Formación de coaliciones en entornos sin restricciones temporales*: se expondrá el sistema de subastas utilizado para asignar múltiples robots a una misma tarea cuando no existen restricciones temporales. También se presentará el método de coordinación de movimiento que se integra con el mecanismo de asignación de tareas.

- ***Capítulo 5***. *Formación de coaliciones en entornos con restricciones temporales*: se presentará el método que permite asignar múltiples robots a una misma tarea cuando ésta tiene restricciones temporales. También se explicará en este capítulo el modelo de interferencia usado para predecir el tiempo que necesitará un conjunto de robots para finalizar una tarea.

- ***Capítulo 6***. *Conclusiones y trabajo futuro*: en este capítulo se revisarán las principales contribuciones de esta tesis y se explicarán cuales son las posible líneas futuras de trabajo.

Capítulo 2

Asignación de Tareas en Sistemas multi-robot: definición y trabajos relacionados

En este capítulo se exponen los aspectos fundamentales del problema de asignación de tareas en sistemas multi-robot presentes en la literatura actual. Se hará especial hincapié en la complejidad del problema relacionándolo con otras áreas de conocimiento como la de sistemas multi-agente o la de planificación de tareas. Las definiciones aquí expuestas se utilizarán a lo largo de esta tesis para estudiar la complejidad de los problemas propuestos en los siguientes capítulos. Como se podrá observar, en la mayoría de casos los problemas de asignación de tareas para la creación de coaliciones, al igual que la mayoría de problemas propuestos en este trabajo, son de tipo NP-hard. Esta gran complejidad hace necesario el uso de heurísticas para obtener una solución lo más próxima posible a la óptima en un tiempo razonablemente corto.

Se hará un repaso de los principales sistemas de asignación de tareas que actualmente se utilizan en sistemas multi-robot. Se dividirán estos métodos en 3 grandes grupos: centralizados, basados en *swarm intelligence* y basados en subastas. Se prestará especial atención a los dos últimos paradigmas y se explicarán las notables limitaciones que presentan para tratar tareas con *deadlines*, cuando éstas han de ser

ejecutadas por una coalición de robots. Los algoritmos presentados en esta memoria mejoran los actuales mecanismos de *swarm* y subastas logrando subsanar dichas limitaciones.

En la literatura actual se pueden encontrar varia taxonomías que permiten clasificar los sistemas multi-robot en función de múltiples parámetros [54, 44, 42]. De todas ellas, una de las más utilizadas es la propuesta por Gerkey [54] según la cual los problemas que han de solucionar los algoritmos MRTA se clasifican según tres criterios básicos:

- *Single-task robots* (ST) o *multi-task robots* (MT): en un problema ST los robots únicamente pueden ejecutar una tarea simultáneamente. En cambio, en los problemas MT un mismo robot puede estar ejecutando varias tareas al mismo tiempo.

- *Single-robot tasks* (SR) o *multi-robot tasks* (MR): si varios robots pueden colaborar en la ejecución de una misma tarea, el sistema será considerado de tipo MR. En caso contrario, si tan sólo se puede asignar un robot por tarea, el problema será de tipo SR.

- *Instantaneous assignment* (IA) o *time-extended assignment* (TE): en un problema de tipo IA, la asignación de tareas a un robot se realiza sin tener en cuenta posibles decisiones futuras, es decir no se realiza planificación. En cambio, en los sistemas TE no tan sólo se tiene en cuenta la tarea asignada a un robot en un determinado instante de tiempo, sino que también se consideran las futuras asignaciones.

A lo largo de esta memoria se utilizará la taxonomía de Gerkey para facilitar la descripción de los diferentes algoritmos que se irán presentando. A pesar de ello, cabe mencionar que esta taxonomía no permite clasificar totalmente un sistema siguiendo únicamente estos tres criterios. Por ejemplo, los algoritmos presentados en el capítulo 3 corresponderían a la clase ST-SR-IA, ya que en la asignación de tareas al robot no se tiene en cuenta futuras tareas, ni se permite la reasignación de las ya otorgadas.

En cambio, sí se permite una planificación local a cada robot que condiciona en cierta forma futuras decisiones. Desde este punto de vista se podría considerar un sistema ST-SR-TE. De igual forma, los algoritmos propuestos en el capítulo 5 son de tipo ST-MR-IA, pero al incluir también un modelo que estima el tiempo de ejecución con restricciones temporales, también presenta características de los algoritmos ST-MR-TE.

2.1. Definición del Problema

En esta sección se repasan las principales definiciones del problema de asignación de tareas dadas por diferentes autores. Además se relacionará el problema MRTA con otros problemas estudiados en campos tales como los sistemas multi-agente o la planificación de procesos en sistemas multi-procesador. Al ser la gran mayoría de estos problemas de tipo NP-hard, no existe actualmente ningún algoritmo con un orden de complejidad acotado por un polinomio que proporcione la solución óptima.

2.1.1. Un único robot por tarea: problemas SR

A continuación se mostrarán algunas definiciones utilizadas para abordar los problemas de tipo SR, es decir aquellos algoritmos que únicamente permiten asignar un robot por tarea. Tal como Gerkey demuestra en [52], los algoritmos ST-SR-IA se pueden describir en términos del clásico problema de asignación óptima (*Optimal Assignment Problem*-OAP). El problema OAP se define de la siguiente manera: dados n agentes (robots) y m tareas a realizar, cada agente puede ser asignado a una única tarea y cada tarea requiere un único agente. Para cada pareja agente-tarea se define un valor que predice el rendimiento del agente en la tarea, es decir, indica su utilidad. El objetivo es asignar las tareas a los agentes, de manera que el rendimiento o utilidad total del sistema sea la máxima posible, esto es, se requiere maximizar el valor de U:

$$U = \sum_{1 \leq i \leq n} \sum_{1 \leq j \leq m} \alpha_{ij} U_{ij} w_j \tag{2.1}$$

donde α_{ij} valdrá 1 si la tarea i se asigna al agente j y 0 en caso contrario, U_{ij} es la utilidad correspondiente a asignar la tarea i al agente j y w_j indica la importancia o peso de la tarea j, de manera que tareas más prioritarias tendrán un valor de w_j mayor. El método Húngaro [86] propuesto por Kuhn permite obtener una solución óptima al problema OAP en un tiempo $O(nm^2)$ mediante un algoritmo centralizado que utiliza programación dinámica. Se ha de notar que, si las utilidades están interrelacionadas, es decir, que la utilidad de un agente para una tarea depende de las asignaciones realizadas anteriormente a ese mismo agente, entonces el problema puede convertirse en uno de tipo ST-SR-TE y ser de tipo NP-hard. En esta tesis los problemas de tipo SR se exponen en el capítulo 3 pero, al contrario de lo que sucede en la definición inicial del problema OAP, se han añadido restricciones temporales a cada una de las tareas. Además, la planificación, que en algunos casos se utiliza para decidir en qué orden se deben ejecutar las tareas, los convierte en problemas de tipo NP.

2.1.2. Múltiples robots por tarea: problemas MR

En este apartado se explicarán las definiciones existentes actualmente para los problemas de tipo MR, donde los robots pueden formar coaliciones para llevar a cabo una misma tarea. En este caso se puede utilizar la definición del problema de partición de conjuntos (*Set Partitioning Problem*-SPP) [5] de la siguiente manera: dado un conjunto finito E, una familia F de posibles subconjuntos de E y una función de utilidad $U : F \rightarrow \Re^+$, se ha de encontrar una familia X de elementos de F que tenga la máxima utilidad y que sea una partición de E. Este tipo de problemas son NP-hard, tal como demuestran Korte y Vygen en [85]. El problema de cobertura de conjuntos (*Set Covering Problem*-SCP) es similar al SPP pero es suficiente que X sea una cobertura de E, siendo este un problema NP-completo [49]. Ambos problemas se pueden asimilar al de creación de coaliciones de robots, donde E es el conjunto de robots, F es el conjunto de posibles parejas coalición de robots-tarea y U es la función de utilidad asociada a cada una de las asignaciones [127]. Si se quiere introducir limitaciones temporales, se deberían tratar como una condición más para

poder considerar válida a una familia F de subconjuntos de E. La definición de SCP permite que un mismo robot pueda formar parte de varias coaliciones al mismo tiempo, por tanto se puede asemejar a problemas de tipo MT. Por otra parte, la definición SPP es más restrictiva ya que que cada robot sólo puede formar parte de una coalición simultáneamente y, por tanto, este tipo de problemas se clasifican como de tipo ST. En [115] Service y Adams demuestran que el problema de creación de coaliciones, utilizando las definiciones de SPP o de SCP, tanto en agentes como en robots, no tan sólo es NP-hard, sino que, en general, no puede ser aproximado por un factor $O(m^{1-\varepsilon})$, $\forall \varepsilon > 0$, donde m es el número de tareas. Esto significa que, a no ser que $P = NP$, la solución obtenida mediante un algoritmo con una complejidad acotada por un polinomio siempre será $O(m^{1-\varepsilon})$, $\varepsilon \leq 0$, peor a la solución óptima. Por tanto, como se puede observar, la complejidad de los algoritmos de coaliciones es muy elevada, lo que justifica el gran número de aproximaciones propuestas.

En [116] Shehory y Kraus proponen un algoritmo de creación de coaliciones basado en las definiciones SPP y SCP para agentes en el que se fija el número máximo de elementos (k) en cada coalición. La complejidad del algoritmo es $O(n^k m)$ donde n es el número de agentes y m el número de tareas. En [127] Vig propone un algoritmo con una complejidad $O(n^k m)$ que extiende los propuestos en [116] para sistemas multi-robot. En [115] Service y Adams presentan una mejora respecto al método propuesto por Vig, en la que la complejidad del algoritmo es $O(n^{\frac{3}{2}} m)$, aunque, en general, la solución aportada sigue sin ser óptima. Si los robots se pueden agrupar en j grupos homogéneos, de manera que cada robot forme parte de alguno de estos grupos, el algoritmo definido por Service y Adams proporciona una solución óptima al problema de generación de coaliciones con una complejidad algorítmica igual a $O(n^{2j} m)$. En este algoritmo cada tarea ha de tener un número mínimo de robots para poder empezar a ejecutarse, a partir del cual la tarea i tendrá una utilidad constante u_i. En el capítulo 4 se propondrá una tarea que se ajusta al problema SPP, en la que los robots no tienen porqué ser homogéneos y el valor de k no se conoce a priori ni está limitado. Esta tarea ha sido ampliada en el capítulo 5 para que la utilidad de

la tareas dependa, entre otros factores, del número de robots que tenga asignados, es decir el valor de u_i es función del número de robots. Por tanto, en ninguno de estos dos casos es aplicable el algoritmo de Service y Adams que proporciona la solución óptima, siendo ambos problemas de tipo NP.

2.1.3. Planificación de tareas y coaliciones: problemas ST-MR-TE

Las definiciones anteriores, basadas en SPP o SCP, no tienen en cuenta la planificación temporal de las tareas, esto es, decidir en qué orden han de ser ejecutadas. Para ello, se puede definir el problema de asignación de tareas mediante el problema *Job Shop Scheduling Problem* (JSS). En general, el problema JSS se define de la siguiente manera [123, 106, 75]: dado un conjunto de m trabajos $J = \{J_1, J_2, ..., J_m\}$ y un conjunto de n de máquinas $M = \{M_1, M_2, ..., M_n\}$, cada trabajo J_i tiene m_i operaciones asociadas $O_i = \{O_{i1}, O_{i2}, ..., O_{im_i}\}$ y cada una de ellas tiene un determinado tiempo de ejecución en cada máquina T_{ij}. El objetivo es decidir (planificar) en qué orden se van a ejecutar las tareas en las máquinas, de manera que el tiempo de ejecución sea mínimo. Además, se puede establecer un orden de precedencia entre operaciones e incluso establecer tiempos máximos de ejecución, tal como Balas et al. proponen en [6]. Como demuestran Garey et al. en [50], este tipo de problemas siguen siendo, en general, de tipo NP-hard. En el caso de un sistema multi-robot, el problema JSS consistiría en decidir el orden de ejecución de las tareas asignadas a cada robot, para minimizar el tiempo de ejecución, la cantidad de energía consumida por los robots, o cualquier otro parámetro del sistema, esto es, se clasificarían como de tipo TE, siguiendo la taxonomía de Gerkey. Por ejemplo, en [79] Jones et al. utilizan algoritmos genéticos para solucionar una variante del problema JSS en sistemas multi-robot.

Otra aproximación, también basada en modelos de planificación de tareas, es la propuesta por Dahl en [30] donde se utilizan conceptos propios de la planificación en sistemas multi-procesador para analizar la complejidad del problema de creación de coaliciones. En esta definición se incluye el concepto de dinámica del grupo, modelada

mediante una función del tiempo $g(t)$. La función $g(t)$ describe el tiempo de ejecución de una tarea en función de cómo se ha realizado la asignación de tareas en otras máquinas (robots) y de cómo ésta evoluciona a lo largo del tiempo. Es decir, el tiempo de ejecución de una tarea dependerá de la asignación y planificación de las otras. Este concepto está estrechamente relacionado con la interferencia física producida entre robots que ejecutan diferentes tareas. También en [30] se introduce el concepto de trabajos espacialmente clasificables (*sc*), donde las tareas se dividen en una serie de grupos, de manera que el tiempo de ejecución de una tarea tan sólo depende de las tareas de su mismo grupo. Las tareas con trabajos espacialmente clasificables se pueden definir como problemas de planificación de tipo NP-hard aún en el caso extremo en que cada grupo únicamente esté formado por una tarea, es decir, a pesar de que el tiempo de ejecución dependa únicamente de los robots asignados a cada una de las tareas. En el capítulo 5 se ha utilizado, en parte, el concepto de trabajos de tipo *sc* para obtener el modelo de interferencia física entre robots.

Muy relacionado con la planificación, se encuentra el problema de asignación de rutas de vehículos con ventanas temporales (*Vehicle Routing Problem with Time Windows*-VRPTW), en el que un conjunto de vehículos ha de transportar mercancías a una serie de 'clientes'. Cada vehículo tiene una determinada capacidad de carga y cada cliente demanda una cierta cantidad de estos bienes que han de ser entregados dentro de un cierto intervalo de tiempo (ventana temporal). El objetivo es encontrar la ruta más corta posible y con el menor número de vehículos que satisfaga las necesidades de los clientes, de manera que cada uno de ellos sólo necesite ser visitado por un vehículo [51]. Al igual que en la definición de JSS, este problema continúa siendo NP-hard. En [14] Beck et al. analizan las diferencias y similitudes entre el problema JSS y el VRPTW, los experimentos realizados muestran que, para problemas de definición de rutas de vehículos, los algoritmos basados en VRPTW mejoran los resultados obtenidos con los métodos basados en JSS. Por otra parte, utilizar algunas características de los algoritmos JSS, como por ejemplo restricciones en el orden de

ejecución de las tareas, permite ampliar el ámbito de aplicación del problema VRPTW. Esta definición se ajusta muy bien a un gran número de problemas reales de transporte de mercancías, que pueden ser aplicados al ámbito de la robótica [19, 118] mediante alguna de las técnicas o algoritmos que se explicarán más adelante en este mismo capítulo. Una variante del problema VRPTW es el de asignación de rutas con ventanas temporales no estrictas (*Vehicle Routing Problem with Soft Time Windows*-VRPSTW) [21, 39], en el que las restricciones temporales son más flexibles y se permite que alguna de ellas no se cumpla. El problema de creación de coaliciones planteado en el capítulo 5 de esta tesis también se ajusta al problema VRPTW donde la ventana temporal de la tarea siempre empieza en el momento de su aparición en el entorno. Además, los métodos propuestos en el capítulo 5 permiten que haya tareas con una utilidad que empiece a decrecer a partir de un determinado instante de tiempo, en este caso el problema se convierte en uno de tipo VRPSTW. Al contrario de lo que sucede en la mayoría de sistemas VRPTW, las propuestas de esta tesis no realizan explícitamente una planificación temporal, a pesar de ello, los resultados experimentales obtenidos demuestran la efectividad de los algoritmos a la hora de ejecutar tareas con restricciones temporales.

Sariel en [111] define el problema de asignación de tareas y creación de coaliciones en términos del problema RCPSP (*Resource Constrained Project Scheduling Problem*) [135], en el que se establece, no sólo un orden de precedencia entre tareas, como ya pasaba en el problema JSS, sino que también se limita a un número fijo y predeterminado los robots asignados a cada una de ellas. A pesar de que los robots son homogéneos, este problema continúa siendo NP-hard, no pudiéndose resolver directamente mediante los algoritmos basados en SPP o SCP, ya explicados anteriormente. Los métodos de creación de coaliciones propuestos en esta tesis relajan estas restricciones ya que el número de robots destinados a cada tarea y el tiempo de ejecución no necesitan ser conocidos a priori, sino que se pueden aprender y ser ajustados mientras se lleva a cabo la misión. A lo largo de este capítulo, y posteriores, se discutirá más detalladamente las diferencias de los métodos aquí propuestos con la propuesta de

Sariel.

Uno de los problemas probablemente más abordados por los sistemas multi-robot es el del viajante de comercio (*Traveling Salesman Problem-TSP*), en el que un conjunto de robots o agentes han de visitar una serie de puntos del espacio (ciudades) [88]. El problema TSP se puede considerar como una parte del problema de asignación de rutas a vehículos y, al igual que éste, también es NP-hard. A lo largo de este capítulo se detallarán algunas de las soluciones aportadas para abordar este problema en sistemas multi-robot. Al problema TSP se le pueden añadir restricciones temporales, de manera que cada uno de los puntos ha de ser visitado antes de un determinado instante de tiempo, a este tipo de problemas se les denomina $\Delta - D_L TSP$ [13] y, evidentemente, continúan siendo NP-hard. Parte de los algoritmos propuestos en el capítulo 3 de esta tesis tratan el problema $\Delta - D_L TSP$, con el objetivo de comparar cómo afectan al rendimiento del sistema respecto a los sistemas centralizados de asignación de tareas. En este caso, los robots planifican la ruta a seguir (orden de los objetivos a visitar) utilizando las tareas previamente asignadas.

Finalmente, se ha de mencionar que los problemas de tipo MT, en los que se permiten múltiples tareas asignadas al mismo tiempo a un mismo robot, normalmente no son factibles en sistemas con robots. La mayor parte de tareas a realizar por los robots, requieren que éstos estén físicamente en un determinado lugar, por tanto, en la práctica totalidad de situaciones, los robots podrán ejecutar únicamente una tarea al mismo tiempo. En el capítulo 3 se verá que los robots pueden tener varias posiciones a donde ir, pero deben pasar por ellas de manera secuencial, siendo inviable una ejecución simultánea.

2.2. Mecanismos de asignación de tareas

A continuación se exponen y comentan los principales trabajos realizados por otros autores en el campo de la asignación de tareas y que están relacionados con los métodos propuestos en esta tesis. Este apartado se centrará especialmente en los

métodos de creación de coaliciones y en aquellos que permiten tener en cuenta tareas con restricciones temporales. Para ello, se dividirán los métodos en tres grupos, en función de cómo se distribuye el proceso de asignación entre los robots: sistemas centralizados, sistemas auto-organizados (*swarm*) y sistemas distribuidos o basados en comunicación. Dentro de cada grupo se utilizará la taxonomía de Gerkey para clasificar las diferentes propuestas.

2.2.1. Sistemas centralizados

En los sistemas centralizados toda la información sobre los robots o sobre características del entorno, es enviada a un agente central que toma todas las decisiones. Normalmente en estos sistemas se utilizan, por ejemplo, mecanismos basados en programación lineal o búsquedas basadas en alguna heurística con el objetivo de obtener la solución óptima o la más próxima posible a la óptima. Un ejemplo de sistema centralizado es el propuesto por Koes et al. [84] en el que se utilizan técnicas de programación lineal para solucionar simultáneamente los problemas de planificación, asignación de tareas y planificación de caminos. Yu y Cai proponen en [137] un mecanismo, también centralizado, llamado *Heterogeneous Interactive Cultural Hybrid Algorithm* (HICHA) en el que, entre otro tipo de técnicas, se utilizan algoritmos genéticos. Otro mecanismo centralizado es el propuesto en [119] por Smith y Bullo en el que se analizan diferentes políticas de asignación en función de la frecuencia de llegada de las tareas. Así, se demuestra que para determinados valores de los parámetros, se puede conseguir una solución k-óptima, siendo k una constante. También destacan los trabajos realizados por Shehory y Kraus [116] en el ámbito de los sistemas multiagente, ya explicados anteriormente. En [127] Vig modifica y descentraliza parte de los algoritmos de Shehory y Kraus para su aplicación en sistemas multi-robot.

El elevado tiempo de ejecución que, en general, requieren los algoritmos centralizados, junto con los problemas de punto único de fallo y saturación en las comunicaciones, hacen inviable estos mecanismos en entornos dinámicos o con un elevado número de robots. Recientemente, el sistema centralizado propuesto por Ramchurn en [108] tiene en cuenta las restricciones temporales en problemas de tipo MR. El

tipo de problema que Ramchurn aborda es muy similar al planteado en el capítulo 5, en cambio el nivel de complejidad de sus algoritmos es muy superior al de cualquiera de los métodos propuestos en esta memoria.

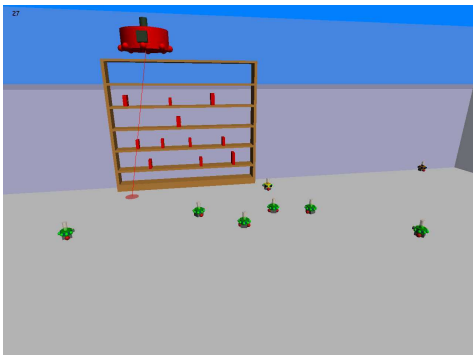
2.2.2. Sistemas basados en *swarm intelligence*

Los sistemas basados en *swarm intelligence* se inspiran en el comportamiento de las colonias de insectos, tales como hormigas o abejas. Se trata de sistemas auto-organizados (*self organized systems*) definidos en [23] de la siguiente manera: *"[...] systems of non-intelligent robots exhibiting collectively intelligent behavior"*. Es decir, se trata de sistemas en los que, sin existir ningún sistema central, aparece un comportamiento cooperativo a partir de la interacción de comportamientos individuales muy simples implementados en cada robot. Las principales ventajas de estos sistemas son las siguientes [110, 12]:

- Sencillez y bajo coste: dados los escasos requerimientos de los sistemas basados en *swarm*, se permite tener robots pequeños y de bajo coste.
- Escalabilidad: este tipo de algoritmos normalmente no requieren protocolos de comunicación entre los robots y en caso de necesitarlos son muy sencillos. Este hecho, añadido a la simplicidad de los métodos utilizados, permite incrementar el número de robots del sistema sin aumentar excesivamente el tiempo de ejecución de los algoritmos. Así, es habitual que los sistemas basados en *swarm* estén formados por un gran número de robots.
- Robustez: al poder tener una gran cantidad de robots, el fallo de uno o varios de ellos no afecta de manera crítica al funcionamiento global de la colonia. Además, debido de nuevo a su sencillez, errores en el sistema de comunicación entre robots no suelen suponer errores importantes en el conjunto del sistema.
- Flexibilidad: los robots pueden coordinar sus comportamientos de múltiples maneras diferentes, de forma que pueden ofrecer múltiples soluciones a un mismo

problema, en función de las condiciones del entorno.

(a) Ejemplo de ejecución de los robot iRobot en una tarea de cobertura de superficie.



(b) Ejemplo de utilización de robots heterogéneos aplicando métodos basados en *swarm*.

Figura 2.1: Ejemplos de tareas basadas en *swarm intelligence*.

El principal problema de los sistemas basados en *swarm* es la falta de un conocimiento global, que limita o impide totalmente que un robot pueda saber cuáles son las decisiones tomadas por los otros miembros de la colonia. Es por ello que las soluciones obtenidas con estos sistemas pueden estar muy lejos de una solución óptima. Además, las tareas que se llevan a cabo suelen ser muy simples y débilmente acopladas, tales como, la recogida de objetos, o la cobertura de una superficie usando robots homogéneos. McLurkin y Smith en [99] proponen un sistema basado en *swarm* para cubrir con robots una determinada área de manera uniforme. Para ello se utiliza comunicación local entre robots mediante infrarrojos y robots muy simples. La figura 2.1(a) muestra un ejemplo de ejecución de esta tarea. Recientemente se han propuesto mecanismos de *swarm* en los que los robots son heterogéneos y las tareas requieren una coordinación algo más estrecha, ejemplos de estos sistemas son [101, 41]. En [41] Ducatelle et al. utilizan dos tipos de robots: aéreos y terrestres, con el objetivo de recoger una serie de objetos. La comunicación entre estos robots se realiza mediante el uso de luces de colores que pueden ser captadas, de manera muy sencilla, mediante cámaras situadas en los robots. La figura 2.1(b) muestra un ejemplo de ejecución de

esta tarea. Actualmente, es aún tema de estudio hasta qué punto los sistemas basados en *swarm* son capaces de realizar tareas complejas.

Los llamados sistemas de umbral de respuesta (*response threshold*) son probablemente uno de los mecanismos de asignación de tareas basados en *swarm* más utilizados hoy en día. Ejemplos de sistemas de *response threshold* en tareas de recogida de objetos se describen en [136, 94, 72, 28, 2, 1, 16]. En estos sistemas, cada robot tiene un estímulo asociado a cada una de las tareas a ejecutar. El estímulo va variando a lo largo del tiempo e indica lo 'atractiva' que es una tarea para el robot. Cuando el valor del estímulo supera un determinado umbral (*threshold*) el robot empieza a ejecutar la tarea y la dejará de ejecutar cuando el estímulo sea inferior a otro determinado valor. En [16] Bonabeau et al. utilizan el valor del umbral y el estímulo para calcular la probabilidad que tiene un robot de ejecutar una determinada tarea. Se trata, por tanto, de un sistema no determinista. El estímulo de la tarea va aumentando a medida que transcurre el tiempo y ésta no finaliza. Por contra, el estímulo disminuye a medida que el número de robots asignados a ella se incrementa. Además, se propone un sistema que permite ajustar o aprender el valor del umbral en función de las condiciones del entorno. Un ejemplo de un sistema similar a *response threshold* es la clásica arquitectura ALLIANCE [104] de Parker. En esta arquitectura cada robot tiene asociado para cada tarea un valor llamado impaciencia, que indica las 'ganas' que tiene el robot de ejecutarla. Por otra parte, un valor, llamado *acquiesce*, indica las 'ganas' del robot de abandonar la tarea. La probabilidad de que un robot ejecute una tarea es función de ambos valores. Posteriormente, otros autores como Yang et al. [136] también han utilizado mecanismos de *response threshold* con una selección no determinista de la tarea a ejecutar.

En [1] Agassounon y Martinoli proponen un sistema para tareas de recogida de objetos, similar a *response threshold* totalmente determinista, es decir, cuando el estímulo supera un determinado umbral se inicia inmediatamente la ejecución de la tarea. En este caso, se utiliza el tiempo necesario para encontrar un objeto como estímulo para decidir si un robot debe pasar a ejecutar la tarea o descansar. Otro

sistema de *response threshold* determinista para la recogida de objetos es el propuesto por Lin et al. en [94]. El objetivo de este método es llegar a un compromiso entre el número de objetos recolectados y la energía consumida por los robots. Los resultados muestran como sistemas de comunicación muy simples entre robots permiten mejorar los resultados de los mecanismos de *swarm*. Haboush en [72] también utiliza un sistema totalmente determinista, pero para solucionar el problema de JSS en entornos multi-procesador. En [81] Kalra y Martinoli utilizan un sistema similar en entornos multi-robot en el que se deben recoger objetos del entorno, cuya posición es conocida a priori. En este caso se demuestra cómo los sistemas totalmente deterministas, en los que cada robot selecciona la mejor tarea desde su punto de vista, proporcionan mejores resultados que los basados en probabilidades y *response threshold*. Por lo que sabemos, no existe hasta la fecha ningún mecanismo de *swarm* que combine métodos deterministas con no deterministas o probabilistas. En el capítulo 3 de esta memoria se presentará una propuesta en este sentido, es decir, un método de *swarm* pseudo-aleatorio, resultado de la combinación de métodos deterministas y no deterministas. Los resultados experimentales muestran que este nuevo sistema mejora los resultados hasta ahora obtenidos por el resto de métodos basados en *swarm*

Existen documentados muy pocos trabajos que utilicen mecanismos de *swarm* para llevar a cabo tareas con restricciones temporales. En el campo de sistemas multi-agente cabe destacar el trabajo de Oliveira et al. [33] donde se utilizan mecanismos de *response threshold* en los que se tiene en cuenta el *deadline* de las tareas en problemas de tipo ST-SR-IA. A pesar de ello, la fórmula utilizada requiere ajustar una gran cantidad de parámetros y no se demuestra su utilidad en sistemas multi-robot. En [34] Acebo y de la Rosa presentan un sistema multi-robot basado en *swarm* que también tiene en cuenta los *deadlines* asociados a cada tarea para problemas de tipo ST-SR-IA. Para solucionar conflictos entre robots se utilizan métodos de comunicación relativamente complejos, que se asemejan más a los mecanismos de subastas que a métodos de *swarm*. Los mecanismos de *swarm* propuestos en el presente trabajo requieren de sistemas de comunicación muy simples, simplificando notablemente todo

el proceso. Además, los métodos de subastas propuestos amplían el ámbito de aplicación del método de Acebo y de la Rosa [34] ya que permiten afrontar problemas de tipo MR, sin necesidad de conocer a priori el tiempo de ejecución requerido.

Recientemente, algunos mecanismos basados en *swarm*, como por ejemplo el propuesto por Ferreira et al. en [45], permiten tener en cuenta la interdependencia entre tareas, es decir, cómo la ejecución de una de ellas puede afectar a las demás en problemas de tipo ST-SR-IA. Para ello se utilizan sistemas de comunicación relativamente complejos basados en el paso de *tokens*, sin restricciones temporales. Otros mecanismos de *swarm* son una combinación entre varios sistemas, por ejemplo en [32, 138] se combinan subastas (véase apartado 2.2.3) con métodos de *swarm*, o en [27] donde Cheng y Dasgupta también utilizan técnicas basadas en teoría de juegos. En todos estos casos los mecanismos de comunicación entre robots son mucho más complejos que los propuestos en los algoritmos de *swarm* de esta tesis y no se tienen en cuenta las restricciones temporales.

La mayoría de sistemas basados en *swarm* únicamente permiten afrontar problemas de tipo SR utilizando, además, robots homogéneos. Una excepción a estas restricciones es el trabajo realizado por Ducatelle et al. en [41], explicado anteriormente. Otro ejemplo de mecanismo basado en *swarm* que permite crear coaliciones de robots es el propuesto por Low et al. en [95], donde se utiliza *response threshold* y comunicación entre robots. La misión de los robots consiste en seguir con sus sensores a un conjunto de objetos que se encuentran en un entorno dividido en una serie de areas de tamaño fijo. El objetivo de este sistema de asignación de tareas es encontrar el número óptimo de robots a asignar a cada zona del espacio. Debido a la necesidad de mecanismos de comunicación relativamente complejos, el sistema de Hsiang no ofrece las ventajas de simplicidad y escalabilidad presentes en otros sistemas basados en *swarm*. Además, al contrario que los sistemas de subastas propuestos en los capítulos 4 y 5 de esta tesis, en el sistema de Hisiang los robots son homogéneos, el número de grupos o coaliciones es constante y no hay restricciones temporales ni prioridad en las tareas.

Un aspecto muy importante a tener en cuenta en los sistemas basados en *swarm* es el tratamiento que se hace de la interferencia física entre robots. Este aspecto ha sido ampliamente tratado en este tipo de sistemas, véase por ejemplo [107, 58, 56]. Debido a la importancia que tiene el fenómeno de la interferencia física en los métodos desarrollados en esta tesis, se ha dedicado la sección 2.2.4 a discutir los trabajos realizados hasta el momento al respecto.

2.2.3. Sistemas basados en comunicación: subastas

Buena parte de los métodos de asignación de tareas que se explicarán en este trabajo están basados en mecanismos de subastas. Los mecanismos de subastas utilizan protocolos de comunicación entre robots para llevar a cabo una negociación explícita sobre qué tareas ha de realizar cada uno de ellos. Estos protocolos están basados en el modelo *Contract Net Protocol* (CNP) propuesto en el año 1980 por Smith [117]. En el trabajo de Smith se definía un mecanismo para asignar tareas a nodos de comunicación (máquinas) conectados entre sí, que debían solucionar un determinado problema de manera cooperativa y distribuida. Cuando un nodo, llamado *manager*, encontraba o generaba una nueva tarea, antes de empezar a ejecutarla enviaba un mensaje a los demás nodos anunciándola. Cada uno de los otros nodos (*bidder*) enviaba al *manager* un valor (puja o *bid*) indicando lo adecuado que era para ejecutar la nueva tarea. El nodo *manager*, una vez recibidas todas las pujas, seleccionaba al mejor nodo para llevar a cabo la tarea. En el caso de sistemas multi-robot los nodos son los diferentes robots del sistema y al robot *manager* se le suele llamar subastador (*auctioneer*). A pesar de la existencia de múltiples tipos de subastas [134], éste es el más simple y más utilizado en sistemas multi-robot. A partir de él se han realizado múltiples ampliaciones y mejoras para adaptarlo a diferentes tareas [38]. Diversos autores [140, 83, 87] utilizan un sistema de subastas llamado subastas secuenciales de un solo ítem (*Sequential Single-item Auctions*-SSA) para solucionar problemas similares al TSP. En las subastas SSA hay un único robot o agente que subasta las tareas de manera secuencial. Se trata de un mecanismo muy simple pero muy efectivo, tal

como demuestran los resultados tanto teóricos como experimentales. Otros ejemplos de subastas tipo SSA se pueden encontrar en [125, 132]. Este tipo de subastas son similares a las utilizadas por los métodos propuestos en el capítulo 3 de esta tesis, donde se utilizan algoritmos de tipo *Earliest Deadline First* (EDF), ampliamente utilizado en sistemas multi-procesador, para decidir el orden de ejecución de los distintos procesos.

Uno de los primeros sistemas multi-robot que utilizó el concepto de subasta para la asignación de tareas fue el sistema MURDOCH propuesto por Gerkey y Mataric en [53, 55]. En este sistema los robots pueden ser heterogéneos y envían sus pujas a un robot subastador que sigue el esquema de subastas antes explicado para el modelo CNP. El subastador monitoriza la ejecución de la tarea, de manera que si no progresa al ritmo deseado se puede iniciar otro proceso de subasta. La principal ventaja de este sistema es su sencillez, en cambio no tiene en cuenta muchos problemas, como la interferencia física entre robots, las restricciones temporales de las tareas o las predicciones en el tiempo de ejecución en función del tipo de robots; problemas que sí se tienen en cuenta en el presente trabajo. Además, el método de estimación del tiempo de ejecución, explicado en el capítulo 5, permite no tener que utilizar ningún proceso de monitorización para ejecutar la tarea. En [18] Botelho y Alami proponen el sistema M+ que utiliza mecanismos similares a las subastas junto con un método de planificación que permiten solucionar posibles conflictos entre robots. Este mecanismo únicamente soluciona problemas de tipo ST-SR y no incluye restricciones temporales. Dias y Stentz presentan en [35, 36, 37] una solución también basada en subastas llamada *TraderBots*. Este sistema está inspirado en cómo actúan los agentes en un mercado capitalista, donde cada robot intenta maximizar sus propios beneficios (tareas a ejecutar) de una manera egoísta, sin tener en cuenta en sus pujas a los demás robots. Si se maximizan los beneficios individuales de cada robot, también se maximizará el beneficio total del sistema en su conjunto. Este mecanismo se utiliza sobre todo para solucionar el problema del viajante de comercio con múltiples robots. Posteriormente fue mejorado y ampliado por Zlot en [141] para unir el proceso de

planificación y el de asignación de tareas en un solo método, permitiendo solucionar problemas de tipo MR. Al igual que en todos los casos anteriores, no permite tareas con restricciones temporales. Otros sistemas, como por ejemplo el propuesto por Drozd en [40], utilizan mecanismos de programación genética para obtener la fórmula de la puja y conceptos propios de *swarm* para decidir cuándo pujar. En los experimentos realizados para verificar estos métodos no se tenía en cuenta la interferencia física entre robots y únicamente se permitía un único robot por tarea. Además, el tiempo de ejecución de los algoritmos genéticos es muy elevados, sobre todo si se compara con el de los mecanismos de aprendizaje propuestos en el capítulo 5 de este documento.

La mayoría de los sistemas de subastas vistos hasta el momento sólo permiten asignar un robot por tarea, son, por tanto, problemas de tipo SR. En cambio, los problemas de tipo MR que requieren la creación de coaliciones han sido tratados escasamente en sistemas de subastas. El sistema RACHNA propuesto por Vig en [128] permite tareas de tipo MR pero sin restricciones temporales. El método propuesto por Sariel en [111] también permite múltiples robots por tarea, pero el tamaño de los grupos ha de ser conocido antes del inicio de la ejecución de cada tarea. Otro mecanismo que también utiliza métodos basados en subastas es el propuesto por Chaimowicz en [25, 24]. En este sistema, al igual que en algunos de los propuestos en esta tesis, cada tarea tiene un robot líder que se encarga de formar los grupos de trabajo o coaliciones. En cambio, el número de robots necesarios para ejecutar cada tarea está prefijado y, al igual que en todos los métodos hasta ahora mencionados, tampoco tiene en cuenta las restricciones temporales. Finalmente, Paquet en [103] propone un método en el que el tamaño del grupo es desconocido a priori y se tiene en cuenta los *deadlines* de las tareas. Este mecanismo únicamente ha sido probado en el simulador de la RoboCup Rescue y no sobre robots. Además, la decisión sobre cuántos robots son necesarios en cada tarea no puede realizarse en tiempo de ejecución, debido a su elevada complejidad. En cambio, los métodos de subastas aquí propuestos se aplican sobre robots, tienen en cuenta la interferencia física y permiten ajustar el

tamaño de los grupos durante la ejecución de la tarea. Tang y Parker proponen en [122] el uso de subastas junto con el método llamado ASyMTRe, lo que permiten la ejecución de tareas fuertemente acopladas. Otros métodos que también permiten la creación de coaliciones de robots se describen en [129, 131, 79, 76, 139]. En todos estos trabajos no se tiene en cuenta la interferencia física entre robots ni las restricciones temporales de las tareas.

Entre los métodos basados en subastas que permiten restricciones temporales el más destacable de cara al presente trabajo es el propuesto por Jones et al. en [78]. Este método utiliza un sistema de aprendizaje para decidir el valor de las pujas en problemas de tipo SR. Anteriormente, Lemaire et al. proponen en [90] un método de planificación de tareas en las que éstas deben ejecutarse, de manera aproximada, antes de un determinado *deadline*. El método de Lemaire no permite *deadlines* estrictos ni múltiples robots por tarea. En [100] Melvin et al. proponen un método de subastas para solucionar el problema VRPTW, en el que las ventanas temporales de las tareas no se pueden solapar. En cambio, en el método aquí propuesto no existe restricción alguna en los tipos de *deadlines*. En [22] Campbell et al. proponen un nuevo método, no totalmente basado en subastas, que permite tareas con *deadlines* y en el que los agentes (robots) pueden decidir no seleccionar una tarea a pesar de ser, en ese momento, la mejor opción a llevar a cabo. Los resultados experimentales muestran una clara mejora respecto a los mecanismos clásicos de subastas. En los mecanismos de subastas propuestos en esta tesis, los robots libres siempre pujan por todas las tareas. En [113] Sellner y Simmons presentan un método de predicción del tiempo de ejecución utilizando técnicas basadas en *kernels*, aunque no se tiene en cuenta la interferencia entre robots y sólo afronta problemas de tipo SR. Además, los mecanismos de asignación utilizados son mucho más complejos que los métodos basados en subastas.

En todos los tipos de subastas mencionados hasta ahora los robots pujan por una única tarea cada vez. Existe otro tipo de subastas, llamadas subastas combinatorias (*combinatorial auctions*), en las que los robots pujan por un conjunto de tareas. En

[96] López et al. utilizan subastas combinatorias durante la competición RoboCup Rescue. Este sistema permite la distribución de agentes en un entorno dinámico, aunque una vez asignada una tarea a un agente, éste no puede ejecutar otra hasta que no acabe con la primera. En [46] Frias-Martínez et al. realizan una comparación entre subastas combinatorias y no combinatorias en el entorno RoboCup Soccer. A pesar de utilizar un entorno muy limitado (el cambio de rol únicamente se puede realizar después de cada gol, sólo hay tres roles, el equipo contrario se mueve de manera aleatoria, hay relativamente pocos robots, etc), se puede ver como las subastas combinatorias mejoran los resultados. Este tipo de subastas también ha sido utilizada en otros entornos multi-robot, por ejemplo en [15, 37]. A pesar de las mejoras que suponen las subastas combinatorias, su complejidad las hace inviables en entornos muy dinámicos y con un gran número de robots, tal como se apunta Sariel en [112].

Tal como se ha visto en esta sección, los mecanismos de subastas permiten distribuir el proceso de decisión entre los diferentes robots que componen el sistema utilizando protocolos de comunicación. De esta manera, se evitan buena parte de los problemas de los sistemas centralizados, como por ejemplo el punto único de fallo o la sobrecarga en las comunicaciones del agente central. Además, estos algoritmos son, en general, mucho menos complejos que los utilizados en los métodos centralizados. Por contra, la solución obtenida mediante subastas suele incrementar el tiempo total de ejecución o la energía consumida por los robots, respecto a los mecanismos centralizados. Por otra parte, las subastas son métodos más complejos que los basados en *swarm* ya que siempre requieren protocolos de comunicación para coordinar los robots. En cambio, la solución obtenida con subastas suele mejorar el rendimiento del sistema (incrementa el número de tareas finalizadas, reduce la energía consumida por los robots, etc.) respecto a los métodos basados en *swarm*. Además, las subastas, al incluir comunicación explícita entre robots, permiten llevar a cabo tareas más complejas, como por ejemplo tareas fuertemente acopladas.

2.2.4. Tratamiento de la interferencia física

Uno de los objetivos de esta tesis es analizar cómo influye la interferencia física en el proceso de asignación de tareas. En [109] Rosenfeld et al. ponen claramente de manifiesto el gran impacto que puede tener la interferencia física en los sistemas multi-robot. En la literatura especializada se puede encontrar una gran cantidad de trabajos destinados a intentar minimizar los efectos de la interferencia, sobre todo utilizando mecanismos basados en *swarm* y en los que únicamente se permite asignar un robot por tarea. De entre todos estos trabajos, cabe destacar el realizado por Goldberg y Mataric [57] en el que se definen diferentes zonas en las que únicamente puede acceder un robot en cada instante. Estrategias similares a ésta se presentan en el capítulo 4 de esta memoria. En [126] Vaughan et al. proponen un mecanismo en el que a cada robot se le asigna un nivel de agresividad, que indicará su prioridad. Cuando dos robots intentan acceder a la misma posición, el que tenga una menor agresividad se verá repelido, mediante campos de potencial, por aquellos más prioritarios. Lein y Vaughan en [89] presentan una estrategia, llamada *bucket brigade*, para tareas de recolección de objetos. En este método cada robot tiene una área de búsqueda, cuando encuentra un objeto se dirige hacia un punto de descarga hasta llegar al límite de su zona, momento en el cual simplemente deja el objeto. El robot que se encarga de la zona donde se ha dejado el objeto lo vuelve a recoger para llevarlo hacia el punto de descarga.

En [30] Dahl et al. proponen un mecanismo llamado *Vacancy Chains* para llevar a cabo tareas de *foraging* de tipo MR. El sistema va incrementando el número de robots asignados a cada tarea hasta que detecta una disminución en la cantidad de objetos transportados. Esta disminución en el rendimiento del sistema únicamente puede ser debida a la interferencia física entre robots. El método de Dahl no permite tratar tareas con *deadlines* ni estimar el tiempo de ejecución de las mismas. Finalmente, destacar los trabajos de Lerman et al. [92, 91] en los que se modeliza matemáticamente la interferencia en tareas de recogida de objetos sin restricciones temporales. El trabajo de Lerman únicamente utiliza mecanismos de asignación de tareas basados

en *swarm* y tampoco permite estimar el tiempo de ejecución de las tareas.

2.3. Conclusiones

En este capítulo se ha dado una visión general sobre los diferentes mecanismos de asignación de tareas en sistemas multi-robot, que actualmente existen. Se ha mostrado cómo diferentes autores han definido formalmente el problema de asignación de tareas en sistemas multi-robot. Tal como se ha expuesto, la mayoría de estos problemas son de tipo NP-hard y, por tanto, actualmente no existe ningún algoritmo que pueda generar una solución óptima en un tiempo acotado por un polinomio. De esta manera, se evidencia la elevada complejidad de los problemas de asignación de tareas y la necesidad de afrontarlos mediante algoritmos heurísticos que, a pesar de no proporcionar una solución óptima, tienen tiempos de ejecución relativamente bajos. Además, las definiciones expuestas se utilizarán en los siguientes capítulos de esta memoria para formalizar diferentes aspectos de los algoritmos propuestos y analizar su complejidad.

Se han analizado los principales mecanismos de asignación de tareas relacionados con los propuestos en esta tesis. Para ello, se han dividido en tres grandes grupos: algoritmos centralizados, sistemas de *swarm intelligence* y mecanismos basados en subastas. Uno de los principales problemas de los sistemas centralizados es el elevado tiempo de ejecución que necesitan. Los sistemas basados en *swarm* se pueden dividir entre aquellos que utilizan algoritmos deterministas y los que utilizan probabilidades. Hasta la fecha, ningún sistema ha combinado ambas estrategias. Además, muy pocos trabajos han analizado el comportamiento de los métodos basados en *swarm* en tareas con restricciones temporales y los pocos estudios que se han llevado a cabo requieren complejos sistemas de negociación entre los robots. Esta tesis presenta el primer algoritmo de *swarm* pseudoaleatorio, híbrido entre sistemas deterministas y no deterministas. Los resultados experimentales muestran como esta nueva estrategia mejora los resultados respecto a los sistemas propuestos hasta hoy en entornos en los

que las tareas tienen restricciones temporales.

Por otra parte, el análisis de los métodos actuales de asignación de tareas muestra como, hasta ahora, no existe ningún mecanismo basado en subastas que tenga en cuenta la interferencia física entre robots. En esta tesis se propone el primero de estos mecanismos. Tampoco los mecanismos actuales de subastas ofrecen una solución satisfactoria al problema de creación de coaliciones con restricciones temporales asociadas a las tareas. Uno de los principales problemas de estos mecanismos es el de estimar el tiempo de ejecución de la tarea, en función de la coalición de robots que la lleva a cabo. Para solucionarlo, en el capítulo 5 se presentará un nuevo mecanismo basado en subastas que permite afrontar este tipo de tareas, ajustando el tamaño de las coaliciones de robots a las restricciones temporales.

A modo de resumen, la tabla 2.1 muestra algunas de las características de los principales mecanismos de asignación de tareas vistos en este capítulo. La columna *Restricciones Temporales* indica si las tareas pueden tener *deadlines* o no y *Tamaño Grupo* significa que, en el caso de poder solucionar problemas de tipo MR, el tamaño de los grupos de trabajo está o no prefijado. Las tres últimas filas de esta tabla muestran en qué grupo se incluyen los métodos de asignación de tareas presentados en esta tesis.

Referencia	Tipo	MR/SR	TE/IA	Restricciones Temporales	Tamaño Grupo
[18] [53]	subasta	SR	IA	No	-
[83] [140] [112]	subasta	SR	TE	No	-
[90] [100]	subasta	SR	TE	con restricciones	-
[78]	subasta	SR	TE	Sí	-
[36]	subasta	SR	TE	No	-
[25]	subasta	MR	IA	No	fijo
[128]	subasta	MR	IA	No	variable
[76] [129]	subasta	MR	TE	No	variable
[111]	subasta	MR	TE	No	fijo
[103]	subasta	MR	TE	Sí	variable (*off-line*)
[141]	subasta	MR	TE	No	variable
[122]	subasta/ central.	MR	IA	No	variable
[40]	subasta/ *swarm*	SR	IA	No	-
[34]	subasta/ *swarm*	MR	IA	Sí	-
[33]	*swarm*	SR	IA	Sí	-
[41]	*swarm* (comunicación)	MR	IA	No	variable
[136][101] [72] [94] [1]	*swarm*	SR	IA	No	-
[95]	*swarm* (comunicación)	MR	IA	No	variable (con restricciones)
[45]	*swarm* (comunicación)	SR	IA	No	-
[84]	central.	MR	TE	No	Sí
[137]	central.	ST	TE	No	-
[119]	central.	MR	TE	No	con restricciones
Capítulo 3	*swarm* subasta	SR	IA	Sí	-
Capítulo 4	subasta	MR	IA	No	variable
Capítulo 5	subasta	MR	IA	Sí	variable

Tabla 2.1: Características de los principales mecanismos de asignación de tareas.

Capítulo 3

Configuraciones con un único robot por tarea

En este capítulo se analiza el comportamiento de los sistemas SR-ST-IA en entornos con restricciones temporales en las tareas, además se comparan los resultados de varios algoritmos de *swarm* con los obtenidos con algoritmos basados en subastas. También se estudia el impacto de la asignación de múltiples tareas a un sólo robot con respecto a sistemas en los que se puede asignar una única tarea a cada robot. Tener sistemas con varias tareas por robot implica que cada uno de ellos ha de planificar los objetivos que tiene asignados, que en sí ya es un problema NP-hard. Teniendo en cuenta todas las posibles combinaciones de algoritmos probados se han comparado un total 11 métodos diferentes. Éste es el primer estudio en el que se realiza una comparación entre subastas y métodos de *swarm* en entornos con restricciones temporales asociadas a las tareas. Trabajos anteriores, como [81] de Kalra y Martinoli, han realizado comparaciones entre métodos de *swarm* y subastas, pero únicamente han tratado tareas sin restricciones temporales.

También en este capítulo se proponen dos nuevos métodos de *swarm*: llamados *swarm* pesudo-aleatorio con distancia (*Pseudo-random Swarm Distance*, PSW-D) y *swarm* pseudo aleatorio con robots (*Pseudo-random Swarm Robots*, PSW-R). Se trata de los primeros métodos de *swarm* híbridos que combinan tanto características de

algoritmos deterministas como de no deterministas. Los resultados experimentales han mostrado como ambas propuestas mejoran substancialmente los resultados obtenidos con respecto a los actuales algoritmos de *swarm*.

Por último, se han utilizado varios mecanismos basados en subastas: subastas secuenciales sin orden (*Sequential Unordered Auctions*-SUA), subastas EDF (*Earliest Deadline First Auction*-EDFA) y subastas secuenciales mejor robot-tarea (*Sequential Best Pair Auctions*-SBPA). Estas estrategias difieren entre ellas en la manera en que un subastador central anuncia las tareas a los robots y en el tratamiento que posteriormente hace de las pujas recibidas. Los métodos SBPA y SUA ya han sido utilizados por otros autores [133, 53], aunque en este trabajo se analiza por primera vez su comportamiento en entornos con restricciones temporales. En cambio, el método EDFA, presentado en este trabajo, es el primer sistema de subastas en entornos con restricciones temporales que también incluye una planificación EDF. Como resultado de las pruebas realizadas, se presenta un segundo método de subastas llamado mejor pareja EDF (*Earliest Deadline First Best Pair*-EDFBP), un sistema híbrido entre EDFA y SBPA. Los resultados de EDFBP son similares a los obtenidos con SBPA pero con una complejidad computacional que puede llegar a ser un 50 % menor. Por tanto, el método EDFBP permite reducir la complejidad de los algoritmos de subastas actuales basados en un subastador central.

3.1. Tarea prototipo: recolección de objetos

En todos los experimentos realizados se ha utilizado la tarea de recolección de objetos (*foraging*) con restricciones temporales asociadas a cada objeto. Así, la tarea de *foraging* se define de la siguiente manera: los robots, inicialmente situados de manera aleatoria en el entorno, han de recolectar una serie de objetos también situados aleatoriamente. Al contrario de lo que sucede en algunas tareas de *foraging*, los robots no han de retornar los objetos a un punto de descarga, sino que simplemente basta con 'recogerlos' para considerar que se han recolectado. Diversos autores han utilizado

este tipo de tareas, como por ejemplo Kalra y Martinoli en [81] y, por tanto, se podrán comparar sus resultados con los obtenidos por los algoritmos presentados a lo largo de este capítulo.

Nuevos objetos (tareas) aparecen en el entorno siguiendo un proceso de Poisson con una tasa de llegadas λ, donde λ representa el número medio de nuevos objetos por unidad de tiempo. Existe además un agente o subastador central que conoce la posición de todos los objetos en el entorno y que puede comunicar esta posición a todos los robots. Cada objeto tiene un peso asociado, que representa la cantidad de trabajo necesario para poderlo recoger por completo. Cada robot tiene una capacidad de carga que es la cantidad de peso del objeto que puede procesar por unidad de tiempo. Para considerar una tarea como finalizada (objeto recogido), el robot ha de situarse cerca de ésta e iniciar el proceso de recogida que durará un tiempo igual a $\frac{peso}{capacidadCarga}$, transcurrido el cual el robot vuelve a estar libre e intenta obtener otra tarea. Cada objeto sólo podrá ser procesado por un único robot simultáneamente, es decir siempre se tratará de un problema de tipo SR.

Cada objeto tiene un límite temporal (*deadline*) para ser recogido por un robot. El tiempo límite de ejecución de una tarea empieza a contar en el momento en que aparece en el entorno. Si un objeto no ha podido ser recogido completamente antes de su tiempo máximo de ejecución, desaparecerá del entorno y ningún robot podrá seguir realizando esa tarea. El objetivo de los mecanismos de asignación de tareas será maximizar el número de objetos recogidos antes de su límite temporal.

Las características de esta tarea se ajustan a los requisitos de muchos escenarios reales: por ejemplo, tareas de cooperación entre robots terrestres y aéreos en situaciones de catástrofes o incendios, como las propuestas en el proyecto CROMAT o COMETS [48]. En estos tipos de entornos, el agente central podría ser un robot aéreo con buenas capacidades sensoriales capaz de detectar objetivos y comunicarlos a los robots terrestres. Otro posible entorno es el de tareas de rescate como las propuestas en RoboCup Rescue, en el que las tareas a ejecutar pueden ser la atención a heridos en una situación de catástrofe.

Si no se planifica el orden de ejecución de las tareas ni se asignan restricciones temporales a los objetos, esta tarea de recogida se ajusta perfectamente a la definición del problema OAP, ya explicado en el capítulo 2. Tal como ya se comentó, el algoritmo Húngaro permite encontrar una solución óptima en un tiempo $O(nm^2)$, donde n es el número de agentes y m el de tareas. No obstante, debido a que los robots pueden tener múltiples objetos asignados y planificar en qué orden se han de visitar (ver sección 3.4), cada robot ha de solucionar el problema del viajante de comercio (TSP) que es de tipo NP-hard. Además, el hecho de que haya restricciones temporales en las tareas, hace que este problema se pueda asemejar al VRPTW, también NP-hard, en el que la ventana temporal de cada tarea se inicia en el momento de su aparición. Así, para esta misión de recogida de objetos, la asignación de un conjunto de tareas $T = \{t_1, ..., t_m\}$ a un conjunto de robots $R = \{r_1, ..., r_n\}$ se puede representar mediante el conjunto $TA = \{C^0, ..., C^{n'}\}$, donde cada C^i=$\{r_i, T^i\}$ y $T^i \subseteq T$ es el conjunto de tareas asignadas al robot r_i. T^0 corresponde al conjunto de tareas sin ningún robot asignado. Los robots sin tareas asociadas no aparecen en el conjunto TA, por tanto $n' \leq n$. Para que una asignación TA sea válida ha de cumplir las siguientes características:

- $\bigcup_{i=0}^{n} T^i = T$.
- $T^i \bigcap T^j = \emptyset$ para todo $i \neq j$. Es decir, cada tarea únicamente puede ser asignada a un robot.
- Si $j > 0$, todas las tareas de T^j han de cumplir su *deadline*.

El objetivo del algoritmo de asignación de tareas es encontrar una asignación, llamada asignación óptima TA^*, que maximice una determinada función de utilidad $U(TA)$. Esta definición amplía la dada por Viguria [129] y Zlot et al. [141] para permitir tareas con restricciones temporales. Para la tarea de recogida llevada a cabo, la asignación TA^* es la que permite finalizar un mayor número de tareas antes de su *deadline*. Además, en entornos dinámicos, como el aquí propuesto, el conjunto de

tareas puede variar a lo largo del tiempo, por tanto, la asignación TA también irá variando.

3.2. Métodos basados en *swarm*

En esta sección se explicarán los mecanismos de *swarm* implementados. En todos los casos se han utilizado estrategias ST-SR-IA. No se han implementado problemas de tipo TE debido a que, al requerir planificación, incumplirían los requisitos de sencillez de los sistemas basados en *swarm*.

En los métodos de *swarm* cada robot selecciona la siguiente tarea a ejecutar mediante algoritmos muy sencillos que requieren muy poca o ninguna capacidad de comunicación entre robots. De esta manera, aparece un comportamiento social a partir de la agregación de comportamientos individuales muy simples. La tarea propuesta en la sección 3.1 sólo requiere robots con capacidad de recepción, ya que el agente central es el único que necesita transmitir información y, por tanto, es susceptible de poder ser implementada mediante mecanismos de *swarm*.

3.2.1. Método determinístico

La estrategia de *swarm* más simple implementada es la de elegir siempre la tarea más cercana al robot (*Nearest First Task*-NFS), se trata, por tanto, de un algoritmo totalmente determinista. A medida que las tareas van llegando, el agente central envía la información a todos los robots. Cada robot seleccionará como siguiente tarea a ejecutar aquella que esté a la menor distancia de él. Para evitar seleccionar tareas demasiado alejadas, los robots podrán seleccionar únicamente aquellas que estén a una distancia menor a un umbral D. La complejidad algorítmica de la estrategia NFS es $O(m)$ donde m es el número de tareas de las que tiene conocimiento cada robot. A nivel de comunicaciones, la complejidad también es $O(m)$ ya que el agente central ha de enviar un mensaje por cada una de las nueva tareas que vayan apareciendo. La simplicidad de este método implica muchos problemas, el principal de ellos es la

interferencia entre robots, producida cuando dos o más de ellos seleccionan la misma tarea para ejecutar. Como se verá en las pruebas experimentales, la interferencia tiene un gran impacto sobre los resultados obtenidos.

3.2.2. Método de *response threshold* no determinístico

La siguiente estrategia de *swarm* analizada está basada en el método clásico de *response threshold* (RTH) [136, 1]. En esta estrategia cada robot r tiene asociado a cada tarea t un estímulo $s_{r,t}$, que representa lo adecuado que es el robot para la tarea. Por ejemplo, en los experimentos realizados a lo largo de este trabajo $s_{r,t}$ es igual a la inversa de la distancia entre robot y objeto. Cuando $s_{r,t}$ supera un determinado valor umbral (threshold, θ_r), el robot r empieza a ejecutar la tarea t. Para evitar una excesiva dependencia del valor umbral, la elección de la tarea a ejecutar es no determinista [16], de manera que la probabilidad de que el robot r ejecute la tarea t, ($P_{r,t}$) será igual a:

$$P_{r,t} = \frac{s_{r,t}^n}{s_{r,t}^n + \theta_r^n} \tag{3.1}$$

Como se puede observar, si $s_{r,t}$=θ_r, la probabilidad de ejecutar la tarea será de 0,5. La figura 3.1 muestra los valores de la ecuación 3.1 en función del valor del estímulo ($s_{r,t}$) para diversos valores del exponente n y con un valor de θ_r igual a 50. Siguiendo las recomendaciones de trabajos de otros autores [136, 81], el valor de n en todos los experimentos es siempre igual a 2. En cuanto al parámetro θ, se han probado distintos valores, tal como se mostrará en la sección de resultados experimentales. El algoritmo 1 muestra la implementación realizada en esta tesis, donde $random(0.,1)$ es una función que retorna un número aleatorio entre 0 y 1. La complejidad algorítmica de este algoritmo es $O(m)$, donde m es el número de tareas, aunque si el robot llega a seleccionar alguna tarea, el número de iteraciones va a ser mucho menor.

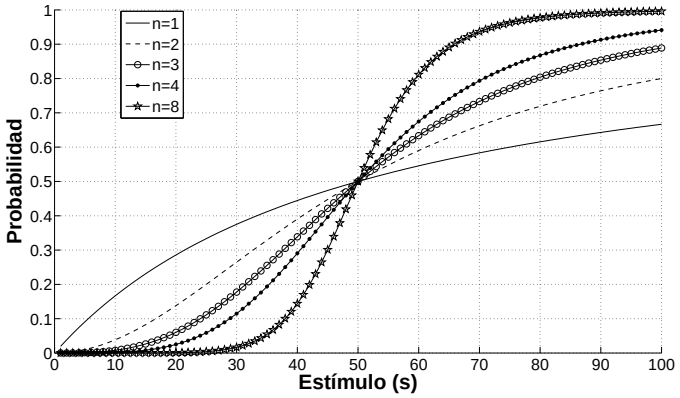


Figura 3.1: Valores de $P_{r,t}$ (ecuación 3.1) en función del estímulo y del valor de n con $\theta_r = 50$.

Algoritmo 1 Algoritmo de *response threshold* (RTH) para el robot r

Entrada: T=Lista de tareas no asignadas
1: **para todo** $t \in T$ **hacer**
2: **si** $P_{r,t} > random(0.,1)$ **entonces**
3: Retornar t {Se empezará a ejecutar la tarea t}
4: **fin si**
5: **fin para**
6: Retornar *null*

3.2.3. Métodos de *swarm* pseudo-aleatorios

Con el objetivo de reducir la interferencia entre robots, se ha propuesto un nuevo método llamado *swarm* pseudo aleatorio (*Pseudo-random Swarm*-PSW), combinación de los 2 métodos anteriores: NFS y RTH, del cual se han implementado 2 variantes: *Pseudo-random Swarm Robot* PSW-R y *Pseudo-random Swarm Distance* (PSW-D).

En general, en los métodos PSW cada robot r seleccionará de manera pseudo-aleatoria un subconjunto de las tareas disponibles y, de entre ellas, se seleccionará la que esté más cerca como siguiente tarea a ejecutar. El esquema concreto que sigue el método PSW se puede ver en el algoritmo 2. Por una parte este algoritmo se comporta como un método de *response threshold* en el que la selección de una tarea depende de una probabilidad $P_{r,t}$ (ver línea 3). Por otra parte, el algoritmo también es

determinista ya que la mejor tarea (la más cercana) es seleccionada, de manera análoga al funcionamiento del sistema NFS (ver líneas 4-5). La complejidad algorítmica de PSW es igual a $O(m)$ donde m es el número de tareas, complejidad similar a los algoritmos de *response threshold* clásicos.

Algoritmo 2 Algoritmo PSW para el robot r

Entrada: T=Lista de tareas[1..N] no asignadas
1: $t_{best} \leftarrow null$
2: **para todo** $t \in T$ **hacer**
3: **si** $P_{r,t} > random(0.,1)$ **entonces**
4: **si** t más cerca que t_{best} **entonces**
5: $t_{best} \leftarrow t$
6: **fin si**
7: **fin si**
8: Retornar t_{best}
9: **fin para**

Se han propuesto dos métodos PSW en función de cómo se realiza la selección pseudo-aleatoria de las tareas: PSW con distancia (PSW-D) y PSW con robots (PSW-R). En el método PSW-D la probabilidad $P_{r,t}$ se calcula utilizando la fórmula de *response threshold* que aparece en la ecuación 3.1, donde el estímulo será igual a la inversa de la distancia a la que se encuentre un objeto o tarea del robot. En cambio, el método PSW-R, preselecciona las tareas en función del número de robots que estén dentro de una región circular de radio A cuyo centro se sitúa en el robot r. De esta manera, la probabilidad $P_{r,t}$ de preseleccionar un robot en el método PSW-R es igual a:

$$P_{r,t} = \frac{1}{Na_r} \tag{3.2}$$

donde Na_r representa el número de robots cuya distancia al robot r es menor que A. Cuanto menor sea el valor del radio A, más cercano será el comportamiento de PSW a un algoritmo determinista, llegando a comportarse como el método NFS si $A = 0$. Se ha decidido utilizar la fórmula 3.2 en lugar de una similar a 3.1 ya que esta última necesita un parámetro más (el umbral θ) y dado que las pruebas

experimentales mostraron que ambas ecuaciones presentaban resultados similares. El principal inconveniente del algoritmo PSW-R es la necesidad de comunicación entre los robots. De todas maneras, el mecanismo de comunicación requerido es muy simple ya que únicamente se necesita enviar el identificador y la posición del robot. Como se ha demostrado en las pruebas experimentales, los resultados del algoritmo no varían sustancialmente cuando se producen pequeñas variaciones sobre la distancia A. Es más, el valor de A muestra un comportamiento muy estable, debido a que la probabilidad de elección de una tarea depende de los robots y no del entorno.

Algoritmo 3 Algoritmo de ajuste del valor de A

```
1: si robot libre y deadline no cumplido entonces
2:     A_{i+1} ← (1 − α)A_i
3: si no
4:     si interferencia detectada entonces
5:         A_{i+1} ← (1 + α)A_i
6:     si no
7:         A_{i+1} ← A_i
8:     fin si
9: fin si
```

A modo de ejemplo, para poder ajustar el valor de la distancia A se ha propuesto el algoritmo 3, donde α es un factor de aprendizaje entre 0 y 1 ($\alpha \in [0, 1]$). Tal como se puede observar, cuando un robot libre recibe del agente central el mensaje de que una tarea no ha finalizado antes de su *deadline*, la distancia A disminuye su valor para que en un futuro la probabilidad de seleccionar una tarea sea mayor. En cambio, si un robot está ejecutando una tarea y detecta una interferencia con otro robot, la distancia A se incrementa, disminuyendo las probabilidades de ejecutar una nueva tarea en el futuro. Cuando una tarea no ha podido ser ejecutada antes de su *deadline*, el agente central comunica este hecho a todos los robots. La manera en que un robot detecta que ha habido una interferencia con otro robot depende del tipo de tarea que se esté ejecutando. Por ejemplo, en una misión de recolección de objetos, si un robot aún no ha llegado al objeto que le ha sido asignado pero el agente central notifica que ese mismo objeto ya ha sido recogido, entonces este robot sabrá que otro ya seleccionó

la misma tarea y que, por tanto, se ha producido una interferencia.

Tal como se demostrará en la sección 3.5, gracias a la combinación de una parte probabilística y de una totalmente determinista que tienen los algoritmos PSW, se reducen los efectos perniciosos de la interferencia entre robots apuntados en [81]. Es más, los resultados mejoran los obtenidos con los algoritmos clásicos basados en *response threshold*.

3.3. Mecanismos de subastas en entornos ST-SR-IA

En esta sección se exponen los algoritmos basados en subasta en los que una única tarea puede ser asignada a cada robot (SR). Las estrategias basadas en subastas implementadas son las siguientes: subastas secuencial mejor robot-tarea (*Sequential Best Pair Auctions*-SBPA), subastas EDF (*Earliest Deadline First Auction*-EDFA), subastas secuenciales sin orden (*Sequential Unordered Auctions*-SUA) y subastas EDF mejor pareja (*Earliest Deadline First Best Pair*-EDFBP). En todos los casos habrá un agente central que actuará de subastador que enviará la información de las tareas a los robots, recibirá las pujas y seleccionará al mejor robot para cada tarea.

3.3.1. *Sequential Best Pair Auctions* (SBPA)

En la estrategia de mejor robot-tarea (*Sequential Best Pair Auctions*-SBPA) se ha utilizado la selección clásica ya usada en algunos algoritmos, como por ejemplo en el método BLE de Werger y Mataric [133] o analizada por Gerkey et al. en [54]. Se ha optado por utilizar el algoritmo de mejor pareja debido a su simplicidad y a que es el más utilizado por los mecanismos de asignación de tareas en sistemas multi-robot. En este algoritmo, cada vez que una nueva tarea aparece en el entorno o cuando un robot finaliza la ejecución de su objetivo, el subastador empieza una nueva ronda de subastas siguiendo el proceso que se puede ver en el algoritmo 4. En primer lugar, envía un mensaje solicitando a todos los robots libres una puja para cada una de las tareas pendientes de ser ejecutadas (ver líneas 1-3). Cada robot libre

envía un mensaje con el valor de la puja al subastador central. En los experimentos realizados, el valor de la puja ha sido el tiempo estimado por el robot para acabar la tarea. Si un robot estima que no podrá finalizar la tarea antes de su *deadline*, entonces no efectuará ninguna puja. Para calcular el tiempo estimado se utilizarán las características cinemáticas del robot y la distancia entre robot y tarea. Una vez el subastador ha recibido todas las pujas, inicia un bucle (ver línea 4 y siguientes) en el que en cada iteración se selecciona la mejor pareja robot-tarea, esto es, se selecciona el menor tiempo de ejecución, y se notifica esta elección al robot seleccionado. Este algoritmo se utilizará para comparar los resultados obtenidos con los del resto de algoritmos.

Algoritmo 4 Algoritmo SBPA

Entrada: T=Lista de tareas no asignadas
1: **para todo** tarea $t \in$ T **hacer**
2: Solicitar las pujas a los robots libres
3: **fin para**
4: **repetir**
5: Seleccionar la mejor pareja robot-tarea (mejor puja)
6: Enviar mensaje de asignación al robot seleccionado
7: Eliminar la tarea-robot seleccionados de la lista de robots libres y tareas no asignadas
8: **hasta que** no haya más robots o tareas sin asignar.

El análisis detallado de la complejidad computacional del algoritmo SBPA implementado es el siguiente: sea m el número de tareas y n el número de robots, entonces la complejidad del bucle de la línea 1 a la línea 3 es $O(m)$. Para encontrar la mejor pareja robot-tarea (línea 5) es necesario comprobar cada robot y cada tarea, siendo su complejidad igual a $O(m \cdot n)$. Debido a que en cada iteración del bucle se elimina un robot y una tarea (ver línea 7) la complejidad del bucle que va de las líneas 4 a 7 es $O(\sum_{i=0}^{min(n,m)-1}(n-i)(m-i))$. Por tanto, la complejidad algorítmica del método SBPA es igual a $O(m+\sum_{i=0}^{min(n,m)-1}(n-i)\cdot(m-i)) \subset O(min(n,m)\cdot n\cdot m)$. A partir de ahora siempre se utilizará el orden de complejidad más restringido, en este caso $O(m+\sum_{i=0}^{min(n,m)-1}(n-i)\cdot(m-i))$, para comparar los algoritmos propuestos.

Para obtener la complejidad en las comunicaciones, es decir, el orden de magnitud del número de mensajes enviados por los robots, se sigue el mismo razonamiento que para obtener la complejidad algorítmica. El subastador ha de enviar un mensaje a cada uno de los robots solicitando las pujas y luego cada robot ha de enviar dicha puja (ver líneas 1-3), esto supone una complejidad de $O(n)$. El envío de los mensajes de asignación de tarea al robot seleccionado supone una complejidad en las comunicaciones de $O(min(n, m))$. Por tanto, la complejidad de las comunicaciones del algoritmo SBPA es de $O(n + min(n, m)) \subset O(n)$.

3.3.2. *Earliest Deadline First Auction* (EDFA)

El algoritmo EDF (*Earliest Deadline First*) es un método muy conocido para planificar tareas en procesadores para entornos de tiempo real [93, 59]. En este tipo de escenarios, las tareas (procesos) se ordenan según su *deadline*, de manera que las tareas con un *deadline* más próximo son procesadas en primer lugar. El mismo concepto ha sido utilizado en este trabajo para implementar la estrategia de *Earliest Deadline First Auction* (EDFA), que se puede ver en el algoritmo 5. El subastador central ejecuta este método cada vez que una nueva tarea aparece o cuando un robot finaliza la suya. En primer lugar, ordena las tareas por *deadline* y envía a todos los robots una solicitud de puja para la primera tarea, es decir, para la tarea con un *deadline* más próximo. Los robots que estén libres y que puedan finalizar la tarea antes de su *deadline* pujan por ella utilizando su tiempo previsto de ejecución. Finalmente, el subastador selecciona el mejor robot (el robot con un tiempo estimado de ejecución menor) para ejecutar la tarea. Si aún hay más tareas pendientes, se selecciona la siguiente tarea con un *deadline* más próximo volviendo a ejecutarse de nuevo el mismo proceso. Como se puede observar, se trata de un método similar al SSA, visto en el capítulo 2, con una ordenación previa de tareas.

Siguiendo el mismo razonamiento que para el algoritmo SBPA, el análisis de complejidad algorítmica del método EDFA es el siguiente: sea m el número de tareas y n el número de robots, la complejidad del algoritmo de ordenación de tareas, utilizando

Algoritmo 5 Algoritmo EDFA

Entrada: T=Lista de tareas no asignadas
1: ordenar T por *deadline* más próximo (EDF)
2: **para todo** $t \in T$ **hacer**
3: Solicitar puja para t a todos los robots libres
4: Seleccionar la mejor puja
5: Enviar mensaje de asignación al robot seleccionado
6: **fin para**

los métodos *marge sort* o *binary tree sort*, es $O(m \cdot log(m))$. Una vez ordenadas las tareas por *deadline*, el subastador debe comprobar cada puja para seleccionar la mejor, teniendo en cuenta que los robots que ya han sido asignados a una tarea no se han de volver a considerar. De esta manera, en cada iteración un robot es asignado a una tarea y la complejidad del algoritmo es igual a: $O(m \cdot log(m) + \sum_{i=0}^{min(n,m)-1}(n-i))$. Las comunicaciones en este tipo de métodos tienen el mismo orden de complejidad que en el algoritmo SBPA.

3.3.3. *Sequential Unordered Auctions* (SUA)

El algoritmo de subastas secuenciales sin orden (*Sequential Unordered Auctions*-SUA) es el método de subastas más simple que se ha tenido en cuenta en este trabajo y también tiene una estructura similar a las subastas SSA. A medida que las tareas llegan al subastador, éste inicia una nueva ronda de subastas para cada una de ellas. De esta manera, las tareas son procesadas secuencialmente siguiendo una política *First In First Out* (FIFO). Cuando se inicia una nueva subasta, los robots que son capaces de finalizar la tarea antes de su tiempo límite pujan por ella utilizando su tiempo previsto de finalización. El subastador recibe las pujas de los diferentes robots y selecciona como ganador de la subasta al robot con un tiempo de ejecución menor. Finalmente, si quedan aún más tareas para ser asignadas, se inicia un nuevo proceso de subasta. En este proceso, el subastador no tiene que tomar ninguna decisión sobre el orden en que procesar las tareas, por tanto, es similar al método EDFA pero sin necesidad de una ordenación previa por *deadline*.

El análisis de la complejidad algorítmica de este método es el siguiente: sea m el número de tareas y n el número de robots, para cada una de las m tareas el subastador ha de comprobar la puja de cada uno de los n robots. Siguiendo el mismo razonamiento que en el algoritmo EDFA, cada vez que se procesa una tarea también se selecciona un robot, por tanto la complejidad del algoritmo SUA es: $O(\sum_{i=0}^{min(n,m)-1}(n-i)) \subset O(min(n,m) \cdot n)$. Las comunicaciones tienen el mismo orden de complejidad que en el algoritmo EDFA.

3.3.4. *Earliest Deadline First Best Pair* (EDFBP)

En esta sección se propone un nuevo algoritmo de subastas híbrido entre EDF y SBPA llamado *Earliest Deadline First Best Pair* (EDFBP). Los resultados experimentales, expuestos en la sección 3.5, muestran que el algoritmo SBPA presenta mejores resultados que el algoritmo EDFA. Por otra parte, se ha demostrado que el orden de complejidad algorítmica de SBPA es superior al de EDFA. El objetivo del método EDFBP es conseguir resultados similares a los de SBPA pero con una complejidad algorítmica lo más próxima posible a la del algoritmo EDFA. Para conseguir este objetivo se ha diseñado el algoritmo 6, donde en primer lugar el subastador ordena por *deadline* todas las tareas pendientes (línea 2). Una vez hecho esto, se selecciona un subconjunto de tareas (línea 4) y se aplica el algoritmo SBPA (algoritmo 4) sólo sobre ese subconjunto. Para ello, se divide el conjunto total de tareas pendientes T en varios subconjuntos consecutivos en función de su *deadline*. El algoritmo 7 muestra como se realiza esta partición mediante el parámetro β, donde β representa el porcentaje de las tareas totales que se procesará como una unidad por el algoritmo SBPA. Cuando $\beta = 1$, EDFBP se comporta como el método SBPA y cuando $\beta = 0$ EDFBP se convierte en el algoritmo EDFA.

El orden de complejidad algorítmica de EDFBP se calcula de la siguiente forma: sea m el número de tareas y n el número de robots. Con el objetivo de simplificar el proceso, se asume que todas las particiones de T tienen el mismo número de tareas. De igual manera que para el algoritmo EDFA, el proceso de ordenación de las tareas por

Algoritmo 6 Algoritmo EDFBP

Entrada: T=Lista de tareas no asignadas
1: N= número de tareas en T
2: Ordenar T por *deadline* más próximo (EDF)
3: **mientras** $T \neq \emptyset$ **hacer**
4: $TW \leftarrow tasksOfInterest(T, N)$
5: $SBPA(TW)$
6: $T \leftarrow T - TW$
7: **fin mientras**

Algoritmo 7 Algoritmo tasksOfInterest

Entrada: T=Lista de tareas no asignadas
Entrada: N=Tamaño de tareas pendientes al inicio del algoritmo EDFBP
1: $N_g \leftarrow MIN$(número de tareas en $T, \lceil \beta \cdot N \rceil$)
2: $TInt \leftarrow T[1..N_g]$
3: **devolver** TInt

deadline tiene una complejidad de $O(m \cdot long(m))$. El algoritmo SBPA se ejecuta sobre $(\beta \cdot m)$ tareas y por tanto su complejidad es igual a $O(\sum_{i=0}^{min(\beta m,n)-1} (\beta m - i)(n - i))$. Consecuentemente, la complejidad algorítmica de EDFBP es igual a: $O(m \cdot log(m) + \frac{1}{\beta} \sum_{i=0}^{min(\beta m,n)-1} (\beta m - i)(n - i))$, siempre que $\beta \geq 1$. En el caso en el que $m < \frac{1}{\beta}$, la complejidad del algoritmo SBPA se debería multiplicar por m en lugar de por $\frac{1}{\beta}$.

Se puede observar que la complejidad del algoritmo EDFBP para valores de $\beta < 1$ es menor que la complejidad de SBPA. Modificando el parámetro β se puede también modificar la complejidad del algoritmo y ajustarlo a la capacidad de cálculo del subastador. La figura 3.2 muestra el aumento de complejidad algorítmica del método SBPA con respecto a EDFBP para diferentes valores de β utilizando 20 robots, $n = 20$. Estos resultados han sido obtenidos dividiendo el orden de complejidad de SBPA por el de EDFBP, por tanto, cuanto mayor sea el resultado de esta división, menor complejidad relativa tendrá EDFBP con respecto a SBPA. Como se puede observar, a medida que disminuye el valor β, la complejidad relativa de EDFBP respecto a SBPA es cada vez menor. Además, en todos los casos, existe un número óptimo de

tareas para el que la reducción de complejidad es máxima. Cuanto mayor sea el valor de β mayor será la complejidad de algoritmo EDFBP, pero también mejorará su rendimiento (número de tareas que han cumplido su *deadline*). Tal como se verá en los resultados experimentales, EDFBP alcanza resultados similares a los obtenidos con SBPA para valores de β muy inferiores a 1 y, por tanto, con una disminución importante en el tiempo de cálculo.

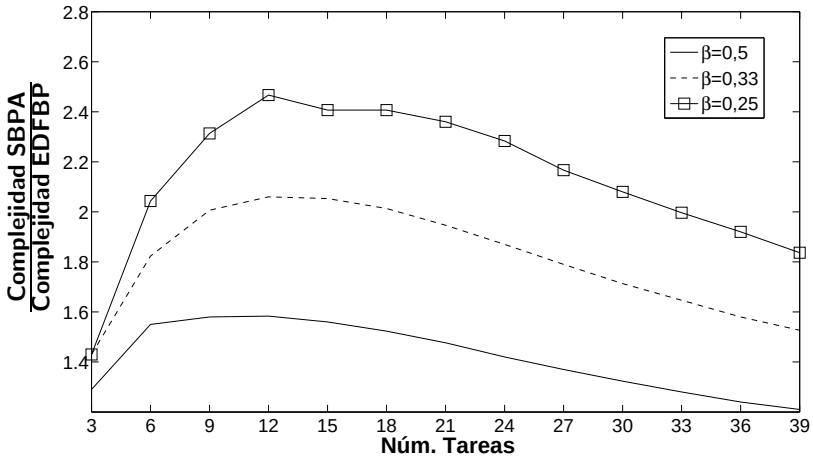


Figura 3.2: Incremento de la complejidad algorítmica relativa de SBPA respecto a EDFBP utilizando 20 robots ($n = 20$) para diferentes valores de β.

La tabla 3.1 muestra un resumen del orden de complejidad algorítmica de los sistemas presentados. Tal como se puede observar, el método algorítmicamente más complejo es el SBPA, seguido de EDFBP, EDFA y SUA.

3.4. Planificación local de tareas: problema TSP

En esta sección se estudian los métodos que permiten asignar múltiples tareas a un mismo robot. Al tener múltiples tareas asignadas los robots han de planificarlas, es decir, deben decidir su orden de ejecución. A partir de ahora se denominará planificación local (*Local Planning*-LP) a este proceso de decisión. En la mayoría de ocasiones, como en el caso de las tareas de recogida de objetos, esta planificación implica tener

Swarm	$O(m)$
SBPA	$O(m + \sum_{i=0}^{min(n,m)-1} (m-i) \cdot (n-i))$
EDFBP	$O(m \cdot log(m) + \frac{1}{\beta} \sum_{i=0}^{min(\beta m,n)-1} (\beta m - i) \cdot (n-i))$
EDFA	$O(m \cdot log(m) + \sum_{i=0}^{min(n,m)-1} (n-i))$
SUA	$O(\sum_{i=0}^{min(n,m)-1} (n-i))$

Tabla 3.1: Orden de complejidad algorítmica para problemas ST. m es el número de tareas, n es el número de robots.

que visitar una serie de posiciones del espacio. Se ha de notar que, estas estrategias no abordan problemas de tipo MT ya que las tareas no se ejecutan simultáneamente.

Para implementar la planificación local, cada robot tiene inicialmente asignada una planificación P, consistente en una lista con todos sus puntos objetivo (objetos a recoger) ordenados. Por tanto, P es el camino o plan que el robot ha de seguir. Cuando un robot recibe una solicitud del subastador para realizar una puja sobre la tarea t, el robot intentará añadir la tarea t a P creando una planificación provisional P', de manera que el tiempo de finalización del nuevo camino P' sea tan corto como sea posible y que todas las tareas en P' verifiquen su *deadline*. En el caso en que no sea posible incluir t y cumplir el *deadline* de las tareas ya existentes, el robot no pujará. Si t se puede incluir en P, el valor de la puja por t ($b(t)$) será igual al nuevo tiempo previsto para finalizar todas las tareas, esto es, la puja será igual a:

$$b(t) = \sum_{t_j \in P'} tExpected(t_{j-1}, t_j) \tag{3.3}$$

donde $tExpected(a, b)$ retorna el tiempo estimado que el robot necesitaría para ir de la tarea a a la tarea b, la tarea t_0 representa la posición actual del robot. Se han probado otros criterios para realizar la puja, como por ejemplo pujar utilizando la diferencia entre el tiempo estimado para finalizar P y el tiempo de P', pero se ha comprobado que la ecuación 3.3 siempre producía los mejores resultados. Una vez los robots han realizado sus pujas, el subastador seleccionará al mejor robot (aquel que

tenga un valor menor de $b(t)$) usando cualquiera de las estrategias ya explicadas (SUA, EDFA o SBPA). Finalmente, se asigna el plan provisional P' al robot seleccionado.

La minimización del tiempo de ejecución del camino P' es un problema NP-hard llamado el problema del viajante de comercio (*traveling salesman problem*) con *deadlines* en espacios métricos ($\Delta - D_L TSP$). Tal como se explicó en el capítulo 2, existe un gran número de métodos y algoritmos para intentar solucionar este problema problema [13, 10]. En nuestro caso se ha utilizado la siguiente estrategia, similar al algoritmo del vecino más próximo, para obtener P' a partir de P y t. Se inserta t entre cada 2 tareas ya existentes en P y de entre todos los caminos resultantes se selecciona el que tenga un menor tiempo de ejecución previsto. En este proceso sólo se tendrán en cuenta aquellos caminos en los que el *deadline* de todas sus tareas se verifique. La complejidad de este algoritmo se puede calcular de la siguiente manera: sea m el número de tareas en P, entonces para crear P' se han de realizar $m+1$ comprobaciones, tantas como posibles lugares en los que se puede insertar la nueva tarea t, y como máximo $m+1$ comprobaciones más para verificar que todas las tareas siguen cumpliendo el *deadline*. Por tanto, la complejidad del algoritmo es $O(m^2)$. Este método es muy simple y tiene una complejidad algorítmica relativamente baja, aunque la distancia total recorrida suele ser mucho mayor a la obtenida con otros algoritmos.

Permitir múltiples tareas por robot supone tener que modificar el método SBPA, el algoritmo 8 muestra el método resultante de estas modificaciones. Tal como se puede observar, el subastador después de cada asignación ha de volver a solicitar una nueva puja a todos los robots (líneas 2-4) ya que el valor de la puja de un robot para una tarea depende de las asignaciones realizadas en iteraciones anteriores. Los métodos SUA y EDFA también han tenido que ser modificados, de manera que el subastador solicite las pujas a todos los robots antes de procesar cada una de las tareas. Los métodos resultantes de realizar estas modificaciones para permitir múltiples tareas por robot se llamarán a partir de ahora: SUA-LP, EDFA-LP y SBPA-LP. Los métodos sin planificación local seguirán con el mismo nombre utilizado hasta ahora. Tal como

se explica en la sección 3.5, la planificación local que realiza cada robot provoca que los resultados de los experimentos realizados con EDFBP sean siempre similares a los del algoritmo SBPA, por este motivo, no se ha implementado la versión LP del algoritmo EDFBP. Tampoco se ha implementado una versión para múltiples tareas de los algoritmos de *swarm*, ya que la planificación local implica que los robots tengan una capacidad de cálculo que no es compatible con la simplicidad que exige este tipo de algoritmos.

Algoritmo 8 Algoritmo SBPA-LP

Entrada: T=Lista de tareas no asignadas
1: **repetir**
2: **para todo** $t \in T$ **hacer**
3: Solicitar puja a todos los robots
4: **fin para**
5: Seleccionar la mejor pareja robot-tarea (mejor puja).
6: Enviar un mensaje de asignación al robot seleccionado.
7: **hasta que** no haya más tareas pendientes o ningún robot pueda incluir una nueva tarea

Todas estas modificaciones también implican que la complejidad algorítmica de los métodos SBPA-LP, EDFA-LP y SUA-LP se incremente respecto a las versiones sin planificación local. El cálculo de la complejidad de los algoritmos LP se realiza de la siguiente manera: sea m el número de tareas y n el número de robots. En todos estos métodos cada vez que se procesa una tarea se han de tener en cuenta los n robots, es decir, en el peor de los casos, en cada iteración el número de robots se mantendrá constante. Cada vez que se quiera evaluar la idoneidad de un robot, se deberá aplicar el algoritmo de planificación, con una complejidad del orden $O(i^2)$, siendo i el número de tareas que ya le han sido asignadas hasta el momento. Siguiendo el mismo razonamiento que en entornos sin planificación local, la tabla 3.2 muestra el nuevo orden de complejidad algorítmica de los diferentes métodos.

El número de mensajes que se han de enviar también ha aumentado respecto a algoritmos sin planificación local en los robots. El subastador, en los métodos EDFA-LP y SUA-LP, ha de enviar para cada tarea una solicitud a todos los robots (1

SBPA-LP	$O(m + n\sum_{i=0}^{m-1}((m-i)\cdot(i^2+1)))$
EDFA-LP	$O(m\cdot log(m) + n\sum_{i=0}^{m-1}(i^2+1))$
SUA-LP	$O(n\sum_{i=0}^{m-1}(i^2+1))$

Tabla 3.2: Complejidad algorítmica de los métodos ejecutados en el subastador en entornos con planificación local (LP). m es el número de tareas y n el número de robots.

mensaje), esperar a recibir las n pujas (una de cada robot) y finalmente enviar el mensaje de asignación al robot ganador (1 mensaje). Por tanto, en el peor de los casos, la complejidad de las comunicaciones será $O(m\cdot(n+2)) \subset O(n\cdot m)$, donde m es el número de tareas. Este razonamiento es muy similar al que se ha de seguir para el método SBPA-LP, donde en cada iteración del algoritmo 8 el subastador ha de enviar un mensaje solicitando pujas para las tareas aún sin asignar (1 mensaje). Por tanto, su complejidad también será $O(n\cdot m)$, aunque la longitud de los mensajes será mucho mayor, al tener que solicitar pujas para un conjunto de tareas y no para una sola, como sucedía en los algoritmos EDFA-LP o SUA-LP.

3.5. Resultados experimentales

En esta sección se explican los experimentos llevados a cabo para validar todas las estrategias explicadas en este capítulo: *swarm*, PSW, SBPA, EDFA, SUA y EDFBP, tanto en entornos sin planificación local como en aquellos en los que cada robot planifica las múltiples tareas que puede tener asignadas.

3.5.1. Diseño de los experimentos

En todos los experimentos realizados se ha utilizado la tarea de *foraging* con límites temporales explicada en la sección 3.1. Para ello se han utilizado dos simuladores: RoboSim, sobre el que se han obtenido la mayor parte de los resultados, y Player/Stage

[29]. El simulador RoboSim ha sido desarrollado como parte de esta tesis con el objetivo de simular el comportamiento de entornos con un número elevado de robots y con gran cantidad de tareas. Después de cada unidad de tiempo, RoboSim actualiza la posición de todos los robots a partir de sus características cinemáticas y procesa todos los eventos sucedidos en este intervalo: nuevas tareas en el entorno, tareas finalizadas con éxito, tareas que han llegado a su *deadline* sin ser ejecutadas, etc. Para reducir el tiempo de simulación, se supone que los robots no pueden colisionar entre sí, ni con ningún otro objeto del entorno. Por tanto, no es necesario implementar ningún algoritmo de evitación de obstáculos. Además, los robots no se mueven a no ser que tengan alguna tareas asignada. Así pues, se ha de considerar a RoboSim como un simulador no realista pero muy útil como herramienta para comparar el rendimiento global de las diferentes estrategias implementadas. En el anexo B se puede encontrar una descripción mucho más detallada de las diferentes características y opciones de RoboSim. A fin de validar los resultados obtenidos con RoboSim, se han llevado a cabo experimentos más precisos con un número menor de robots y de tareas utilizando el simulador Player/Stage. Player/Stage es un simulador realista, muy conocido y utilizado por gran cantidad de grupos de investigación. En el anexo A se explican detalladamente las estrategias de evitación de obstáculos y la arquitectura de control implementadas en este simulador.

3.5.2. Entornos sin restricciones temporales

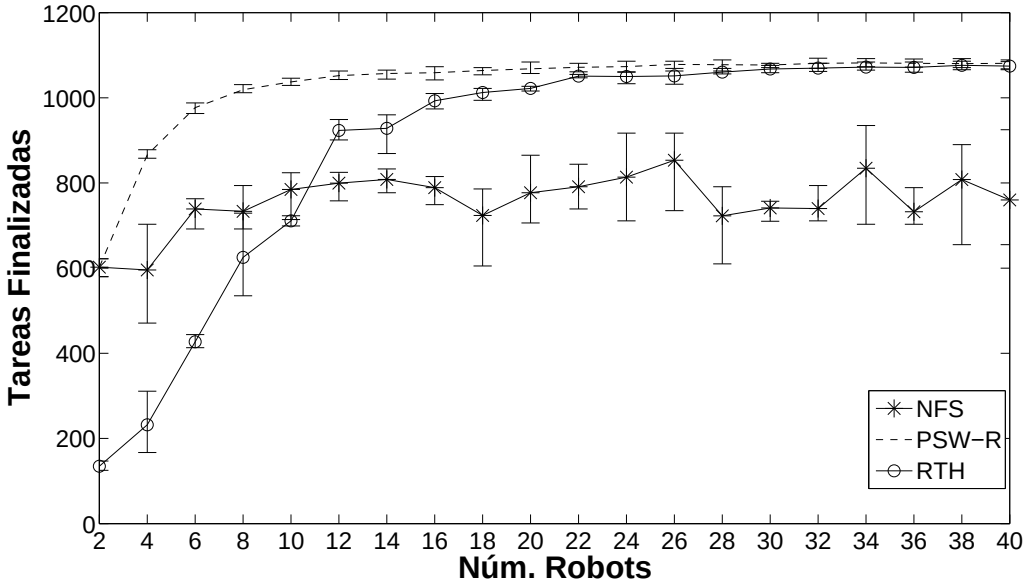
En esta sección se explican los resultados obtenidos con tareas sin restricciones temporales, es decir sin un *deadline* asociado a las mismas. Todos los resultados mostrados corresponden a experimentos llevados a cabo en el simulador RoboSim, utilizando 3 valores diferentes de λ para el proceso de Poisson que determina la llegada de nuevos objetos al entorno: $\lambda = 0,05$, $\lambda = 0,1$ y $\lambda = 0,3$. El número inicial de objetos en el entorno se ha fijado en 20, 30 ó 40. Las gráficas muestran el valor medio de los resultados obtenidos para estas 3 condiciones iniciales. Los resultados de los experimentos muestran que la posición inicial de los robots y de los objetos

tiene muy poca influencia sobre el resultado final. El parámetro D tiene un valor infinito en estos experimentos, es decir, siempre se tienen en cuenta todas las tareas independientemente de la distancia a la que se encuentren de los robots.

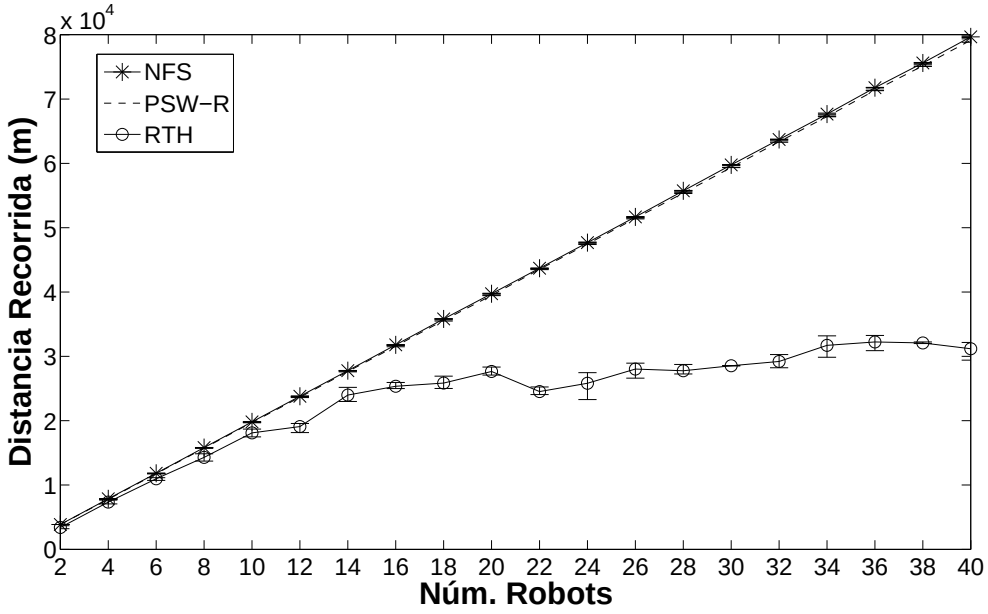
Todos los robots tienen siempre las mismas características con una velocidad de 0,2 m/s, su posición inicial es aleatoria y su número variará entre 2 y 40. Las dimensiones del entorno son de 160mx160m, donde sólo estarán los objetos a procesar y los robots. En total se han efectuado 1.920 simulaciones, cada una de ellas de 10.000 segundos simulados, durante las cuales se han recogido un total de 270.000 objetos.

Resultados de los algoritmos basados en *swarm*

La figura 3.3 muestra los resultados obtenidos utilizando los métodos de *swarm* NFS, PSW-R con un valor de A igual a 10m y RTH con un valor de θ igual a 0,2. Los valores de los parámetros A y θ se han ajustado manualmente para obtener los mejores resultados posibles en cada caso. El proceso de Poisson para las llegadas tiene un valor de λ igual a 0,1. Se han repetido los experimentos con distintos valores de λ habiéndose llegado a resultados muy similares a los que muestra la figura 3.3. El valor de las barras verticales representa el valor máximo y mínimo obtenido en los experimentos. Tal como se puede observar, PSW-R es el método con un mayor número de tareas finalizadas, mostrando, de esta manera, la validez del nuevo algoritmo pseudo-aleatorio. La diferencia es más marcada en favor de este método cuando el número de robots es inferior a 20. Para grupos con un gran número de robots PSW-R y RTH presentan resultados bastante similares, en cambio, el método NFS es el que muestra los peores resultados a partir de 12 robots. También se ha de destacar la gran variabilidad en los resultados que muestra RTH, en cambio los resultados con PSW-R tiene un valor máximo y mínimo muy similar. En cuanto a la distancia recorrida, RTH mejora los resultados de los otros dos métodos, sobre todo con un número elevado de robots. Cabe destacar que el valor de A es muy estable y prácticamente no depende del entorno, en todos los escenarios analizados el mejor valor de A que se ha podido encontrar siempre ha sido igual a 10m. Por contra, con el algoritmo RTH



(a) Número de tareas finalizadas.



(b) Distancia total recorrida por los robots.

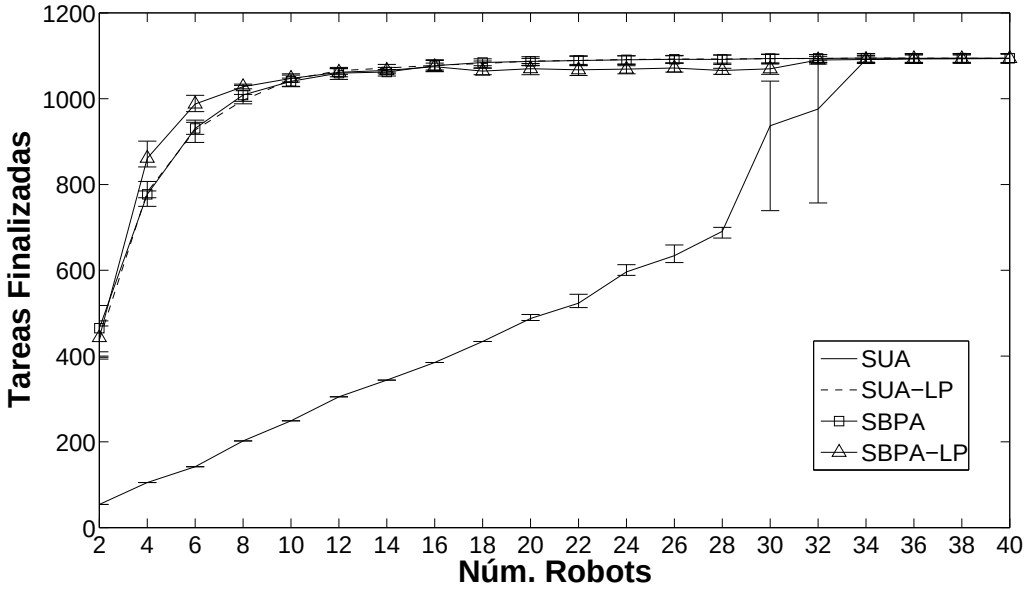
Figura 3.3: Resultados con tareas sin restricciones temporales utilizando los métodos basados en *swarm*: NFS, PSW-R con $A = 10m$ y RTH con $\theta = 0, 2$. $\lambda = 0, 1$.

cada escenario presentaba un valor de θ óptimo muy diferente. En la sección 3.5.3 se analizarán más detalladamente el motivo de esta variabilidad.

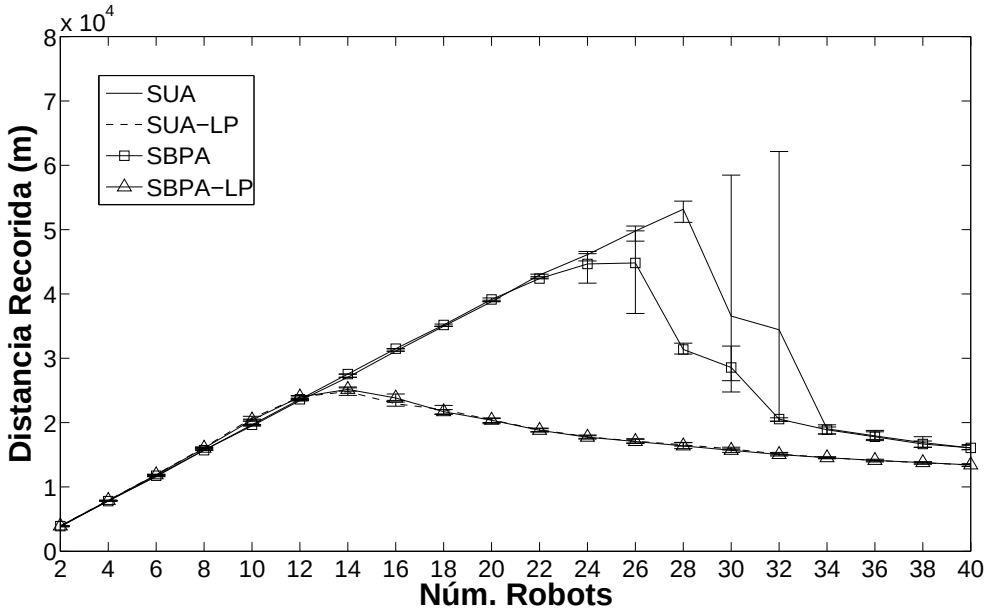
En la figura 3.3 no aparecen los resultados del algoritmo PSW-D ya que presenta un comportamiento muy similar al PSW-R. En todos los entornos analizados, PSW-R obtiene resultados iguales o con una mejora de entre un 2 % y un 8 % respecto a PSW-D. En el apartado de pruebas con restricciones temporales se analizarán más detalladamente esta diferencias.

Resultados de los algoritmos basados en subastas

La figura 3.4 muestra los resultados obtenidos utilizando los métodos de subastas: SBPA y SUA, tanto usando planificación local como sin ella. Las tareas no tienen restricciones temporales y aparecen en el entorno siguiendo un proceso de Poison con una tasa de llegadas $\lambda = 0{,}1$. No se muestran los resultados del método EDFA ya que al no haber *deadlines* son iguales a los obtenidos con el algoritmo SUA. La figura 3.4(a) muestra el número total de tareas finalizadas y la figura 3.4(b) la suma de las distancia recorrida por todos los robots. La figura 3.5(a) muestra el número total de tareas finalizadas sin restricciones temporales y con $\lambda = 0{,}05$ y la figura 3.5(b) muestra la distancia total recorrida por los robots en este mismo escenario. En ambos casos, existe un número de robots a partir del cual la distancia total recorrida empieza a disminuir, siendo menor en si $\lambda = 0,1$. Estos resultados muestran que, cuando no se utiliza planificación local en el método SUA, el número de tareas finalizadas se incrementa casi linealmente con respecto al número de robots. El método SBPA no se ve afectado por la planificación local de tareas en cada robot, ya que el número de tareas finalizadas, tanto con planificación local como sin ella, es bastante similar. En cambio, la planificación local mejora claramente los resultados de las subastas SUA, llegando a tener un número similar de tareas finalizadas a las obtenidas con SBPA. Así, se demuestra que incrementar la complejidad en el subastador es equivalente, en cuanto a las tareas finalizadas, a incrementar la complejidad en cada uno de los robots.

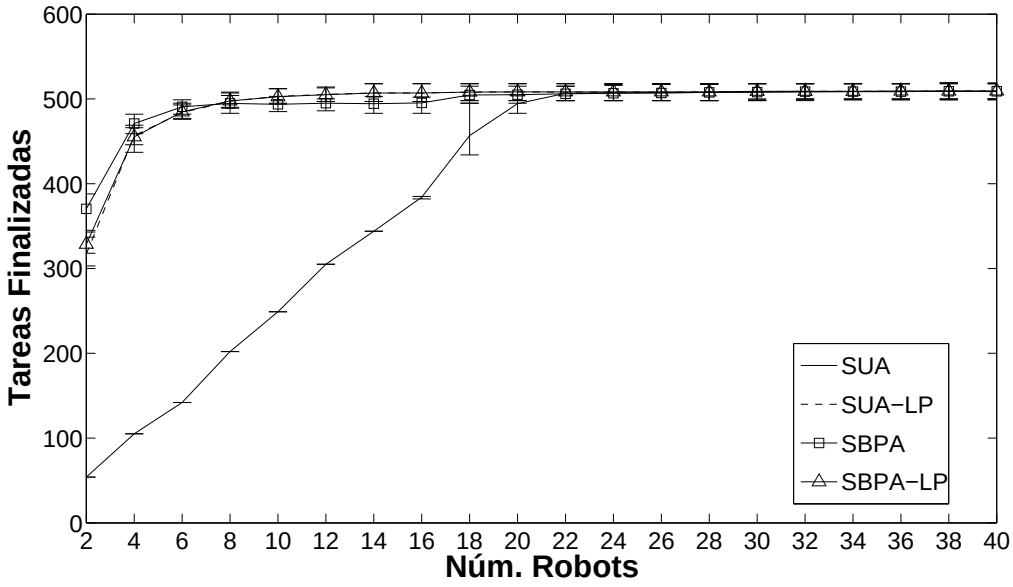


(a) Número total de tareas finalizadas.

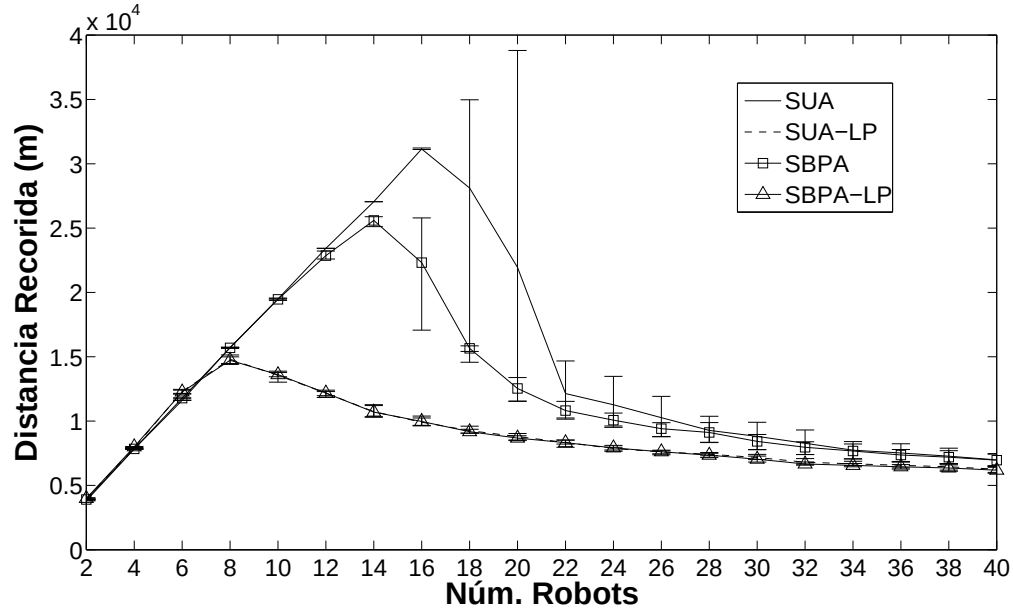


(b) Distancia total recorrida por los robots.

Figura 3.4: Resultados con tareas sin restricciones temporales utilizando los métodos SUA, SBPA, SUA-LP y SBPA-LP. $\lambda = 0,1$.



(a) Número de tareas finalizadas.



(b) Distancia total recorrida por los robots.

Figura 3.5: Resultados con tareas sin restricciones temporales utilizando los métodos SUA, SBPA, SUA-LP y SBPA-LP. $\lambda = 0{,}05$.

Comparación entre los métodos basados en subastas y los basados en *swarm*

A continuación se realiza un análisis comparativo entre los métodos basados en subastas y las estrategias basadas en *swarm*. La figura 3.6 muestra la distancia total recorrida por todos los robots con $\lambda = 0,3$ utilizando las estrategias SBPA-LP, PSW-R y RTH. La distancia total recorrida por los robots con la estrategia SBPA es muy similar a la obtenida utilizando PSW-R o RTH y no se ha representado para simplificar la figura. Al comparar estos resultados con los obtenidos en entornos con $\lambda = 0,1$, se observa que ahora la estrategia RTH no reduce la distancia total recorrida por los robots (véase las figura 3.3(b)). Por tanto, para valores elevados de λ no se produce una reducción de la distancia en función del número de robots cuando no se utiliza planificación local, o al menos no es apreciable para menos de 40 robots. El único algoritmo que muestra la disminución antes observada es el método SBPA-LP gracias a la planificación del camino a seguir realizada por cada robot.

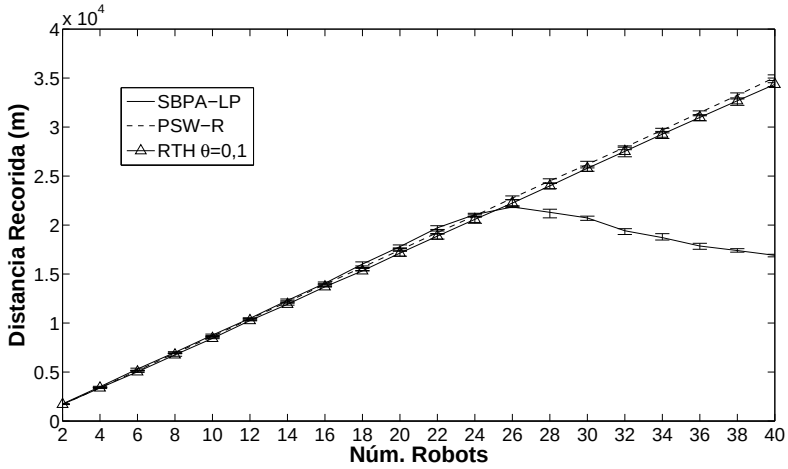


Figura 3.6: Distancia total recorrida por los robots con $\lambda = 0{,}3$.

Si se comparan los resultados de las gráficas 3.3 y 3.4, se observa que los algoritmos PSW-R, SBPA y los métodos que utilizan planificación (LP) presentan resultados similares en cuanto al número de tareas realizadas, sobre todo a partir de 4 robots.

En cambio, los algoritmos basados en subastas reducen substancialmente la distancia total recorrida por los robots, especialmente cuando su número es elevado. Por tanto, los algoritmos basados en subastas aúnan el incremento de las tareas finalizadas que proporcionan los algoritmos PSW-R y PSW-D con el ahorro de distancia recorrida del algoritmo de *response threshold* RTH.

3.5.3. Entornos con restricciones temporales

Las simulaciones realizadas sobre RoboSim para tareas con restricciones temporales tienen las mismas características (valores de λ, tiempo de simulación, dimensiones del entorno, etc.) que las simulaciones sin restricciones explicadas en el apartado 3.5.2. La única diferencia se encuentra en las características de las tareas, para ello se han utilizado 3 tipos diferentes de tareas o configuraciones:

- Configuración 1 (Tareas uniformes): todas las tareas tienen el mismo peso (1 unidad de peso) y el mismo *deadline* (250 s)
- Configuración 2 (*Deadline* aleatorio): todas las tareas tienen el mismo peso 1 unidad pero los *deadlines* de cada una de ellas se han generado siguiendo una variable aleatoria uniforme de entre 100 y 500 segundos.
- Configuración 3 (Tareas híbridas): se trata de una configuración similar a la 2 pero con una probabilidad de 0,2 de que la nueva tarea que aparezca en el entorno no tenga restricciones temporales. Por tanto, en un mismo entorno coexisten tareas con y sin *deadline*.

Resultados de los algoritmos basados en *swarm*

La figura 3.7 muestra el número de objetos recogidos antes de su *deadline* utilizando los algoritmos PSW-D con $\theta = 0,05$, PSW-R con $A = 10m$ y RTH con $\theta = 0,05$. Para poder comparar los métodos de subastas con los basados en *swarm*, el gráfico también incluye la estrategia SBPA, al ser ésta la mejor de las estrategias basadas en

subastas. La configuración utilizada es la 1, en la que todas las tareas tienen el mismo *deadline*, con $\lambda = 0,1$. Al igual que para el resto de gráficas presentadas en esta sección, todos los resultados han sido obtenidos utilizando el simulador RoboSim. No se muestran las gráficas con las distancias totales recorridas por los robots ni las barras verticales con los valores máximos y mínimos ya que el comportamiento es muy similar a las ya presentadas para entornos sin restricciones temporales. Como se puede observar, el algoritmo SBPA presenta los mejores resultados seguido de PSW-R/D y RTH. La diferencia entre PSW-R y PSW-D se produce principalmente cuando hay pocos robots en el entorno y es especialmente significativa cuando la frecuencia de aparición de nuevas tareas es baja. Por ejemplo, la figura 3.8 muestra los resultados de las dos estrategias cuando λ es igual a 0,05 , en este caso la diferencia es mucho más notable, incrementándose en un 8 % de media el número de tareas cuando se utiliza la estrategia PSW-R.

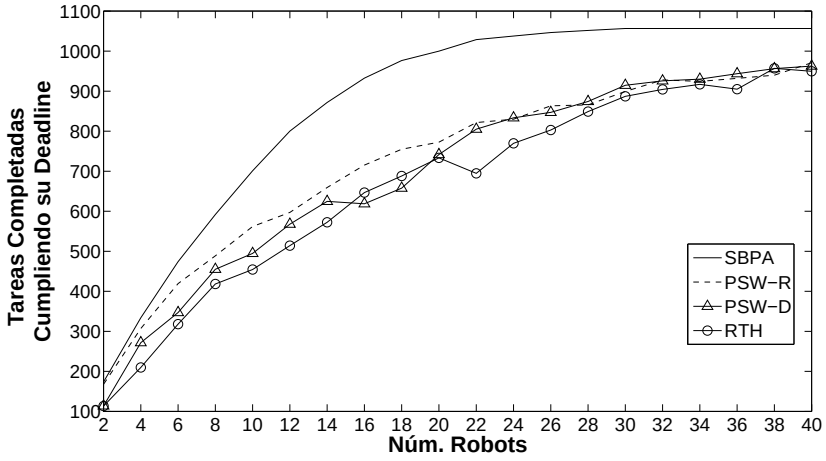


Figura 3.7: Número de tareas finalizadas antes de su *deadline* utilizando SBPA, PSW-D con $\theta = 0,05$, PSW-R con $A = 10m$ y RTH con $\theta = 0,05$. Se han utilizado tareas uniformes (configuración 1) y $\lambda = 0,1$.

La figura 3.9 muestra el número total de tareas finalizadas antes de su *deadline* utilizando los algoritmos PSW-R con $\theta = 0,033$, PSW-R con $A = 10m$ y RTH con $\theta = 0,1$, en la configuración 2 (*deadlines* aleatorios) y $\lambda = 0,3$. Tal como se puede

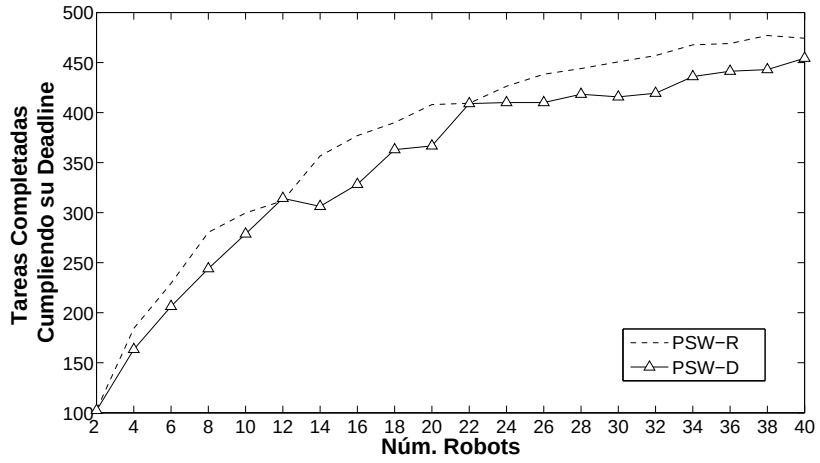


Figura 3.8: Número de tareas finalizadas antes de su *deadline* utilizando PSW-R con $A = 10m$ y PSW-D con $\theta = 0,025$. Se han utilizado tareas uniformes (configuración 1) y $\lambda = 0,05$.

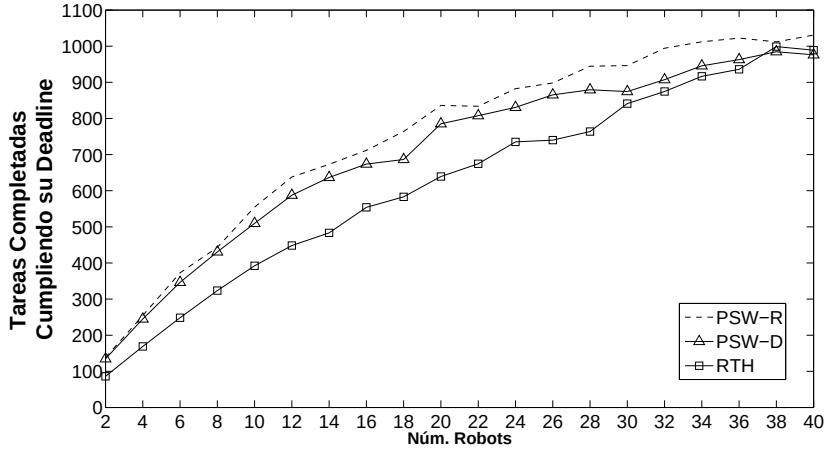


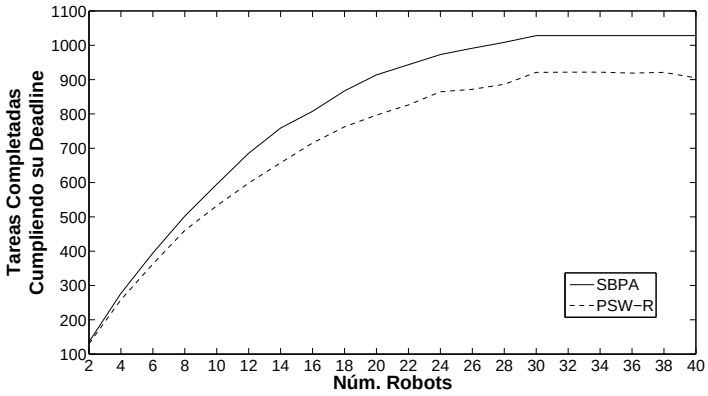
Figura 3.9: Número de tareas finalizadas antes de su *deadline* utilizando PSW-R con $A = 10m$, PSW-D con $\theta = 0,033$ y RTH con $\theta = 0,1$. Se han utilizado el configuración 2 (*deadline* aleatorio) y $\lambda = 0,3$.

ver, RTH es el método que presenta los peores resultados, en cambio el algoritmo PSW-R es el que consigue finalizar un mayor número de tareas.

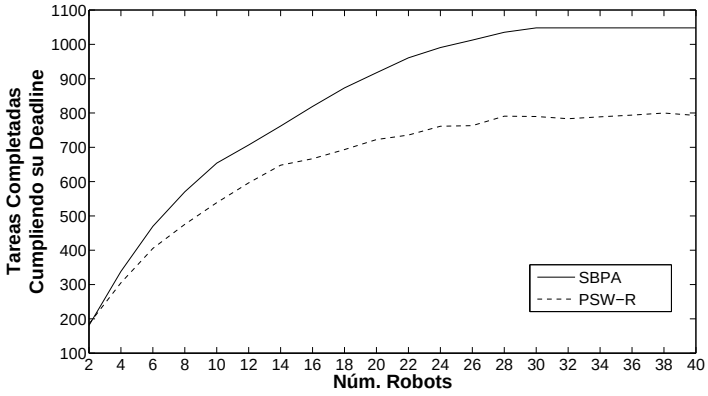
La gráfica 3.10 compara el rendimiento del mejor algoritmo *swarm* implementado (PSW-R), con el mejor algoritmo de subastas sin planificación local (SBPA) en la configuración con *deadlines* aleatorios (configuración 2) para diferentes valores de λ. El mejor valor del parámetro A del algoritmo PSW-R siempre ha sido 10m. En esta figura se observa como el algoritmo SBPA mejora siempre los resultados de PSW-R, con una forma siempre similar a la ya vista en la figura 3.7. También se ha de notar que la diferencia menos significativa entre SBPA y PSW-R, en términos porcentuales se produce cuando el valor de λ es máximo (0,3).

Por lo que se refiere a la variabilidad de los parámetros de los métodos analizados, cabe destacar que los mejores valores de θ para el algoritmo RTH varían dependiendo de las características del entorno en el que se realizan los experimentos. Por ejemplo, para la configuración 2 con $\lambda = 0,3$ el mejor valor de θ es 0,033. Para la configuración 1 con $\lambda = 0,05$, el mejor valor es igual a 0,05, mientras que sin restricciones temporales y $\lambda = 0,1$ el mejor valor de θ es 0,2. En cambio, en todos las configuraciones analizadas, independientemente del tipo de tareas o del valor de λ, el mejor valor del parámetro A del método PSW-R es 10m. Esta estabilidad es debida a que la decisión de empezar a ejecutar una tarea no depende sólo de las tareas como pasa con RTH o PSW-D, sino que sobre todo depende del número de robots, valor conocido gracias al uso de mecanismos de comunicación.

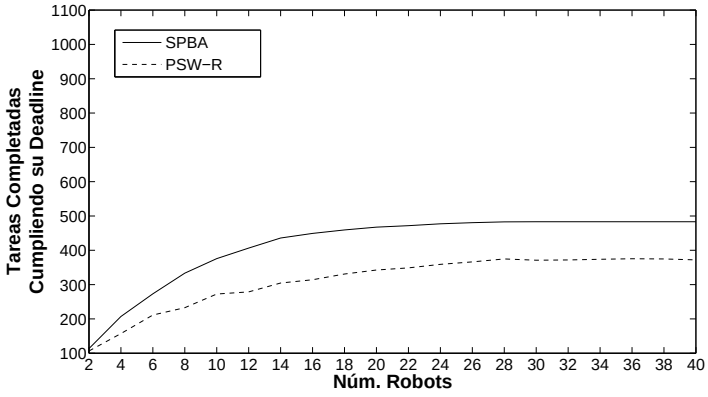
La figura 3.11 muestra el número de tareas finalizadas antes de su *deadline* utilizando PSW-R para diferentes valores de A con $\lambda = 0,1$ y utilizando la configuración 2. Se obtuvieron gráficas con formas similares para otros valores de λ. Puede observarse como los peores valores se producen para $A = \infty$, es decir, cuando se tienen en cuenta todos los robots del entorno. Por tanto, tener gran capacidad de comunicación y utilizar la información de todos los robots del entorno resulta contraproducente. Por otra parte, cuando la distancia A es muy pequeña ($A = 0,1m$), el número de tareas finalizadas decae con respecto a otros valores de A.



(a) Tareas finalizadas con $\lambda = 0,3$.



(b) Tareas finalizadas con $\lambda = 0,1$.



(c) Tareas finalizadas con $\lambda = 0,05$.

Figura 3.10: Número de tareas finalizadas antes de su *deadline* utilizando SBPA y PSW-R en la configuración 2 (*deadline* aleatorio).

Una vez constatado el efecto del parámetro A sobre los resultados del método PSW-R, se procedió a realizar un experimento para verificar la influencia del algoritmo de aprendizaje aplicado a dicho parámetro. Para ajustar el valor de A se ha utilizado el algoritmo 3 con $\alpha = 0,1$, la figura 3.12 muestra los resultados obtenidos en la configuración 2 (*deadlines* aleatorios) y con $\lambda = 0,1$, es decir se trata de la misma configuración que en la figura 3.11. Se iniciaron las simulaciones con diferentes valores de A ($A = 0,1m$ y A =1.000m) de manera que durante su ejecución el valor de A se modificó mediante el algoritmo de aprendizaje. Como se puede observar, a pesar de que el valor inicial de A estuviese muy lejos de su valor óptimo ($A = 10m$), el algoritmo de aprendizaje incrementa en todos los casos el número de tareas finalizadas con respecto a un sistema con una valor de A constante. Estos resultados muestran, a modo de ejemplo como, al igual que existen algoritmos de aprendizaje para ajustar el valor de θ en métodos de *response threshold* [136, 16], también se puede ajustar el valor de A del algoritmo PSW-R.

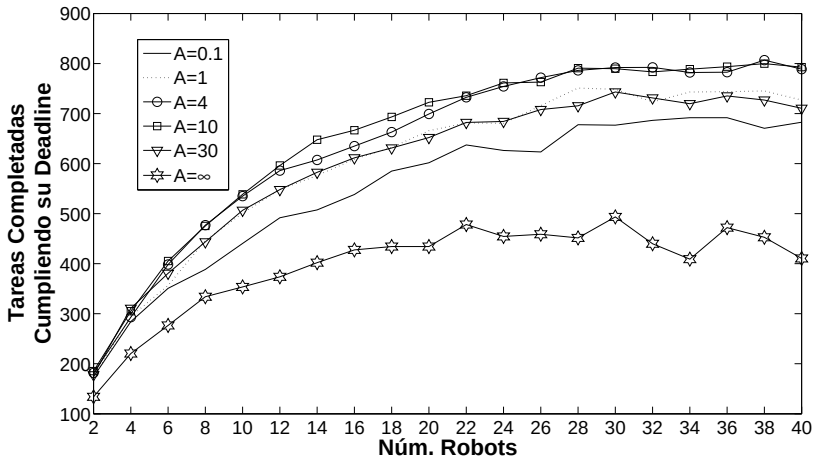


Figura 3.11: Número de tareas finalizadas antes de su *deadline* utilizando PSW-R en la configuración 2 (*deadline* aleatorio), con $\lambda = 0,1$ y diferentes valores para el parámetro A.

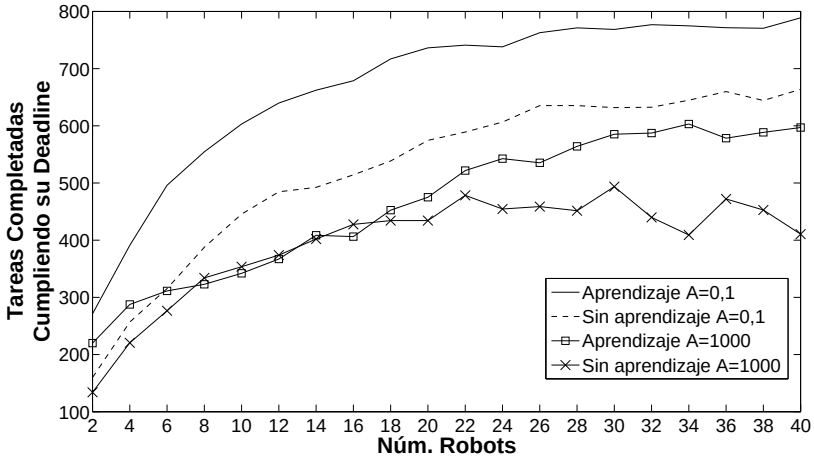
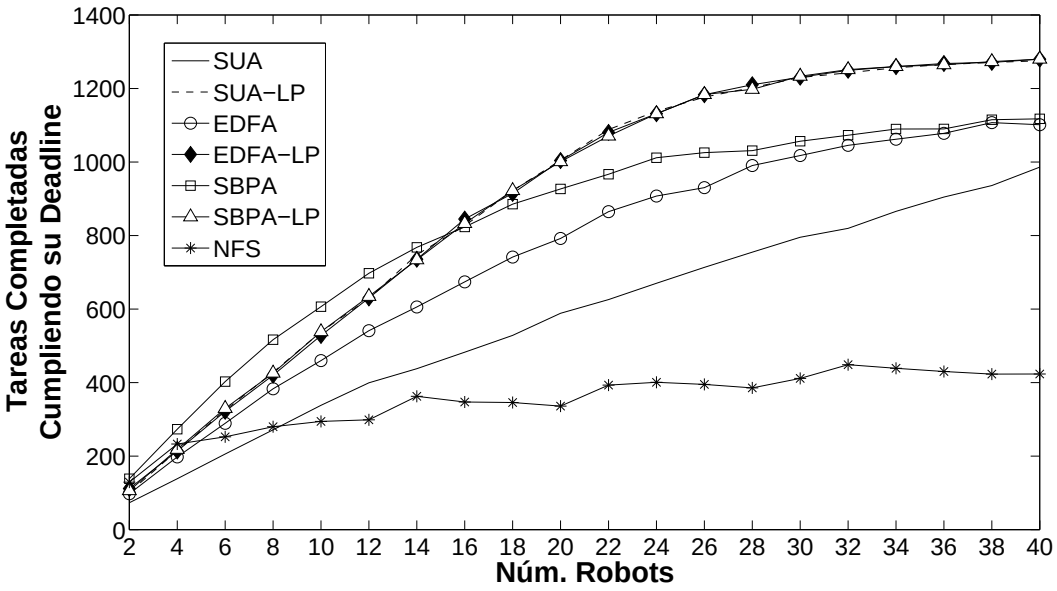


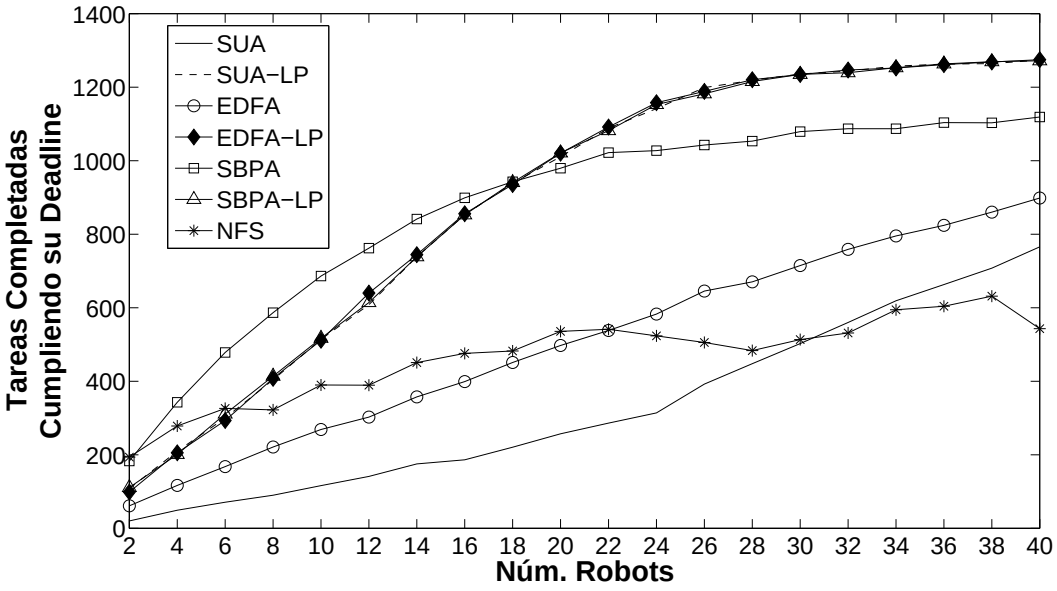
Figura 3.12: Número de tareas finalizadas antes de su *deadline* utilizando PSW-R en la configuración 2 (*deadline* aleatorio), con $\lambda = 0,1$. El algoritmo fue ejecutado utilizando tanto con como sin aprendizaje y con diferentes valores iniciales de A y con $\alpha = 0,1$.

Resultados de los algoritmos basados en subastas

A continuación se comparan los resultados obtenidos con los algoritmos de subastas, prestando especial atención a los sistemas que permiten planificación local. Los experimentos realizados usando tareas con restricciones temporales muestran que todas las estrategias basadas en subastas presentan resultados similares cuando la tasa de llegadas de nuevas tareas es baja (valores pequeños de λ), por este motivo nos centraremos en las pruebas realizadas con $\lambda = 0,3$. La figura 3.13 muestra el número de tareas finalizadas antes de su límite temporal con $\lambda = 0{,}3$ utilizando el simulador RoboSim y con dos tipos diferentes de entornos: configuración 2 (*deadlines* aleatorios) y configuración 3 (tareas híbridas). Para poder comparar los resultados con los obtenidos mediante *swarm*, el gráfico también incluye la estrategia NFS. Se ha utilizado NFS ya que otros autores [81] también utilizan una estrategia similar para comparar métodos basados en *swarm* con algoritmos de subastas. Todos los métodos de tipo LP presentan resultados muy similares, independientemente del mecanismo de subasta



(a) Configuración 2 (*deadlines* aleatorios).



(b) Configuración 3 (tareas híbridas).

Figura 3.13: Número de tareas finalizadas antes de su *deadline* con $\lambda = 0,3$ y diferentes tipos de configuraciones.

utilizado. Además, en todos los casos, el comportamiento tanto de SUA como de EDFA mejora substancialmente usando las versiones con planificación local. En cambio, el algoritmo SBPA-LP mejora los resultados de SBPA si hay más de 16 ó 18 robots, en función de la configuración. Al haber pocos robots, cada uno de ellos puede tener muchas tareas asignadas. Debido a la simplicidad del algoritmo de planificación, un incremento en el número de tareas a planificar en cada robot puede suponer que los caminos generados se alejen cada vez más del óptimo, disminuyendo el número de tareas finalizadas antes de su *deadline*. También se puede ver como los resultados de EDFA son, en general, mejores en la configuración 2 que en el 3. Esto es debido a que en la segunda configuración todas las tareas tienen *deadline*, y por tanto se pueden beneficiar de la ordenación EDF.

Se ha utilizado el simulador Player/Stage para analizar más detalladamente los entornos con un número relativamente bajo de robots. Con este simulador se puede estudiar la interferencia física entre robots. En estos experimentos se han utilizado varios robots Pioneer 3DX, ya que se trata de una de las plataformas más utilizadas para la investigación en robótica. Para ello se han utilizado las estrategias SBPA, SUA, EDFA y NFS. Las dimensiones del entorno fueron de 18mx18m y la velocidad máxima de los robots se limitó a 0,25m/s. Los robots utilizan esta información para calcular el tiempo estimado para recoger un objeto y, de esta manera, decidir si son capaces de finalizar la tarea antes de su *deadline*. En cada experimento se han usado un total de 200 objetos distribuidos aleatoriamente y con un *deadline* generado mediante una variable aleatoria uniforme entre 12 y 70 segundos. En este caso no se han utilizado llegadas de Poisson, sino que cuando un objeto es recogido inmediatamente aparece otro en una posición aleatoria. De esta manera, el número de objetos en el entorno siempre es igual a M. Se ha utilizado un número constante de tareas en lugar de llegadas aleatorias para tener un entorno similar al utilizado en [81], y así poder comparar los resultados. En los experimentos se analizará la influencia de M sobre el número de objetos recogidos antes de su *deadline*. La figura 3.14 muestra un ejemplo

de una simulación utilizando Player/Stage con 4 robots, las figuras cuadradas representan los objetos a recoger y la etiqueta que aparece junto a cada robot indica la información sobre el objeto que tiene asignado.

Figura 3.14: Ejemplo de simulación utilizando Player/Stage con 4 robots. Las figuras cuadradas representan los objetos a recoger.

La tabla 3.3 muestra el número de tareas que no cumplen su *deadline* usando el algoritmo SBPA, donde R es el número de robots, M el número de objetos simultáneos en el entorno y D el parámetro asociado al método. En los métodos basados en subastas el parámetro D tiene el mismo significado que en los algoritmos de *swarm*, explicados en la sección 3.2.1. Entre paréntesis aparece el incremento porcentual en el incumplimiento de *deadline* al utilizar la estrategia NFS. En todos los casos, SBPA es mejor que NFS, especialmente cuando se aumenta el número de robots. Por ejemplo, con 12 robots, 5 objetos y $D = 9$, el método SBPA mejora en un 60 % los resultados obtenidos con NFS. Es más, la distancia máxima D no tiene un gran impacto en los resultados, al contrario de lo que pasaba en las pruebas sin *deadlines* y con un simulador no realista realizadas en [81].

		R=4	R=8	R=12
D=9	M=5	67 (26 %)	38 (38 %)	18 (60 %)
	M=10	90 (17 %)	48 (46 %)	48 (44 %)
D=12	M=5	69 (39 %)	35 (53 %)	24 (63 %)
	M=10	90 (29 %)	56 (47 %)	56 (47 %)
D= ∞	M=5	78 (32 %)	45 (46 %)	24 (61 %)
	M=10	87 (33 %)	59 (46 %)	68 (23 %)

Tabla 3.3: Número de tareas que no cumplen su *deadline* utilizando el algoritmo SBPA. Entre paréntesis aparece el incremento porcentual de estas tareas cuando se utiliza la estrategia NFS. R es el número de robots, M el número de objetos simultáneos y D el parámetro del método.

Las tablas 3.4 y 3.5 muestran el número de tareas que no cumplen su *deadline* usando, respectivamente, los algoritmos SUA y EFDA, siempre usando estrategias sin planificación local. Tal como se puede observar, SBPA presenta mejores resultados que los métodos SUA y EDFA, es más, en la mayoría de casos NFS es mejor que estos últimos. Los resultados de los algoritmos SUA y EDFA mejoran a medida que la relación entre el número de robots y la distancia ($\frac{R}{D}$) disminuye. Cuando hay muchos robots respecto a la distancia, como por ejemplo cuando $D = 9$ y $R = 12$, los resultados de EDFA pueden llegar a ser un 73 % peores que con el método de NFS, en cambio cuando la distancia es máxima y con 4 robots los beneficios de EDFA llegan al 16 % respecto a una estrategia NFS. La estrategia EDFA también mejora los peores resultados del método SUA, por ejemplo cuando $D = 9$ y $R = 12$ se incrementa en un 118 % el número de incumplimientos de *deadline* utilizando SUA respecto a *swarm*, en cambio cuando se utiliza EDFA ese incremento se reduce hasta un 73 %.

Los resultados de las tablas 3.3, 3.4 y 3.5 son coincidentes con los ya obtenidos con el simulador RoboSim, donde con un número escaso de robots NFS mejoraba los resultados de SUA y EDFA. Se ha de tener en cuenta también que, en todos los experimentos realizados sobre RoboSim, el valor de D siempre era infinito, y por tanto el valor $\frac{R}{D}$ era 0. Además, como ya se ha comentado, Player/Stage permite tener en cuenta los efectos de la interferencia física entre robots, que al moverse en un entorno

tan reducido (18mx18m), comparado con los 160mx160m del entorno de RoboSim, hace que los posibles beneficios de EDFA o SUA respecto a métodos de *swarm* se vean reducidos. En los capítulos 4 y 5 se hará un análisis detallado de cómo incluye la interferencia sobre el comportamiento en sistemas multi-robot y cómo modelizarla para mejorar los resultados de los métodos basados en subastas.

		R=4	R=8	R=12
D=9	M=5	105 (-17 %)	94 (-54 %)	98 (-118 %)
	M=10	128 (-17 %)	93 (-4 %)	96 (-13 %)
D=12	M=5	102 (11 %)	85 (-13 %)	89 (-39 %)
	M=10	128 (-2 %)	94 (11 %)	89 (15 %)
D= ∞	M=5	98 (14 %)	84 (0 %)	90 (-48 %)
	M=10	122 (6 %)	100 (9 %)	88 (-13 %)

Tabla 3.4: Número de objetos que no cumplen su *deadline* con la estrategia SUA. Entre paréntesis aparece el incremento porcentual de estas tareas cuando se utiliza el método NFS. R es el número de robots, M el número de objetos simultáneos y D el parámetro del método.

		R=4	R=8	R=12
D=9	M=5	99 (-10 %)	87 (-43 %)	78 (-73 %)
	M=10	135 (-24 %)	87 (2 %)	90 (-6 %)
D=12	M=5	97 (15 %)	81 (-8 %)	91 (-42 %)
	M=10	126 (0 %)	97 (15 %)	100 (-5 %)
D= ∞	M=5	96 (16 %)	94 (-12 %)	85 (-39 %)
	M=10	132 (-2 %)	94 (8 %)	96 (-9 %)

Tabla 3.5: Número de objetos que no cumplen su *deadline* con la estrategia EDFA. Entre paréntesis aparece el incremento porcentual de estas tareas cuando se utiliza el método NFS. R es el número de robots, M el número de objetos simultáneos y D el parámetro del método.

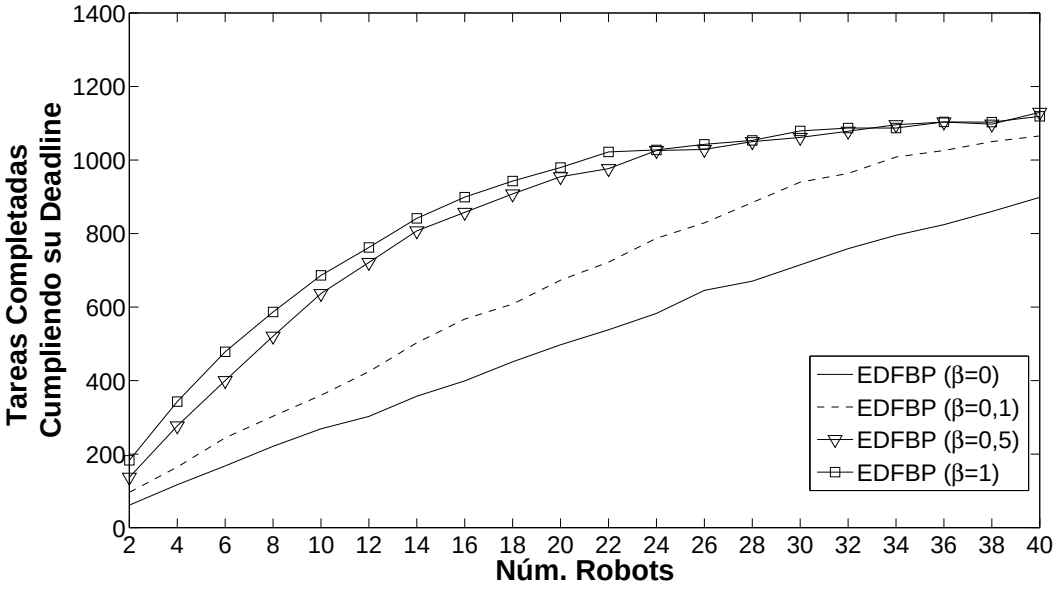
Resultados del algoritmo EDFBP

La figura 3.15(a) muestra algunos de los resultados de los experimentos realizados para validar el algoritmo EDFBP (algoritmo 6) en la segunda configuración (*deadlines* aleatorios), con $\lambda = 0,3$, sobre el simulador RoboSim y siempre utilizando la estrategia sin planificación local. Se ha de recodar que el algoritmo EDFBP con $\beta = 1$ es igual a SBPA y con $\beta = 0$ se comporta como el algoritmo SUA. Como se puede observar, sólo dividiendo el conjunto de tareas en 2 grupos (β = 0,5), los resultados del algoritmo EDFBP son muy similares a los obtenidos con SBPA, pero con un coste computacional muy inferior, tal como ya se explicó para la figura 3.2. La figura 3.15(b) muestra los resultados de los mismos experimentos pero en el configuración 1 donde todas las tareas tienen el mismo *deadline*. En este caso, los resultados en función de β son más similares entre ellos que en el caso anterior, mostrando SBPA y EDFBP para $\beta = 0,5$ resultados muy parecidos.

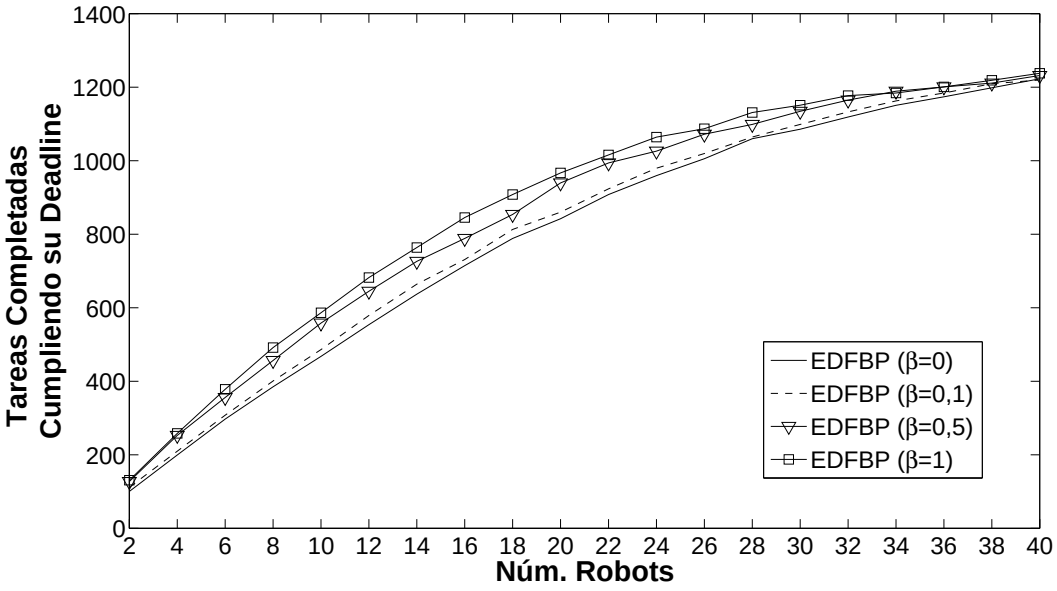
3.6. Conclusiones

En este capítulo se ha analizado el comportamiento de un gran número de estrategias de asignación de tareas en entornos con un único robot por tarea (SR), utilizando tanto algoritmos basados en *swarm* como en subastas. Se han analizado estos algoritmos sobre todo en entornos con tareas con restricciones temporales, donde cada tarea ha de ejecutarse antes de un tiempo límite. Para ello se ha ejecutado la tarea de recolección de objetos en la que cada objeto ha de ser recogido antes de un determinado instante de tiempo (*foraging* con *deadlines*). También se ha analizado para este tipo de entornos el impacto de las estrategias que permiten una planificación local de las tareas asignadas a cada robot con respecto a las estrategias en las que los robots ejecutan las tareas por orden de llegada.

Si se tienen en cuenta todas las posibles combinaciones (con planificación-sin planificación, *swarm*-subasta) se han analizado un total de 11 algoritmos diferentes. La tabla 3.6 muestra un resumen de todos los algoritmos clasificados en función del



(a) Configuración 2 (*deadlines* aleatorios).



(b) Configuración 1 (tareas uniformes).

Figura 3.15: Número de tareas finalizadas antes de su *deadline* utilizando el algoritmo EDFBP con $\lambda = 0,3$ y con diferentes tipos de configuraciones.

paradigma general utilizado (*swarm* o subasta) y de si los robots son capaces o no de planificar localmente (LP). Por lo que sabemos del estado del arte, es la primera vez que se realiza un estudio de este tipo donde se comparan algoritmos *swarm* con subastas teniendo en cuenta tareas con restricciones temporales. Así, se han ampliado trabajos como los realizados por Kalra y Martolini en [81], demostrando que algunas de sus conclusiones no son válidas para algunos algoritmos en entornos con restricciones temporales.

	Sin planificación local	Con planificación local
Subasta	SUA, EDFA, SBPA, EDFBP	SUA-LP, EDFA-LP, SBPA-LP
Swarm	RTH, NFS, PSW-D, PSW-R	-

Tabla 3.6: Resumen de los algoritmos implementados en este capítulo.

De los 11 algoritmos utilizados PSW-D, PSW-R, y EDFBP son métodos nuevos propuestos en esta tesis. Los resultados de los algoritmos PSW-D y PSW-R han demostrado que combinar estrategias deterministas y no deterministas en métodos *swarm* incrementa el número de tareas finalizadas respecto a un algoritmo clásico de *response-threshold* (algoritmo RTH), sin tener que aumentar substancialmente su complejidad algorítmica. También se ha demostrado que, en todos los casos estudiados, el método PSW-R presenta mejores resultados que el algoritmo PSW-D. En cambio, PSW-D no necesita ningún mecanismo de comunicación entre robots, mecanismos imprescindibles en el método PSW-R. La complejidad del mecanismo de comunicaciones de PSW-R es muy inferior a la de otros algoritmos de *swarm* que también requieren comunicación, como por ejemplo [45]. Otra ventaja del algoritmo PSW-R es que a pesar de tener un parámetro (A) su valor óptimo presenta una gran estabilidad en todos los escenarios analizados. Este hecho contrasta con el comportamiento del mejor valor del parámetro θ en el algoritmo RTH, que varía de manera notoria en función del entorno utilizado. La gran desventaja de PSW-D y PSW-R respecto a RTH es el incremento de la distancia total recorrida por los robots, especialmente cuando la tasa de llegada de tareas es baja. De todas maneras, en entornos

con una tasa de llegada de tareas alta, por ejemplo para $\lambda = 0,3$, la distancia total recorrida en los 3 algoritmos de *swarm* es muy parecida.

El algoritmo más complejo implementado en este estudio es el SBPA, pero también es el que mejores resultados genera, sobre todo en entornos con restricciones temporales. El siguiente algoritmo de subastas que produce mejores resultados es EDFA seguido del SUA. Cuando el número de robots es elevado, el número de tareas finalizadas al usar tanto *swarm* como subastas es bastante similar, en cambio en los algoritmos basados en subastas la distancia recorrida por los robots es muy inferior, especialmente para valores bajos de tasa de llegadas ($\lambda = 0,05$ y $\lambda = 0,1$). Por tanto, los algoritmos de subasta aúnan las ventajas de RTH en cuanto a distancia recorrida con el aumento de tareas finalizadas de los métodos PSW, sin necesidad de tener que ajustar ningún parámetro. El coste para conseguir esto es la necesidad de sistemas y protocolos de comunicaciones para implementar las subastas.

En todas las pruebas realizadas los robots únicamente podía realizar planificación local si se utilizaban métodos basados en subastas. Es decir, en estos métodos cada robot podía decidir en qué orden recoger los objetos que se le habían asignado y enviar las pujas dependiendo del resultado de la planificación. Lógicamente, la capacidad de cómputo de los robots en un entorno LP ha de ser mayor a la requerida en sistemas sin planificación local. Como contrapartida, los algoritmos SUA-LP y EDFA-LP han llegado a alcanzar resultados muy similares a los obtenidos con el algoritmo SBPA, sólo por el hecho de utilizar estrategias de tipo LP. Por tanto, se puede concluir que aumentar la complejidad de cada robot individualmente con métodos LP o incrementar la complejidad del subastador central utilizando SBPA es equivalente, en términos de número total de tareas finalizadas.

Se ha propuesto un nuevo algoritmo de subastas EDFBP híbrido entre EDFA y SBPA. EDFBP permite ajustar mediante el parámetro β la complejidad del algoritmo a la capacidad de cómputo del subastador central. Las pruebas realizadas demuestran que EDFBP presenta resultados similares a los del algoritmo SBPA con una complejidad algorítmica aproximadamente un 50 % inferior. Se trata, por tanto, del primer

algoritmo basado en subastas capaz de ajustar la complejidad algorítmica del proceso en tareas con restricciones temporales.

Todos los algoritmos mostrados en esta capítulo sólo permiten asignar un único robot a cada tarea, es decir siempre se han abordado problemas de tipo SR. En cambio, existen muchos tipos de tareas en los que es imprescindible que múltiples robots sean asignados a una misma tarea (MR), formando coaliciones de robots para llevar a cabo misiones específicas. Los problemas MR requieren algoritmos más complejos que los utilizados en sistemas con un único robot por tarea, sobre todo en entornos con restricciones temporales. En los siguientes capítulos se estudiarán los problemas MR, se verá cómo tenerlos en cuenta en métodos basados en subastas y, además, se analizará las modificaciones necesarias para poderlos utilizar en entornos con restricciones temporales, similares a los ya estudiados en este capítulo.

Capítulo 4

Formación de coaliciones en entornos sin restricciones temporales

En este capítulo se trata detalladamente la problemática existente en sistemas de subastas para la asignación de múltiples robots a una misma tarea sin restricciones temporales. Se trata pues, según la taxonomía de Gerkey, de un problema de tipo ST-MR-IA. La solución propuesta consiste en un nuevo mecanismo que permite tener en cuenta la interferencia física entre robots para decidir cuántos de ellos son necesarios para finalizar con éxito cada tarea. Además, se ha propuesto un nuevo método de coordinación del movimiento llamado *lane keeping* para disminuir la interferencia física antes mencionada en tareas de recogida de objetos.

El mecanismo aquí propuesto ha sido uno de los primeros métodos de subasta en permitir múltiples robots en cada tarea. Además, se trata del único método de subastas que tiene en cuenta la interferencia física entre robots en el proceso de negociación. De hecho, el nivel de interferencia física, será el factor clave a la hora de decidir cuántos robots son necesarios para realizar cada tarea. Como se demostrará con los resultados experimentales, incluir o no el nivel de interferencia afecta de manera importante al rendimiento del sistema.

4.1. Entorno utilizado: recolección de objetos con múltiples robots por tarea

Para validar tanto los métodos de subastas como los de coordinación del movimiento, se ha utilizado una tarea de recogida y transporte de objetos, similar a la ya explicada en el capítulo anterior, pero sin restricciones temporales y permitiendo que múltiples robots puedan colaborar para recoger un mismo objeto. Es decir, se trata de un problema de tipo ST-MR-IA. Concretamente, esta nueva misión tendrá las siguientes características. Los robot han de localizar una serie de objetos distribuidos aleatoriamente en un entorno desconocido. La posición inicial de los robots en el entorno también es aleatoria. A diferencia de lo que sucedía en la tarea presentada en el capítulo 3, cuando un robot recoge un objeto, lo transporta hasta un determinado punto de descarga común a todos los robots y objetos. Cada objeto tiene asociado un peso y cada robot una determinada capacidad de carga. Esta capacidad indica la cantidad de peso que el robot puede transportar en un único viaje. Si un robot no puede cargar todo el objeto de una vez, carga una parte de su peso, lo transporta hasta el punto de descarga y regresa de nuevo al objeto para cargar más. De esta manera, los robots pueden transportar los objetos en pequeños trozos, pudiendo colaborar varios de ellos en la recogida de un mismo objeto. Esta característica es la que diferencia esta tarea del *foraging* clásico, donde normalmente cada objeto ha de ser transportado de una sola vez por un único robot. El tiempo de carga del peso en el robot será proporcional a la capacidad de carga, así, un robot para cargar n unidades de peso también necesitará estar n unidades de tiempo cerca del objeto a recoger, antes de iniciar su viaje hacia el punto de descarga. Cada objeto puede tener asociada una prioridad, representada por un valor entero entre 1 y 5, siendo 1 la prioridad más baja y 5 la más alta. La prioridad de la tarea representa la urgencia de su ejecución, es decir, tareas con alta prioridad deberán ser ejecutadas antes que las de prioridad menor. Esta tarea muestra ciertas similitudes con la propuesta por Chaimowicz en [25], pero a diferencia de ésta, nuestro sistema permite no saber a priori el número de robots

necesarios para ejecutar la tarea. La tarea de tratamiento de emergencias planteada por Østergaard et al. en [102] también es similar a la descrita aquí, si bien Østergaard et al. únicamente utilizan robots homogéneos en entornos ST-SR-IA, mientras que el método que se presenta aquí permite trabajar con un grupo heterogéneo de robots.

Para este tipo de problemas, la asignación de un conjunto de tareas $T = \{t_1, ..., t_m\}$ a un conjunto de robots $R = \{r_1, ..., r_n\}$ se puede representar mediante un conjunto $TA = \{C^0, ..., C^{m'}\}$, donde C^j=$\{R^j, t_j\}$ y $R^j \subseteq R$ es el conjunto de robots asignados tarea t_j. R^0 representa el conjunto de robots sin tarea asignada. Nótese que $m' \leq m$ ya que las tareas sin robots asignados no aparecen en TA. Cada conjunto de robots R^j asignados a una tarea se denomina coalición o grupo de trabajo. Para que una asignación TA sea válida, el conjunto de todas las coaliciones de robots ha ser una partición de R. El objetivo del algoritmo MRTA es encontrar una asignación óptima TA^* de manera que maximice una determinada función de utilidad $U(TA)$, que depende de la asignación TA. Así pues, se trata de un problema de particiones de conjuntos, explicado en la sección 2.1.2 y, por tanto, es de tipo NP-hard.

Se define $workCapacity_{i,j} \in \Re$ como la capacidad de trabajo que tiene el robot r_i para llevar a cabo la tarea t_j. El valor de $workCapacity_{i,j}$ dependerá de las características de la tarea: peso y posición, así como de las características del robot: velocidad máxima, posición actual, capacidad de carga, etc. Además, para cada tarea t_j tiene asociada una prioridad $Pri_j \in$ [1..,5], que indica lo urgente que es su ejecución. La función U guarda relación directa con la tarea u objetivos asignados a los robots. En el caso que nos ocupa, se ha elegido como valor de U el peso total transportado por el conjunto de coaliciones, de manera que un conjunto de coaliciones será más útil cuanto más peso sea capaz de transportar.

La tarea aquí definida puede aplicarse a un gran número de situaciones reales. Por ejemplo, tareas de limpieza de superficies o rescate en catástrofes. En el caso de las tareas de recogida de residuos, éstos muchas veces aparecen agrupados por zonas y además se pueden transportar en pequeños trozos. No obstante, tanto el sistema de

coordinación del movimiento, como el mecanismo de subastas son genéricos, pudiéndose extender su funcionalidad a una amplia gama de tareas, sobre todo a aquellas donde la utilidad U dependa del número de robots que formen cada uno de los grupos de trabajo.

4.2. Interferencia física entre robots

En un sistema formado por múltiples robots existen algunos recursos compartidos que sólo pueden ser utilizados por un número limitado de robot de manera simultánea. Uno de estos recursos es el espacio físico por el que se moverán los robots. Este espacio es compartido por todos los robots y, evidentemente, cuando uno de ellos ocupa una determinada posición, ésta no puede ser ocupada por ningún otro robot. La interferencia entre robots se produce cuando dos o más de ellos pretenden acceder a una misma posición del espacio al mismo tiempo, situación que puede incrementar de manera ostensible el tiempo de ejecución de la tarea. Si el grado de interferencia es muy elevado, los robots pueden pasar más tiempo intentando evitarse entre sí que ejecutando la misión que tienen encargada. Además, la manera en que la interferencia influye sobre el comportamiento del sistema varía a lo largo del tiempo dependiendo de las características de los robots y del entorno. Toda esta variabilidad dificulta enormemente la modelización correcta y precisa del proceso de interferencia física. Por tanto, la interferencia física hace que, en general, la utilidad U de una asignación válida TA no pueda ser calculada como un simple sumatorio de las utilidades individuales de cada robot. Esto es, en general se cumple la siguiente condición: $U(TA) \leq \sum_{C^j \in TA} \sum_{r_i \in R^j} workCapacity_{i,j}$.

A continuación se muestra un ejemplo de cómo la interferencia física puede afectar al tiempo de ejecución de la tarea de recogida de objetos presentada en la sección 4.1. Para ello, se han realizado una serie de simulaciones de la tarea de recogida en la que los robots han de transportar hasta un punto de descarga un único objeto, situado en una posición conocida previamente y con un peso infinito. No existe en el entorno

ningún otro elemento, a parte de los robots y el objeto a recoger. Además, la capacidad de carga de los robots es igual a 2 unidades de peso. La figura 4.1 muestra a modo de ejemplo la situación inicial en las pruebas efectuadas con 9 robots, donde el cuadrado representa el objeto a recoger, los pequeños círculos son los robots y el círculo de gran tamaño es la zona de descarga. Para realizar los experimentos se ha utilizado el simulador *Robot Colonies Tool* (RoboCoT), creado por los autores como trabajo previo a esta tesis [62, 60, 61]. RoboCoT es un simulador realista que, al contrario de RoboSim, permite tener en cuenta los efectos de la interferencia física entre robots. Además, al ser una herramienta creada específicamente para este trabajo, reduce el tiempo de ejecución respecto a otros simuladores genéricos como Player/Stage. En el anexo B se puede encontrar una descripción detallada de este simulador.

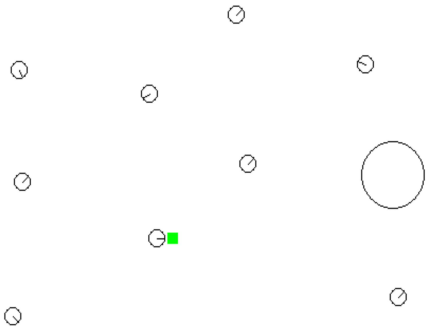


Figura 4.1: Ejemplo de disposición inicial de los robots en los experimentos para modelizar la interferencia.

La figura 4.2 muestra la cantidad de peso transportado en función del número de robots después de 35.000 unidades de tiempo. Para obtener estos resultados se han utilizado 3 tipos de condiciones iniciales diferentes. En todas ellas se mantuvo la distancia entre objeto y punto de descarga y únicamente se modificó la posición inicial de los robots en el entorno. Como se puede observar, el peso transportado no crece linealmente con el número de robots. De esta manera, los beneficios de incluir un nuevo robot son menores a medida que su número se incrementa. Este hecho se debe a que cuanto mayor sea el número de robots asignados a una tarea, mayor será el grado de interferencia. La tabla 4.1 muestra la desviación estándar (σ) del peso

transportado en las distintas pruebas realizadas. Se pone de manifiesto que existe muy poca variabilidad, lo que significa que en esta experiencia el peso transportado es prácticamente independiente de la disposición inicial de los robots.

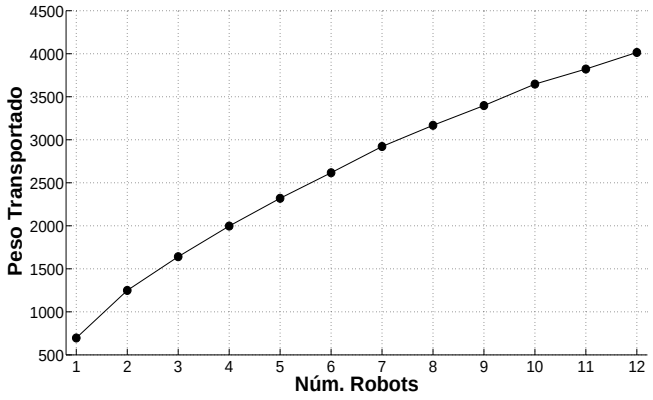


Figura 4.2: Peso transportado por todos los robots en la tarea de recogida de objetos.

	Número de robots											
	1	2	3	4	5	6	7	8	9	10	11	12
σ	0	17,5	5,0	11,4	20,0	20,5	20,1	17,1	41,5	17,5	26,9	25,3

Tabla 4.1: Desviación estándar (σ) del peso transportado por los robots en los mismos experimentos que en la figura 4.2.

Una conclusión del análisis de los resultados mostrados en la figura 4.2 es que, debido a la interferencia física, el número de robots asignados a cada tarea influye notablemente en el resultado final. Por ejemplo, se podría repetir el experimento anterior pero con 12 objetos y con 12 robots, de manera que los robots asignados a un objeto no influyan sobre los robots asignados a otra tarea. Este tipo de tareas corresponden a las llamadas tareas espacialmente clasificables presentadas por Dhal [30], que sigue siendo un problema NP-hard. En este caso, si se asigna un único robot a cada uno de los 12 objetos, se podrán transportar un total de $12 \cdot 696$ =8.352 unidades de peso, donde 696 es el peso transportado por un único robot en los experimentos mostrados en la figura 4.2. En cambio, si se decide que los 12 robots sean asignados a una misma tarea, la interferencia física hará que únicamente se transporten 4.014,7 unidades de

peso, provocando una disminución del 108 % en el peso total transportado. Por tanto, un buen mecanismo de asignación debe, no sólo seleccionar a los mejores robots para cada tarea, sino también decidir cuántos son necesarios para ejecutarla.

En la literatura especializada, la interferencia física ha sido tratada por varios autores [89, 109, 73, 126]. En todos estos casos se trata de entornos en los que sólo se puede asignar un robot por tarea o en los que se utilizan algoritmos de tipo *swarm* para solucionar el problema. Además, en la mayoría de casos se utiliza una tarea de recolección de objetos para verificar los métodos. Cabe destacar que en [109] Rosenfeld et al. demostraron que el nivel de interferencia era especialmente destacable cuando los robots debían llevar el objeto recolectado a un punto de descarga. En cambio, cuando los robots sólo tenían que recoger el objeto, la interferencia tenía efectos mucho menores, este último es el caso de la tarea ya descrita y analizada en el capítulo 3.

En los últimos años también se han publicado diversos trabajos con el objetivo de modelizar matemáticamente el comportamiento de los sistemas multi-robot y, de esta manera, poder predecir los efectos de la interferencia sobre el sistema. Existen dos tipos básicos de modelos: los modelos macroscópicos y los microscópicos. En los modelos microscópicos se analiza el sistema teniendo en cuenta el comportamiento individual de cada uno de sus elementos (robots). En cambio, los modelos macroscópicos tratan al conjunto de robots como un todo, sin entrar en los detalles concretos de los miembros que forman el grupo, siendo, por tanto, estos modelos más simples que los microscópicos. Ijspeert et al. presentaron en [74] un modelo microscópico para tareas de recogida de objetos con un único robot por tarea. Los experimentos demostraron que los resultados obtenidos con el modelo y los obtenidos sobre robots reales eran muy similares. Lerman y Galstyan propusieron en [92, 91] un modelo macroscópico para el análisis de la interferencia física en tareas de recogida de objetos pudiéndose asignar un único robot por objeto. La arquitectura de control de los robots estaba basada en comportamientos y la decisión sobre qué tarea se debía ejecutar se tomaba únicamente a partir de la información local que tenía cada robot. Bajo estas premisas, el sistema se podía tratar como un sistema de Markov, en el que su estado futuro

únicamente depende del actual. Los resultados fueron verificados en sistemas reales, pero el modelo era totalmente dependiente del tipo de arquitectura utilizada para controlar los robots.

En el capítulo 5 se explicará un mecanismo de modelización de la interferencia física que permitirá calcular los tiempos de ejecución previstos de las tareas en entornos con restricciones temporales. Este modelo es independiente de la arquitectura de los robots y puede irse adaptando a las condiciones del entorno a medida que se ejecuta la tarea. Debido a que el presente capítulo se centra únicamente en la utilización de mecanismos de subastas para la ejecución de tareas sin *deadline*, la interferencia ha sido tratada de manera cualitativa. En cambio, en el capítulo 5 se presenta un modelo cuantitativo de la interferencia que permite estimar el tiempo de ejecución de tareas con *deadlines*.

4.3. Mecanismo de subastas propuesto para entornos sin restricciones temporales

En esta sección se explica un nuevo mecanismo de asignación de tareas desarrollado para tratar la interferencia física entre robots en entornos sin restricciones temporales. Nuestro método permite asignar múltiples robots a una misma tarea, mientras que cada robot sólo puede tener una tarea asignada y no se hace ningún tipo de razonamiento sobre los efectos futuros de las decisiones tomadas; se trata por tanto de un sistema ST-MR-IA. Este sistema está inspirado tanto en mecanismos de *swarm* [41, 72, 94, 1], como en algoritmos basados en subastas [36, 53]. Al contrario de lo que sucede con los mecanismos de subastas actuales, el sistema que se presenta aquí aborda explícitamente los problemas relacionados con la interferencia física sin asumir previamente cuantos robots serán necesarios para llevar a cabo una tarea. Este mecanismos será ampliado en el capítulo 5 para tener en cuenta tareas con restricciones temporales.

Existen diversos mecanismos de subastas que permiten crear coaliciones de robots

asignados a una misma tarea [115, 114, 139, 111, 127, 122], sin embargo, el trabajo mostrado en este capítulo es el único que además no requiere un número prefijado de robots por tarea y tiene en cuenta la interferencia física [66, 65, 63, 64]. Otros autores, como Paquet [103], han propuesto también mecanismos para dimensionar el tamaño de los grupos, pero estos no pueden ser aplicados *on-line* debido a su elevado tiempo de ejecución, ni tampoco abordan la interferencia física al estar pensados para entornos multi-agentes, como la RoboCup Rescue, y no para sistemas multi-robot. Así pues, el sistema aquí propuesto aporta la originalidad de ser el único basado en subastas que permite formar coaliciones de un número no prefijado de robots e incluir los efectos de la interferencia física.

4.3.1. Grupos de trabajo y selección de líder

Un concepto básico en el presente algoritmo de subastas es el de *grupo de trabajo*. Un grupo de trabajo se define como un conjunto de robots que cooperan en la ejecución de una determinada tarea, es decir los conceptos de coalición y grupo de trabajo son análogos. Durante la tarea de recogida de objetos, el grupo de trabajo estará formado por todos los robots que estén transportando un mismo objeto. En todos los grupos de trabajo existirá un robot que actuará como líder de la tarea y será el responsable de decidir qué robots y, sobre todo, cuántos formarán parte del grupo de trabajo. Para tomar esta decisión el líder utilizará mecanismos de negociación basados en subastas. Además, el líder se encargará de calcular la cantidad de trabajo a realizar y de monitorizar la ejecución de la tarea.

Inicialmente los robots no forman parte de ningún grupo de trabajo ya que no tienen ninguna tarea asignada. Cuando un robot libre encuentra una tarea, intenta convertirse en su líder enviando un mensaje al resto de robots indicando que ha encontrado una nueva tarea. Los robots pueden detectar una nueva tarea mediante sus propios sensores o bien pueden recibir la información de otro robot o de un sistema central. Los robots que no son líderes y que reciben este mensaje o que también han detectado la tarea, intentan convertirse en líderes. Si existen dos o más robots

que pretenden ser los líderes de una misma tarea, se otorgara el liderazgo al 'mejor' de ellos siguiendo el siguiente proceso: el robot que pretende convertirse en líder, que llamaremos r_1, envía un mensaje *broadcast* a todos los demás informando de su intención. Si otro robot que ya es líder de la misma tarea recibe el mensaje, envía un mensaje a r_1 indicando que la tarea ya ha sido asignada. Cuando r_1 recibe este mensaje desiste de su intención de convertirse en líder. Si el que recibe el mensaje de r_1 es otro robot, r_2, entonces puede ocurrir que r_2 sea mejor que r_1 o viceversa. Si r_2 es mejor que r_1, entonces r_2 envía a r_1 un mensaje indicando que ya hay un robot mejor solicitando la tarea. Cuando r_1 recibe este mensaje, desiste de su intento de obtener el liderazgo. En el caso de que r_1 sea mejor que r_2, este último deja de solicitar el liderazgo sin que se tenga que enviar ningún mensaje. Si transcurrido un determinado período de tiempo el robot que solicita ser líder no recibe ningún mensaje en contra, se convierte en el líder de la tarea. El concepto de 'mejor' dependerá del tipo de tarea que se realice, en el caso de los experimentos mostrados en la sección 4.5 el mejor robot será aquel que tenga la mayor capacidad de carga para la tarea a realizar. El algoritmo 9 muestra los pasos que el robot r_1 sigue durante este proceso, donde TIME_LEADERSHIP representa el tiempo máximo de espera antes de que r_1 se convierta en líder y *tiempo* el instante de tiempo actual. En la sección 4.3.6 se explicarán las ampliaciones que se han realizado sobre este algoritmo para tratar los posibles errores en la comunicación entre robots.

Una vez que un robot se ha convertido en el líder de la tarea, evalúa el trabajo necesario para llevarla a cabo y procede a formar el grupo de trabajo. La evaluación del trabajo necesario dependerá del tipo de tarea, que en el caso de la recogida de objetos será el peso total a transportar. Para formar el grupo de trabajo el líder utiliza dos procesos diferentes: la negociación 'líder a robot' y la negociación 'líder a líder'. Estas dos negociaciones se diferencian en que en la primera únicamente pueden entrar a formar parte del grupo de trabajo robots sin ninguna tarea asignada. En cambio, en la negociación 'líder a líder' se permite el intercambio de robots entre grupos de trabajo. Inicialmente el líder utiliza la negociación 'líder a robot'. Si después de este

Algoritmo 9 Algoritmo de solicitud de liderazgo de la tarea t por el robot r_1

```
1: Enviar solicitud tarea t
2: tiempo_inicio ← tiempo
3: mientras (tiempo − tiempo_inicio) ≤ TIME_LEADERSHIP hacer
4:    si recepción mensaje de r_2 entonces
5:       si r_2 ya es líder de t entonces
6:          Abandonar
7:       fin si
8:       si r_2 solicita t y r_2 mejor que r_1 entonces
9:          Abandonar
10:         devolver
11:      fin si
12:      si r_2 solicita t y r_1 mejor que r_2 entonces
13:         Continuar
14:      fin si
15:   fin si
16: fin mientras
17: r_1 nuevo líder de t
```

proceso de negociación no se ha podido conseguir un grupo de trabajo 'óptimo', se inicia la negociación 'líder a líder'. Más adelante en este mismo capítulo se expondrá cuando se considera que un grupo es óptimo.

4.3.2. Negociación 'líder a robot': estrategia no preemptiva

Después de que un robot haya conseguido el liderazgo de una tarea, intenta formar un grupo de trabajo mediante la negociación 'líder a robot'. En este proceso sólo se seleccionarán los mejores robots sin ninguna tarea asignada, no permitiendo el intercambio de robots entre grupos de trabajo. Siguiendo la terminología utilizada en el área de sistemas operativos, se trata de una estrategia no preemptiva. Para seleccionar los robots que formarán parte del grupo de trabajo se utiliza un proceso de subasta, similar al utilizado en [53] por Gerkey y Mataric. Concretamente, el algoritmo que implementa el proceso de subasta consta de los siguientes pasos:

1. **Inicio de la subasta**: el líder envía un mensaje al resto de robots para informarles de que necesita formar un grupo de trabajo para ejecutar una nueva

tarea.

2. **Envío de las pujas (*bids*)**: los robots sin ninguna tarea asignada, envían al líder sus pujas para la tarea. El valor de la puja dependerá de las características concretas de la tarea a realizar y del propio robot. En nuestro caso, la puja será la capacidad de carga del robot.

3. **Cierre de la subasta**: tras haber trascurrido un determinado período de tiempo, el líder cierra la subasta y no acepta más pujas de los robots.

4. **Selección de los robots**: el líder selecciona los mejores robots, es decir aquellos que tengan una mayor capacidad de carga. Los detalles de este proceso de selección y de la condición utilizada para limitar el tamaño del grupo de trabajo se explicarán más adelante.

5. **Aceptación/Rechazo**: el líder comunica a los robots seleccionados que ya pueden entrar a formar parte del grupo de trabajo y comenzar a colaborar en la ejecución de la tarea. Para ello envía un mensaje de ($AWARD$) a cada robot seleccionado. A su vez, en este punto se ha previsto la posibilidad de que cualquier robot seleccionado pueda enviar al líder un mensaje rechazando la entrada en el grupo. Este rechazo se producirá si durante el proceso de subasta el robot ya ha sido seleccionado por otro grupo de trabajo o si se ha convertido en líder de otra tarea. Cuando el líder recibe un mensaje de rechazo puede decidir iniciar un nuevo proceso de subasta para reclutar más robots.

Como se puede entender, habrá múltiples subastas simultáneas, de manera que un robot libre siempre pujará por todas las peticiones de tareas y decidirá entrar en el grupo de trabajo del primer líder del que reciba un mensaje de confirmación, incluso a pesar de estar a la espera de obtener un liderato. Este proceso de subasta permite seleccionar los mejores robots, pero no decidir el número necesario de ellos para ejecutar la tarea. En los algoritmos clásicos de subastas en la etapa llamada 'Selección de los robots' no existe ningún mecanismo para decidir cuándo se ha de

dejar de seleccionar nuevos robots. Para solucionar este problema, se ha modificado el algoritmo para que el líder seleccione los mejores robots hasta que la proporción entre la cantidad de trabajo a realizar y la capacidad de trabajo de todos los robots del grupo supere un determinado umbral (TH), esto es hasta que la siguiente condición se verifique:

$$TH_g = \frac{Pri_j \cdot taskWorkLoad_j}{\sum_{1 \leq i \leq n_g} workCapacity_{i,j}} < TH \tag{4.1}$$

donde n_g es el número de robots que forman parte del grupo de trabajo. $workCapacity_{i,j}$ es la capacidad de trabajo individual del robot r_i para la tarea t_j a realizar. TH es el umbral del grupo; que servirá para limitar el tamaño de la coalición de robots. $taskWorkLoad_j$ es la carga de trabajo, es decir la cantidad de trabajo necesario para finalizar la tarea t_j. Finalmente, Pri_j es la prioridad de la tarea t_j, tal como se ha definido en el apartado 4.1. Como se puede observar, la condición 4.1 limita el número de robots que pueden formar parte de un grupo de trabajo. Así, a una tarea con alta prioridad, o que requiera mucha capacidad de trabajo para finalizar, se le asignarán más robots que a otra con menor prioridad o con menor carga de trabajo. Los valores de TH_g pueden ir variando a lo largo de la ejecución de la tarea debido a que, por ejemplo, algunos robots abandonen el grupo o porqué el líder reciba un mensaje de rechazo de alguno de los robots seleccionados. El concepto aquí utilizado para el umbral del grupo (TH) es similar al de *response threshold* de algunos algoritmos basados en *swarm*, pero ahora aplicado a un proceso de subasta. Otros autores han utilizado conceptos heredados de *swarm* en métodos de subastas como por ejemplo [32, 31], pero sin utilizar la idea de umbral y sin tener en cuenta la interferencia física.

El algoritmo 10 muestra este proceso de subasta, donde $TIME\_AUCTION$ es el tiempo que el líder espera para recibir las pujas de los otros robots. La complejidad algorítmica de este proceso se puede calcular de la siguiente manera: sea n el número de pujas realizadas, la complejidad del algoritmo de ordenación de pujas (línea 9), utilizando *marge sort* o *binary tree sort*, es $O(n \cdot log(n))$. El bucle para seleccionar los

robots (líneas 12-16) tiene una complejidad $O(n)$, por tanto la complejidad algorítmica de todo el proceso es $O(n \cdot log(n) + n) \subset O(n \cdot log(n))$. El proceso que ha de llevar a cabo cada robot para enviar una puja es mucho más simple y tiene una complejidad algorítmica de $O(1)$. En cuanto a la complejidad de las comunicaciones, el líder ha de enviar un mensaje solicitando pujas (línea 1) más, como máximo, otros n para cada puja recibida y aceptada. Por tanto el número de mensajes a enviar por el líder será $O(n)$.

Algoritmo 10 Algoritmo de subasta tarea t con negociación 'líder a robot'

1: Enviar solicitud de pujas para la tarea t
2: $B_r \leftarrow \emptyset$
3: $tiempo_{inicio} \leftarrow tiempo$
4: **mientras** $(tiempo - tiempo_{inicio}) \leq TIME\_AUCTION$ **hacer**
5: **si** puja b_i recibida del robot r_i **entonces**
6: $B_r \leftarrow B_r \bigcup b_i$
7: **fin si**
8: **fin mientras**
9: Ordenar B_r de mayor a menor capacidad de trabajo
10: Inicializar TH_g
11: $j \leftarrow 0$
12: **mientras** $j \leq |B_r|$ **y** $TH_g \geq TH$ **hacer**
13: Seleccionar robot puja $B_r[j]$. (Mensaje $AWARD$)
14: Actualizar TH_g
15: $j \leftarrow j + 1$
16: **fin mientras**

4.3.3. Negociación 'líder a líder': estrategia preemptiva

Si mediante el proceso de negociación 'líder a robot' no se ha podido alcanzar un tamaño 'adecuado' del grupo de trabajo, esto es, que se verifique la condición 4.1, el líder inicia la negociación 'líder a líder'. En este proceso el líder intentará reclutar a robots que ya tengan una tarea asignada y que, por tanto, ya formen parte de otro grupo de trabajo. Debido a ello, a este sistema también se le ha llamado estrategia preemptiva. Para reclutar robots de otros grupos se inicia una nueva negociación

basada en una subasta en la que los diferentes líderes negociarán entre ellos, de manera similar a como antes lo habían hecho los robots de manera individual. Los robots sin tarea asignada también participarán en esta negociación actuando, a efectos prácticos, como si fuesen líderes.

En la negociación 'líder a líder' se utiliza un nuevo concepto llamado 'energía del grupo de trabajo' (E_g) para decidir cuando los robots pueden abandonar su grupo de trabajo para integrarse en otro. E_g es una medida que indica la tendencia que tiene el grupo a desprenderse de sus robots. Un grupo con un valor de energía alto es un buen candidato a ceder algunos de sus robots a otros grupos. En cambio, un grupo con un valor E_g bajo será un potencial receptor de nuevos robots. El valor de la energía se calcula utilizando la siguiente expresión:

$$E_g = \frac{n_g}{TH_g} \tag{4.2}$$

donde n_g es el número de robots del grupo de trabajo y TH_g es el valor calculado en la condición 4.1. Un valor elevado de TH_g indica que la capacidad de trabajo de los robots del grupo es pequeña en relación con la cantidad de trabajo a realizar. En este caso el grupo necesita más robots y, por tanto, su energía es baja. Además, el líder ha de intentar crear un grupo de trabajo con el menor número de robots posible, para reducir al máximo la interferencia física. Así, un grupo con muchos miembros necesita reducir su número de robots, lo que se traduce en un valor elevado de energía. Como se puede observar, el concepto de energía es similar al de estímulo utilizado en algunos algoritmos basados en *swarm*.

El proceso de subasta que se ha de seguir en la negociación 'líder a líder' consta de los siguientes pasos:

1. **Inicio de la subasta**: el líder envía un mensaje *broadcast* para indicar a los otros robots que se inicia una negociación 'líder a líder'.

2. **Envío de pujas**: los otros líderes envían el valor de la energía y las características de los robots de sus respectivos grupos. La información de los robots

consistirá en el identificador y la capacidad de trabajo de cada uno de ellos. Únicamente los líderes que ejecuten tareas de menor o igual prioridad a la solicitada podrán enviar esta información, por tanto, el intercambio de robots sólo se podrá dar de tareas de menor prioridad a tareas con igual o mayor prioridad. Para evitar cambios constantes de grupo, un robot no podrá abandonar su grupo de trabajo hasta que haya pasado cierto tiempo desde el momento de su incorporación. Cuando un robot sin ninguna tarea asignada recibe una petición de negociación 'líder a líder', también envía su capacidad de trabajo, de igual forma a como ya sucedía en la negociación 'líder a robot'.

3. **Cierre de la subasta**: después de un determinado período de tiempo, el líder que ha lanzado la subasta la cierra y no acepta más pujas.

4. **Selección de los mejores robots sin ninguna tarea asignada**: en primer lugar el líder selecciona aquellos robots con una mayor capacidad de trabajo y que no tengan ninguna tarea asignada. Así, en la negociación 'líder a líder' los robots libres se tratan como si fuesen líderes de otras tareas.

5. **Selección de los robots con una tarea ya asignada**: si tras el paso anterior aún no se cumple la condición 4.1, el líder empieza a seleccionar los robots con mayor capacidad de trabajo ya asignados a otros grupos. Para seleccionar estos robots, el líder utiliza como puja el valor B de la ecuación 4.3. Además se utiliza la ecuación 4.4 para evitar cambios de grupo demasiado frecuentes. El significado y uso de estas dos fórmulas se explicará a continuación.

6. **Aceptación/Rechazo**: de igual manera que en la negociación 'líder a robot', el líder notifica a los robots seleccionados que pueden empezar a trabajar para el grupo y estos pueden aceptar o rechazar la oferta.

Para la puja de los robots que ya forman parte de otro grupo de trabajo no se usa únicamente el valor de su capacidad de trabajo, como sucedía en la negociación

'líder a robot', sino que también entra en juego la energía del grupo. De esta manera, la puja (B) tendrá el siguiente valor:

$$B = workCapacity_{i,j} \cdot E'_{g2} \tag{4.3}$$

donde $workCapacity_{i,j}$ es la capacidad de trabajo del robot r_i en la tarea t_j y E'_{g2} es la energía que tendría el grupo en el que actualmente está el robot después de que éste lo abandonara. De esta manera, se tienen en cuenta tanto las características individuales de cada robot como las características del grupo de trabajo en el que se encuentra.

Uno de los objetivos del método de asignación de tareas es crear un sistema lo más estable posible, esto es, se quiere evitar que los robots estén constantemente pasando de un grupo a otro. Por esta razón, el cambio de grupo únicamente se producirá si la mejora que conlleva compensa sus costes, que en este caso será el tiempo necesario para pasar de un grupo a otro. Cuanto menor sea la energía de los grupos de trabajo más estable será todo el sistema, por tanto, el líder únicamente seleccionará un robot de otro grupo de trabajo si esta acción reduce significativamente el valor máximo de la energía del sistema; esto es, si se verifica la siguiente condición:

$$MAX(E_{g1}, E_{g2}) > O_v \cdot MAX(E'_{g1}, E'_{g2}) \tag{4.4}$$

donde $MAX(a, b)$ es una función que retorna el valor máximo de a y b. E_{g1} es la energía del grupo que solicita al robot y E_{g2} es la energía del grupo que lo cede. E'_{g1} y E'_{g2} son las energías que tendría cada grupo de trabajo si el paso del robot de un grupo a otro finalmente se produjese. Finalmente, O_v es la sobrecarga (*overhead*) porcentual consecuencia del intercambio y que se podría interpretar como el coste que supondría el cambio de grupo para el robot. El valor de O_v dependerá del tipo de entorno por el que se muevan los robots. En un entorno sin muchos obstáculos ir de una tarea a otra puede resultar relativamente sencillo ya que bastaría que el robot fuese en línea recta, lo que supondría que O_v tuviese un valor relativamente bajo. En cambio, en entornos más complejos, con paredes y muchos obstáculos el paso de una

tarea a otra podría resultar mucho más costoso, hecho que provocaría valores de O_v elevados. En los experimentos realizados para validar el sistema, se ha utilizado un valor de sobrecarga del 50 %, esto es un valor de O_v igual a 1,5.

El algoritmo 11 muestra el proceso que lleva a cabo el líder en una negociación 'líder a líder'. Hasta la línea 20 el proceso es muy similar al ya explicado en la negociación 'líder a robot', por tanto su complejidad algorítmica es $O(n \cdot log(n))$, donde n es el número de pujas. Cada vez que se selecciona un robot perteneciente a otro grupo de trabajo, se ha de calcular la nueva energía del grupo excluyendo el robot seleccionado (grupo k), lo que supone que también se han de modificar todas las pujas de los robots del grupo k (ver líneas 24-26). Este proceso se ha de repetir cada vez que se selecciona un robot, lo que supone una complejidad $O(n^2)$. De esta manera, la complejidad algorítmica del proceso de negociación 'líder a líder' que se ejecuta en el subastador es de $O(n \cdot log(n) + n^2) \subset O(n^2)$.

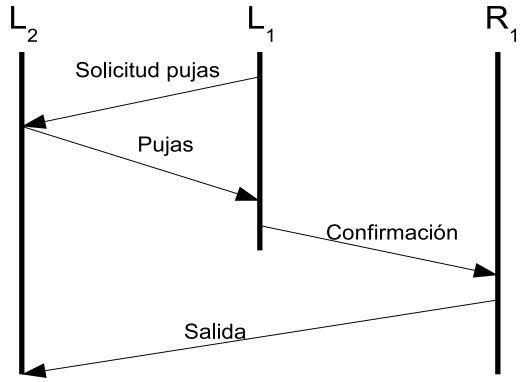


Figura 4.3: Mensajes transmitidos entre los diferentes robots para permitir una salida de grupo. R_1 robot seleccionado para salir del grupo, L_2 robot líder del grupo donde estaba R_1 y L_1 líder del grupo de destino.

Si durante la negociación 'líder a líder' un robot cambia de grupo, ha de avisar de este hecho al líder del grupo de trabajo del que formaba parte inicialmente. La figura 4.3 muestra los mensajes necesarios para que un robot R_1, que inicialmente formaba parte del grupo liderado por el robot L_2, pase a formar parte del grupo del líder L_1. En primer lugar L_1 inicia la negociación 'líder a líder' solicitando el envío de pujas

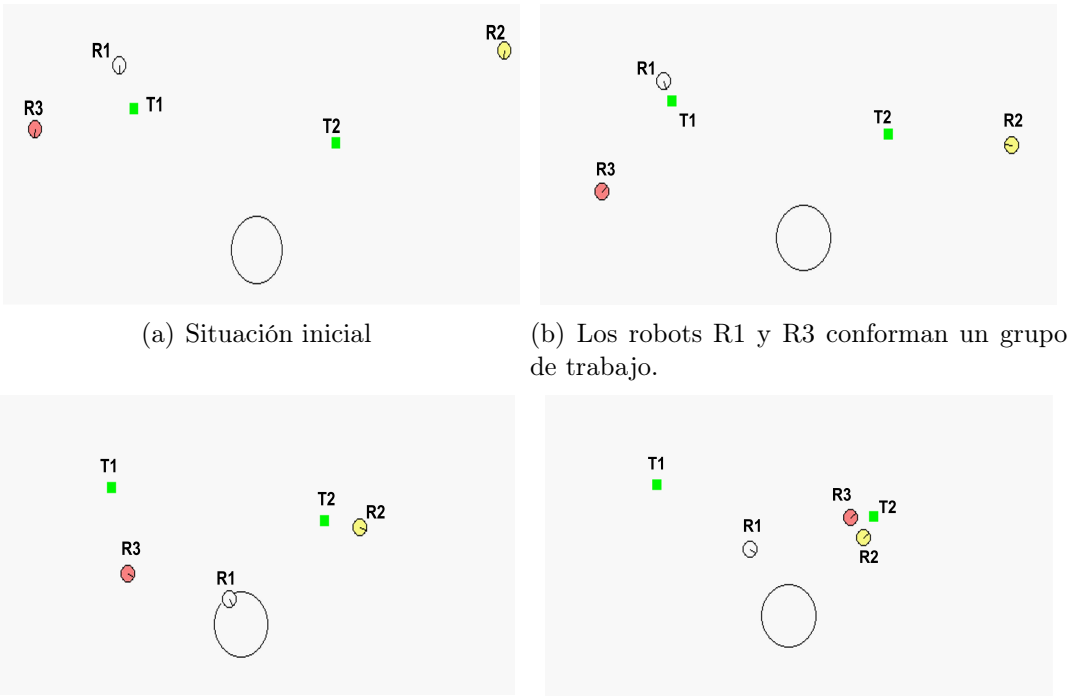
Algoritmo 11 Algoritmo de subasta tarea t con negociación 'líder a líder'

1: Enviar solicitud de pujas para la tarea t
2: $B_r \leftarrow \emptyset$
3: $B_g \leftarrow \emptyset$
4: $tiempo_{inicio} \leftarrow tiempo$
5: **mientras** $(tiempo - tiempo_{inicio}) \leq TIME\_AUCTION$ **hacer**
6: **si** puja b_i recibida del robot libre r_i **entonces**
7: $B_r \leftarrow B_r \bigcup b_i$
8: **fin si**
9: **si** puja b_k recibida del líder del grupo k **entonces**
10: $B_g \leftarrow B_g \bigcup b_k$
11: **fin si**
12: **fin mientras**
13: Ordenar B_r de mayor a menor capacidad de trabajo
14: Inicializar TH_g
15: $j \leftarrow 0$
16: **mientras** $j \leq |B_r|$ **y** $TH_g \geq TH$ **hacer**
17: Seleccionar robot puja B[j]
18: Actualizar TH_g
19: $j \leftarrow j + 1$
20: **fin mientras**
21: **mientras** $B_g \neq \emptyset$ **y** $TH_g \geq TH$ **hacer**
22: Seleccionar el robot con la mayor puja $b_{jk} \in B_g$ y que cumpla la condición 4.4
{b_{jk} valor de puja B del robot j del grupo k}
23: $B_g \leftarrow B_g - b_{jk}$
{Para todas las pujas del del grupo k}
24: **para todo** $b_{lk} \in B_g$ **hacer**
25: Modificar valor de puja b_{lk}
26: **fin para**
27: Actualizar TH_g
28: **fin mientras**

al resto de líderes. Como respuesta a este mensaje, L_2 envía su puja. L_1 tras recibir todas las pujas, selecciona al robot R_1 para que forme parte de su grupo de trabajo enviándole un mensaje de confirmación. El robot R_1 tras recibir la confirmación de L_1, envía un mensaje de salida de grupo a su antiguo líder L_2. Cuando un líder recibe de uno de sus robots un mensaje de abandono de grupo, ha de restar la capacidad de trabajo de este robot a la capacidad total del grupo y volver a calcular el valor de TH_g. Si el nuevo valor de TH_g no verifica la condición 4.1, el líder inicia un nuevo proceso de subasta utilizando la negociación 'líder a líder'. Se ha de tener en cuenta que el propio líder puede decidir abandonar a su grupo, en este caso se pondrá en marcha el proceso de substitución de líder explicado la sección 4.3.4.

Si tras ejecutar el algoritmo de subasta la condición 4.1 sigue sin cumplirse, el líder inicia otra ronda de subastas. Por último, cuando una tarea finaliza, el grupo de trabajo se disuelve inmediatamente.

La figura 4.4 muestra la secuencia de acciones durante un proceso de subasta 'líder a líder' donde aparecen 3 robots ($R1$, $R2$ y $R3$) que han de recoger 2 objetos ($T1$, $T2$). El objeto $T1$ tiene un peso de 15 unidades y el $T2$ de 20. La figura 4.4(a) muestra la disposición inicial de los tres robots y de los objetos. En la figura 4.4(b) se ve como el robot $R1$ se ha convertido en el líder de la tarea de recogida del objeto $T1$ y empieza a cargarlo. A su vez, el robot R3 ha aceptado entrar a formar parte del grupo liderado por el robot $R1$. El robot $R2$ se ha convertido en el líder de la tarea $T2$ y se encamina hacia ella. En la figura 4.4(c) se puede ver como el robot $R1$ realiza la descarga de la parte del objeto que ha cargado previamente. Debido a que el objeto $T2$ es más pesado, y requiere mayor capacidad de trabajo, el robot $R2$ (líder) inicia una subasta 'líder a líder'. Como resultado de esta subasta, el robot $R3$ pasa a formar parte del grupo del robot $R2$, colaborando en la misma tarea, tal como se puede ver en la figura 4.4(d).



(a) Situación inicial

(b) Los robots R1 y R3 conforman un grupo de trabajo.

(c) El robot R1 deja el objeto en la zona de descarga.

(d) Cambio de grupo. Los robots R2 y R3 colaboran en la misma tarea.

Figura 4.4: Ejemplo de recogida de dos objetos utilizando la negociación 'líder a líder'.

4.3.4. Mecanismos de substitución de líder

Puede suceder que durante la negociación 'líder a líder' uno de los robots seleccionados para cambiar de grupo sea el propio líder. Si el líder seleccionado es el único robot que forma parte del grupo de trabajo, entonces éste rechaza la oferta de cambio de grupo. Si el líder acepta cambiar de grupo, se pone en marcha el proceso de substitución de líder, durante el cual el antiguo líder transferirá toda la información sobre el grupo de trabajo al robot que se convierta en el nuevo líder. Esta información consistirá en el identificador y la capacidad de trabajo de todos los robots que forman parte del grupo. El proceso que se lleva a cabo para substituir al líder es el siguiente: el líder selecciona como su substituto al robot del grupo con mayor capacidad de trabajo y le transmite toda la información. El robot seleccionado como nuevo líder puede rechazar el liderazgo debido a que, por ejemplo, ha entrado a formar parte como líder

de otro grupo de trabajo. En este caso, este último robot pasa el liderazgo, junto con toda la información del grupo, al siguiente robot con mayor capacidad de trabajo. Si este nuevo robot tampoco acepta ser líder, se pasa al siguiente, y así sucesivamente. Puede suceder que el último robot que quede en la lista haya pasado a ser líder de otra tarea, en la que él sea el único miembro del grupo de trabajo. En este caso, el robot no deja su tarea actual, sino que envía un mensaje a todos los otros indicando que hay una tarea sin robots asignados. La figura 4.5 muestra los mensajes enviados cuando inicialmente R_1 es el líder y abandona el grupo. Ninguno de los robots que inicialmente formaban el grupo ($R_2..R_n$) ya forma parte de él, y todos son líderes de otras tareas. El robot R_n es además el único miembro de su grupo de trabajo, por lo que no acepta la petición de liderazgo. El caso que sea acaba de describir es excepcional. Por lo general la transferencia del liderato concluye en unos pocos intentos y la coalición continua su tarea con la nueva estructura.

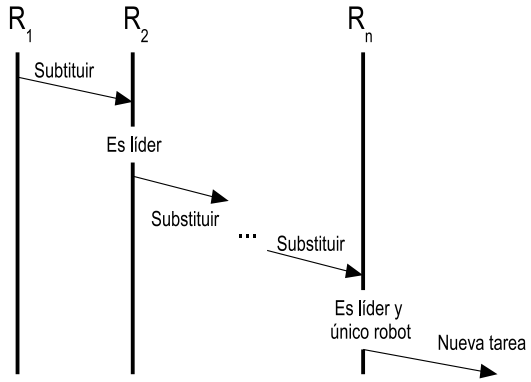


Figura 4.5: Mensajes transmitidos a lo largo del tiempo durante el proceso de substitución de líder. $R_1..R_n$ son los robots que había inicialmente en el grupo.

4.3.5. Mecanismos de segregación y agregación

A parte de la negociación 'líder a líder', existen dos mecanismos más para poder variar el tamaño de un grupo de trabajo durante la ejecución de la tarea: el mecanismo de segregación y el mecanismo de agregación. El mecanismo de segregación permite disminuir el tamaño del grupo, mientras que el de agregación lo aumenta.

El mecanismo de segregación se pone en marcha cuando un robot, que no es líder, detecta una tarea de prioridad mayor o igual a la que actualmente está ejecutando sin ningún robot asignado. En este caso, el robot intenta abandonar su actual grupo para convertirse en el líder de esta nueva tarea utilizando el algoritmo 9. Si el abandono del grupo finalmente se lleva a cabo, el robot segregado comunica esta circunstancia a su antiguo líder, quien a su vez vuelve a calcular el valor de TH_g y, si es necesario, inicia un nuevo proceso de subasta. La figura 4.6 muestra algunos de los mensajes transmitidos durante la ejecución del mecanismo de segregación, donde R_1 es el robot que se quiere segregar, 'Líder' es el antiguo líder de R_1, y 'Robots' representa al resto de robots del sistema. El mecanismo de segregación tiene como objetivo que no haya tareas sin robots asignados y, de esta manera, reducir el tiempo medio de finalización de las tareas. Además, al producirse una distribución más uniforme de los robots entre las diferentes tareas, también se reducen los efectos de la interferencia como consecuencia del aumento de la distribución espacial de los robots.

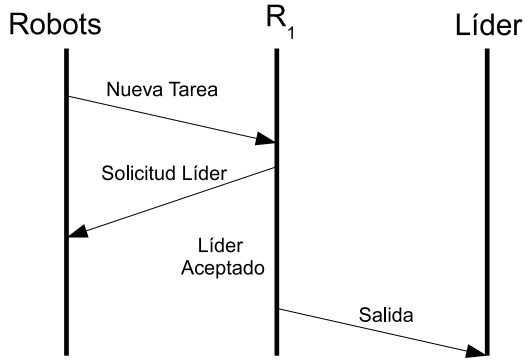


Figura 4.6: Mensajes transmitidos durante la ejecución del mecanismo de segregación, donde R_1 es el robot que se quiere segregar, 'Líder' es el antiguo líder de R_1 y 'Robots' representa al resto de robots del entorno.

Por otra parte, el tamaño del grupo puede aumentar mediante el mecanismo de agregación. Este mecanismo se activa cuando el robot lleva un tiempo sin ninguna tarea asignada, momento en el cual envía un mensaje a todos los robots solicitando ejecutar alguna tarea y entrar a formar parte de algún grupo de trabajo. Si un líder recibe uno de estos mensajes, acepta al robot, siempre que el número de robots del

grupo sea inferior a una determinada cantidad. Se supone que añadir un miembro más a un grupo con pocos robots no incrementa excesivamente la interferencia. En los experimentos llevados a cabo para validar los algoritmos, este límite de robots es 3, debido al número de tareas y robots utilizados. De esta manera, se evita tener miembros inactivos en el sistema sin aumentar excesivamente la interferencia física entre robots. La figura 4.7 muestra los mensajes transmitidos durante el proceso de agregación, donde R_1 es el robot que solicita la agregación y *Líder*, es el líder que acepta a R_1 en su grupo.

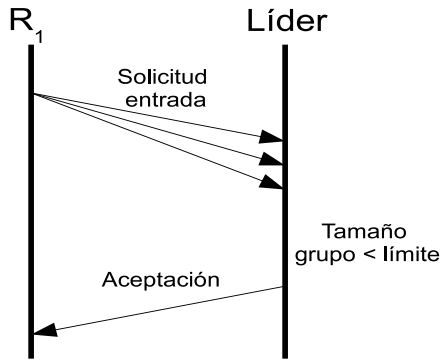


Figura 4.7: Mensajes transmitidos en la ejecución del mecanismo de agregación, donde R_1 representa al robot a agregar.

4.3.6. Tratamiento de excepciones

Definimos como situaciones excepcionales aquellas ocasionadas por los fallos de comunicación y las averías de los robots. En esta sección se analizan estas situaciones y se explican los mecanismos implementados para solucionar los problemas que pueden ocasionar. Los mecanismos aquí descritos están parcialmente inspirados en los trabajos de Sariel [111].

Duplicidad de líderes

El primer problema que puede surgir en nuestro sistema es que haya más de un líder para una misma tarea. Esto podría suceder si se perdiesen los mensajes

de solicitud de liderazgo enviados por el robot o si estos llegasen pasado el tiempo máximo de espera del algoritmo 9. Si esto llegase a suceder, el objeto podría ser transportado igualmente, sólo se perdería eficiencia al no poder calcular el tamaño del grupo de manera correcta. De todas maneras, se ha implementado un mecanismo de seguridad para impedir que esto acabe pasando. En primer lugar, se asignará un identificador único a cada robot antes de iniciar la ejecución de la misión. El líder periódicamente envía mensajes a todos los robots (mensaje *LEADER_ALIVE(id, pos)*) indicando que es líder de una tarea. Este mensaje incluirá el identificador del líder (*id*), y la posición donde se encuentra el objeto a recoger (*pos*). La posición del objeto a recoger se utilizará como identificador de la tarea que se realiza, ya que se supone que no puede haber varios objetos simultáneamente en un mismo lugar. Si el mensaje enviado por un líder L_1 es recibido por un robot L_2 que también es líder de la misma tarea, pero tiene un identificador menor a L_1, entonces L_2 abandonará el liderato y comunicará este hecho a los robots de su grupo enviando el mensaje de *EXIT_LEADER*. Si el robot que recibe el mensaje de *LEADER_ALIVE* está a la espera de que se le conceda ser líder de esta misma tarea, entonces abandonará inmediatamente su pretensión.

Fallo de un líder

Otro posible problema que puede aparecer es que el líder tenga una avería y no pueda continuar la misión. Para detectar este tipo de situaciones también se utilizan los mensajes periódicos de *LEADER_ALIVE*. Si un robot que pertenece a un grupo no recibe ningún mensaje de su líder en un determinado período de tiempo (*TIME_LEADER_ALIVES*), abandonará el grupo de trabajo pasando a estado de libre.

Robots obsoletos en el grupo

Los robots pertenecientes a un grupo enviarán periódicamente, cada *TIME_BETWEEN_ALIVES*, un mensaje (*ROBOT_ALIVE*) a su líder indicando

que forman parte de su grupo y que siguen en funcionamiento. Si un líder no recibe ningún mensaje de alguno de sus robots miembros durante un determinado período de tiempo ($TIME\_BETWEEN\_REMOVE$), procederá de manera similar a cuando recibe un mensaje de rechazo en el proceso de subasta. Por tanto, volverá a calcular el valor de TH_g y, en el caso de que fuera necesario, iniciaría un nuevo proceso de subasta.

A modo de resumen, la figura 4.8 muestra el autómata que representa las transiciones entre los estados del robot durante un proceso de subasta: libre, miembro (de un grupo de trabajo), líder y espera de líder (ejecutando el algoritmo 9), en función de los mensajes recibidos. Además, desde cualquier estado el robot pasa a 'libre' cuando la tarea finaliza.

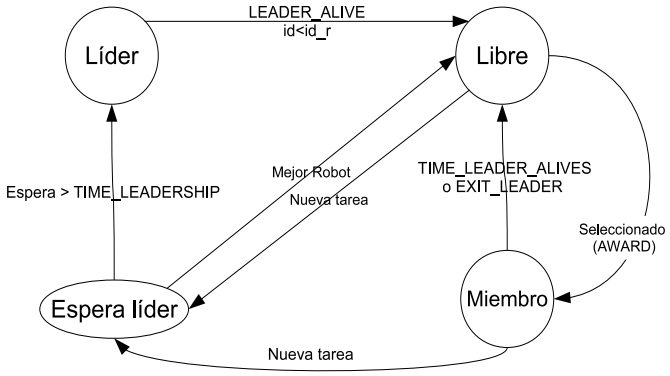


Figura 4.8: Autómata que representa las transiciones entre los estados de los robots.

4.4. Coordinación del movimiento: *lane keeping*

Como ya se ha comentado, en un sistema multi-robot, se han de abordar tres problemas básicos: planificación, asignación de tareas y coordinación del movimiento. En el anterior apartado se ha propuesto un método para la asignación de tareas que reduce la interferencia física. En esta sección se explica un nuevo método de coordinación del movimiento, llamado 'mantenerse en el carril' (*lane keeping*), que se integra con el mecanismo de subastas ya explicado, también con el objetivo de

reducir la interferencia física. Cabe recordar que el estudio de la coordinación no es un objetivo básico de este trabajo. Los mecanismos aquí planteados sólo pretenden ser un ejemplo de cómo con la coordinación del movimiento de los robots también puede reducirse la interferencia.

El método *lane keeping* está especialmente pensado para reducir la interferencia en tareas de recogida de objetos, como la que se ha utilizado para validar el método de subastas, donde los robots han de circular en múltiples ocasiones entre el objeto y el punto de descarga. Este método se centra especialmente en la interferencia que se da cuando dos robots de un mismo grupo se encuentran circulando en la misma dirección y sentidos opuestos. En el caso de la tarea de recogida de objetos aquí propuesta, esta situación se produce cuando un robot que va hacia el objeto a recoger se encuentra con otro que se dirige al punto de descarga, o viceversa. El grado de interferencia en estos casos es mayor que en otras situaciones.

El mecanismo *lane keeping* está inspirado en el comportamiento que tienen los coches que circulan en una carretera de doble sentido. Los coches que circulan en sentidos contrarios también ocupan partes diferentes de la calzada, unos van por la derecha y los otros por la izquierda. *Lane keeping* hace que los robots se comporten precisamente de esta manera. En primer lugar se define cual es el camino, o 'carretera virtual', que los robots han de seguir para ir del punto de descarga al objeto y viceversa. Una vez hecho esto, los robots que se dirijan hacia el objeto a recoger irán por una lado de este camino, mientras que los que vayan al punto de descarga circularán por el lado contrario, tal como se puede observar en la figura 4.9. La decisión del camino a seguir entre objeto y punto de descarga la toma el líder del grupo de trabajo quien, siempre que sea posible, elige la línea recta como camino. En el caso de entornos con obstáculos, los caminos se definen como segmentos rectos, uniendo los *way points* necesarios para sortear dichos obstáculos.

En el contexto del presente trabajo el movimiento de los robots está controlado mediante una arquitectura basada en comportamientos reactivos, implementados usando campos de potencial, en la que hay un comportamiento que indica la dirección

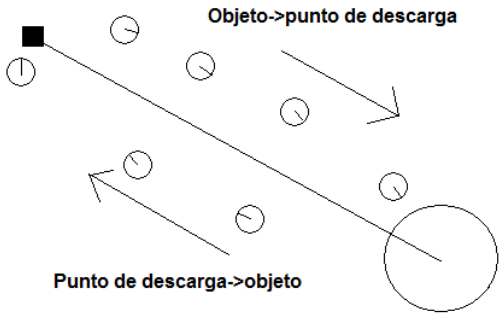


Figura 4.9: Ejemplo de la estrategia *lane keeping*.

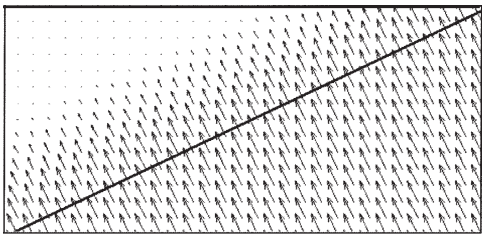


Figura 4.10: Campo de Fuerza generado por el comportamiento *lane keeping* en el momento en el que el robot tiene que circular por la parte superior de la línea de separación.

que ha de seguir el robot para ir hacia el objetivo, otro que indicará la dirección que se ha de seguir para evitar los obstáculos, etc. En el anexo A se explica detalladamente esta arquitectura. Para implementar la estrategia de *lane keeping*, se ha añadido a los comportamientos ya existentes otro, también llamado *lane keeping*, que crea una fuerza que atrae al robot hacia el lado por el que ha de circular. La dirección y el sentido de esta fuerza es perpendicular al camino, como se puede ver en la figura 4.10, donde se muestra la magnitud y dirección de la fuerza *lane keeping* cuando el robot ha de circular por encima de la línea de separación. Se puede observar que el robot continuará sometido a esta fuerza mientras no se encuentre en el lado correcto y a más de una determinada distancia de la línea de separación. De esta manera, se garantiza una cierta distancia de seguridad entre los robots que circulan en sentidos opuestos. La magnitud de esta fuerza ($|V_s|$) se calcula utilizando la siguiente ecuación:

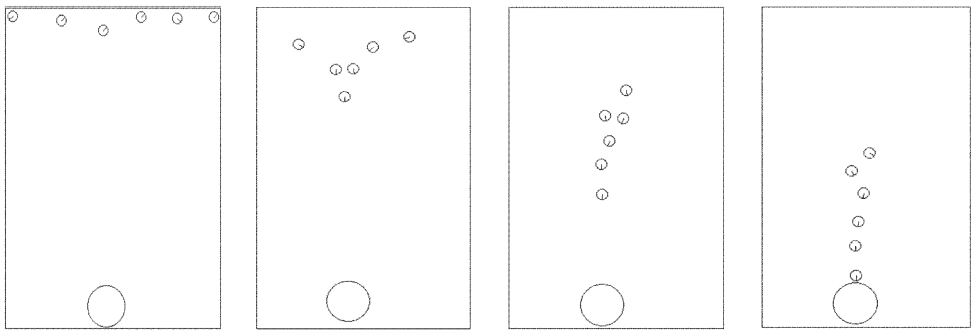
$$|V_s| = \begin{cases} 1 & \text{si } \overline{Circ} \text{ ó } d \leq D_{min} \\ \frac{(D_{max}-d)}{(D_{max}-D_{min})} & \text{si } Circ \text{ i } D_{min} < d \leq D_{max} \\ 0 & \text{de otra forma} \end{cases} \quad (4.5)$$

donde $Circ$ es una expresión booleana que indica si el robot está circulando por el lado correcto, d es la distancia entre el robot y la línea de separación que indica el camino a seguir y D_{min}, D_{max} son los parámetros que determinan la distancia de seguridad entre los robots que circulan por cada uno de los lados.

El comportamiento *lane keeping* se dejará de aplicar, es decir $|V_s|$ valdrá 0, cuando el robot se encuentre muy próximo a su objetivo (punto de descarga u objeto), de esta manera se facilita el acceso simultáneo de múltiples robots. También se dejará de aplicar cuando se detecten obstáculos de gran tamaño, como paredes, que impidan seguir el camino establecido y circular por el lado correcto. Se considera que los robots se encuentran en una situación de este tipo cuando la magnitud de la fuerza del comportamiento de evitar obstáculos sea mayor a un determinado umbral durante un cierto período de tiempo.

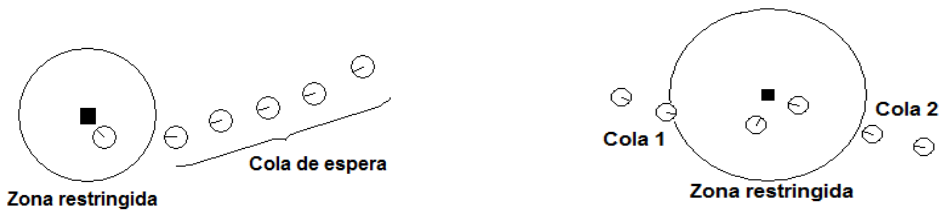
A parte de la estrategia *lane keeping* también se han probado otras muy similares a las ya utilizadas por otros autores para reducir la interferencia física. Se implementó una estrategia de *flocking*, llamada *follow the preceding*, similar a la propuesta en [57] por Goldberg y Mataric, en la que los robots siguen siempre al miembro de su grupo que tienen delante, formando una agrupación para ir hacia un punto objetivo. La figura 4.11 muestra como los robots se mueven en grupo utilizando la estrategia *follow the preceding*. También se implementó un mecanismo de división del entorno de trabajo similar al propuesto en [58], donde se restringe el número de robots que pueden acceder simultáneamente al objeto o a la zona de descarga. De esta manera, los robots forman colas esperando para acceder a estas zonas, tal como se puede ver en las figura 4.12. Los resultados experimentales llevados a cabo mostraron que el uso de estas dos estrategias era, en la mayoría de los casos, contraproducente, incrementando el tiempo de ejecución. Más recientemente, otros autores también han propuesto métodos de división espacial para reducir la interferencia, como Lein y

Vaughan [89] o Pini et al. [107], permitiendo sólo un robot por objeto.



(a) Situación inicial (tiempo=0). (b) tiempo=20. (c) tiempo=50. (d) tiempo=80.

Figura 4.11: Evolución temporal de la estrategia *follow the preceding*. Los robots han de ir desde su posición inicial hasta el círculo situado en la parte inferior.



(a) Limitado a 1 robot. (b) Limitado a 2 robots.

Figura 4.12: Cola de espera para acceder al objeto.

4.5. Resultados experimentales

En esta sección se explican los experimentos llevados a cabo para validar tanto el método de asignación de tareas como el mecanismo de coordinación de movimiento *lane keeping*. Se evaluará el impacto que tienen sobre el sistema parámetros tales como: el valor umbral (TH) de la condición 4.1, los diversos parámetros de los robots y de las tareas, así como los distintos aspectos del método de formación de coaliciones descritos hasta este punto.

4.5.1. Descripción del entorno y de la tarea a ejecutar

Para obtener los resultados experimentales se ha utilizado el simulador *Robot Colonies Tool* (RoboCoT), descrito al inicio de este capítulo. La arquitectura de los robots implementados en RoboCoT está basada en comportamientos reactivos, siguiendo la estructura de *Motor Schemas* propuesta por Arkin [3]. En el anexo A se puede encontrar una descripción más detallada de la arquitectura y de los comportamientos implementados. La estrategia *lane keeping* se implementa como un comportamiento más y se integra con el resto.

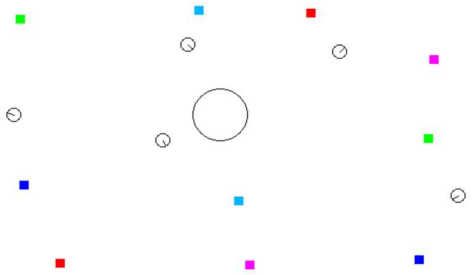


Figura 4.13: Ejemplo de la situación inicial antes de comenzar la ejecución de una simulación con RoboCoT.

En todos los experimentos realizados se ha llevado a cabo la misión de recolección de objetos, explicada en la sección 4.1. El conjunto de robots que colaboran en el transporte de un mismo objeto formará el grupo de trabajo descrito en la sección 4.3. El peso de un objeto t_j corresponde al valor $taskWorkLoad_j$ y la capacidad de carga de un robot r_i al valor $workCapacity_{i,j}$, valor este último que a su vez el robot utilizará como puja en el proceso de subasta. La figura 4.13 muestra un ejemplo de la disposición inicial de objetos y robots antes de iniciar la simulación con RoboCoT. Los pequeños cuadrados representan los objetos a recoger y el círculo de mayor tamaño del centro de la imagen, la zona de descarga. Cuando un objeto ha sido totalmente transportado, aparecerá inmediatamente otro con las mismas características pero en una posición aleatoria. La simulación finalizará tras haber transcurrido un

determinado período de tiempo. De esta manera, se mantienen en todo momento las condiciones iniciales del entorno, ya que siempre hay el mismo número de tareas a realizar.

4.5.2. Entropía del sistema

Uno de los objetivos de los experimentos es estudiar cómo varía la eficiencia del sistema propuesto en función de lo diferentes que son entre sí los robots, esto es, de la heterogeneidad del colectivo. Por simplicidad, se supone que los robots pueden diferenciarse únicamente por su capacidad de carga, siendo iguales el resto de sus características. Para medir el nivel de heterogeneidad de los robots se utilizarán los conceptos de entropía social simple y entropía social jerárquica. El valor de la entropía social simple sigue la conocida fórmula de la entropía de la información de Shannon (H), que tiene la siguiente forma:

$$H = \sum_{1 \leq i \leq M} p_i \log_2(\frac{1}{p_i}) \tag{4.6}$$

donde M es la cantidad de clases de robots que existen y p_i es la proporción de robots de la clase i. Diremos que dos robots pertenecen a una misma clase si los dos tienen exactamente la misma capacidad de carga. Entonces, un grupo formado por 4 robots con las siguientes capacidades de carga: 1, 1, 2 i 2 unidades, tendría la misma entropía simple que uno formado por robots con capacidades de carga de 1, 1, 500 i 500 unidades. A pesar de que ambos sistemas tienen la misma entropía simple, es razonable pensar que el segundo grupo es mucho más heterogéneo que el primero, debido a que las diferencias entre grupos son muy grandes.

Para cuantificar las diferencias entre los sistemas anteriormente indicados se utilizará la entropía social jerárquica propuesta por Balch [7], que sí tiene en cuenta las diferencias cuantitativas entre los diferentes grupos. Para calcular su valor, en primer lugar se utilizan técnicas similares a las utilizadas en biología para clasificar organismos, donde se crea un árbol taxonómico. Las hojas de este árbol representan

los grupos de robots con exactamente las mismas características. En el siguiente nivel del árbol cada nodo representa un grupo de robots para los que la diferencia entre sus capacidades de carga es menor o igual a un valor h. El siguiente nivel representa grupos de robots con una diferencia en su capacidad de carga menor o igual a $2 \cdot h$, en el siguiente nivel la diferencia ha de ser de $3 \cdot h$, y así sucesivamente. De esta manera, se van agrupando los robots hasta que todos ellos acaban perteneciendo a un mismo grupo. En los experimentos realizados h se ha fijado a 1. Una vez obtenida esta estructura, se calcula la entropía simple de cada uno de los niveles del árbol utilizando la expresión 4.6, considerando a cada uno de los nodos como una clase diferente de robots. Se denotará por $H(R, h)$ a la entropía simple calculada cuando la diferencia máxima entre miembros de un mismo grupo es igual a h, el valor R representa al conjunto de todos los robots del sistema. Finalmente, el valor de la entropía social jerárquica (SH) se calcula utilizando la siguiente expresión:

$$SH(R) = \int_0^\infty H(R, h)dh \tag{4.7}$$

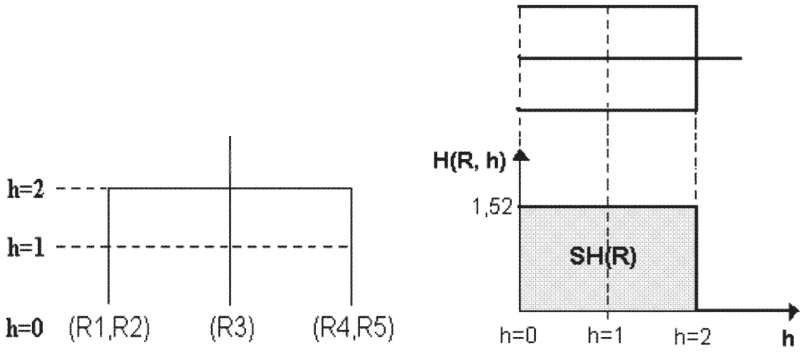
La tabla 4.2 muestra los valores de la entropía simple y jerárquica para los diferentes grupos de robots utilizados en los experimentos. Nótese que la capacidad total de trabajo del grupo es la misma para las 5 configuraciones mostradas. La figura 4.14 muestra gráficamente cómo se ha calculado el valor de la entropía social jerárquica para la configuración número 3, es decir, cuando los robots R_1 y R_2 tienen una capacidad de carga de 1, R_3 una capacidad de 3 y los robots R_4 y R_5 una capacidad de 5. Como se puede observar en la figura 4.14(b), la entropía social jerárquica corresponde al área sombreada de la parte inferior.

4.5.3. Recogida de objetos sin prioridades

En esta sección se analizan los resultados obtenidos sin prioridades asociadas a los objetos y tareas homogéneas. Además, se han comparado los resultados obtenidos utilizando y sin utilizar la estrategia preemptiva, esto es, utilizando la negociación 'líder a líder' o únicamente permitiendo la negociación 'líder a robot'. En todos los

Configuración	R_1	R_2	R_3	R_4	R_5	H	SH
1 (Homogénea)	3	3	3	3	3	0	0
2	1	1	1	1	11	0,72	7,22
3	1	1	3	5	5	1,52	3,04
4	1	2	3	4	5	2,32	2,32

Tabla 4.2: Capacidad de carga de los robots utilizados en los experimentos. $R_1...R_5$ representan la capacidad de carga de los robots. H es el valor de la entropía simple y SH la entropía social jerárquica.



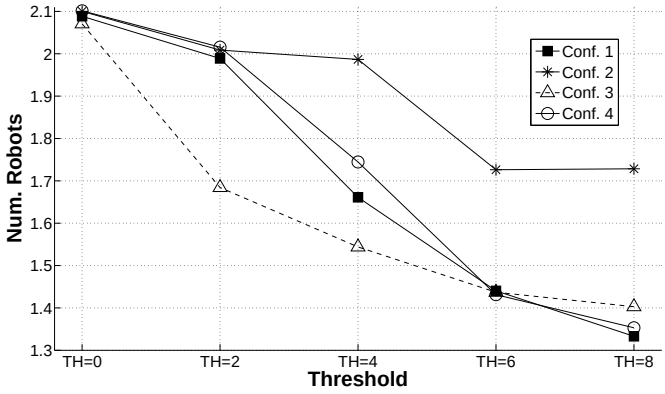
(a) Árbol previo al cálculo de la entropía.

(b) Cálculo de la entropía SH a partir del árbol. SH es igual a la zona sombreada.

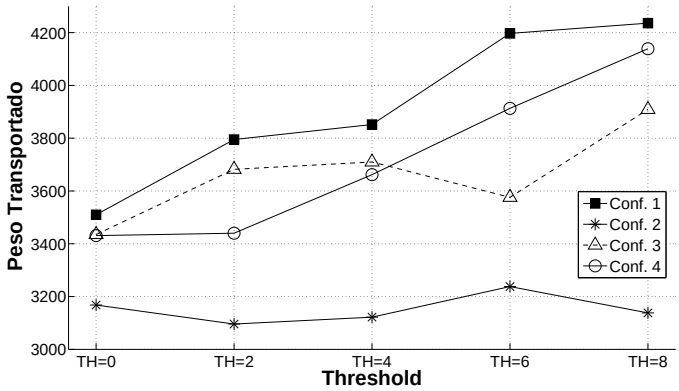
Figura 4.14: Cálculo de la entropía social jerárquica para la configuración número 3 de robots.

entornos se han utilizado 5 robots y 10 objetos para recoger. El peso de los objetos siempre es de 30 unidades y su prioridad varía entre 1 y 5. El valor del umbral TH tomará los siguientes valores: 0, 2, 4, 6 y 8. Si el umbral TH vale 0 quiere decir que la condición 4.1 no se utiliza y que, por tanto, el tamaño del grupo de trabajo no está limitado. Los robots han de llevar a cabo su misión durante 35.000 unidades de tiempo, tras el cual se calcula el peso total transportado por todos los robots o el tiempo medio necesario para finalizar una tarea, entre otros parámetros.

La figura 4.15 muestra algunos resultados obtenidos utilizando tareas sin prioridades y sin utilizar la estrategia preemptiva. Tal como se puede ver en la figura 4.15(a),



(a) Número medio de robots.



(b) Peso total transportado.

Figura 4.15: Resultados sin prioridades y sin utilizar la estrategia preemptiva.

a medida que se incrementa el valor de TH decrece el número medio de robots asignados a una misma tarea. Al reducirse el tamaño de los grupos de trabajo, también se reducen los efectos de la interferencia física, incrementándose el peso total transportado, tal como se puede ver en la figura 4.15(b). Estos resultados también muestran como en la mayoría de casos, el peso total transportado se incrementa respecto al sistema con $TH = 0$, esto es, respecto a un sistema que no utiliza la desigualdad 4.1, llegando a mejoras de un 20 %. La configuración de robots con una entropía social menor (configuración 1), es la que muestra mejores resultados, por contra, la configuración con la mayor entropía social (configuración 2) es la que presenta peores resultados. Además, la configuración 2 es la única que parece no estar afectada por el valor de TH ya que no presenta casi variaciones en el peso total transportado. Este hecho constata que una configuración homogénea permite una mejor distribución de las capacidades de carga de los robots entre las diferentes tareas.

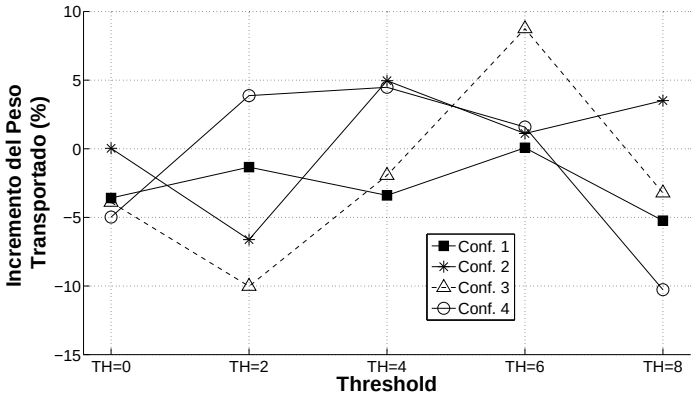


Figura 4.16: Incremento porcentual en el peso total transportado utilizando la estrategia preemptiva respecto al peso transportado sin la estrategia preemptiva. Las tareas no tienen una prioridad asociada.

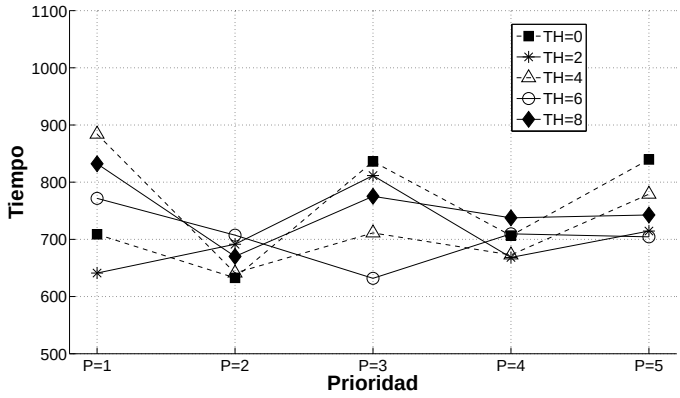
La figura 4.16 muestra el incremento porcentual del peso total transportado utilizando y sin utilizar la estrategia preemptiva sin prioridades en las tareas. Un porcentaje mayor a 0 indica que la estrategia preemptiva incrementa el peso transportado. Tal como se puede observar, ambas estrategias producen resultados similares, ya que

en todos los casos la diferencia es menor a un 10 %. No obstante, y en términos generales, cuando los valores de TH son muy bajos o muy altos, los beneficios de la estrategia preemptiva disminuyen. Cuando el umbral TH es bajo, el número de robots asignados a cada tarea es excesivo y por tanto, el sistema necesita intercambiar un gran número de robots entre grupos de trabajo. En este caso, el tiempo necesario para estabilizar el sistema puede ser mayor que el necesario para finalizar la tarea. En cambio, cuando TH es muy alto, el número de robots en cada grupo es demasiado pequeño como para permitir el intercambio entre ellos, lo que disminuye los potenciales beneficios de la estrategia preemptiva. Como se expondrá en el siguiente apartado, cuando se asignan prioridades a las tareas, la estrategia preemptiva mejora claramente los resultados.

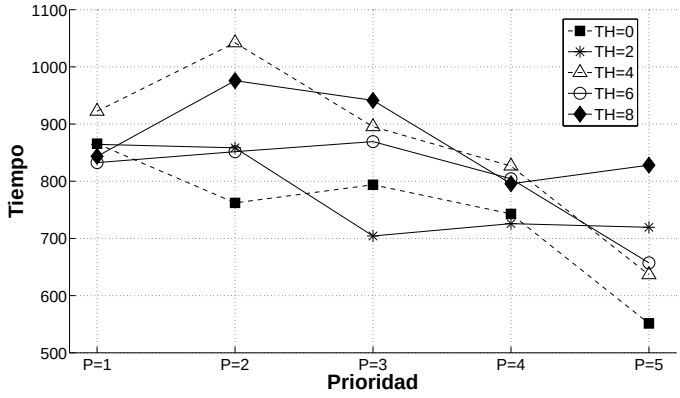
4.5.4. Recogida de objetos con prioridades

A continuación se evaluará el método de subastas propuesto utilizando tareas con prioridades asociadas. Las características de las simulaciones son iguales que en el caso de las tareas sin prioridades, sólo que ahora habrá 2 tareas de cada prioridad: 2 con prioridad 1, 2 con prioridad 2, etc. La figura 4.17(a) muestra el tiempo medio necesario para recoger completamente un objeto en función de su prioridad sin utilizar la estrategia preemptiva, con la configuración 3 de robots y para diferentes valores de TH. En estos primeros experimentos se ha utilizado la configuración 3 de robots ya que, respecto al resto de configuraciones, presenta valores medios de entropía social y simple. Los valores de la desviación estándar (σ) de estos tiempos se pueden ver en la tabla 4.3. Tal como se puede observar, no hay correlación entre la prioridad de la tarea y su tiempo de ejecución ya que estos tiempos dependen mucho de la posición donde se encuentre la tarea respecto al punto de descarga o la posición de los robot, entre otros factores. La figura 4.17(b) muestra los resultados de los mismos experimentos pero utilizando la estrategia preemptiva y en la tabla 4.4 se puede ver la desviación estándar (σ) de estos resultados. Ahora sí que existe una correlación entre la prioridad de la tarea y el tiempo de ejecución, de manera que tareas más

prioritarias finalizan antes que tareas con una prioridad menor.



(a) Resultados sin utilizar la estrategia preemptiva.



(b) Resultados utilizando la estrategia preemptiva.

Figura 4.17: Tiempo medio necesario para transportar completamente un objeto con prioridades utilizando la configuración 3 de robots.

Finalmente, la figura 4.18 muestra el incremento, en tanto por ciento, del tiempo de ejecución de una tarea de la máxima prioridad (prioridad 5) cuando no se usa la estrategia preemptiva, respecto al tiempo de ejecución utilizando la estrategia preemptiva. En esta figura se pueden observar las diferentes configuraciones de entropía analizadas. En la gran mayoría de casos la estrategia preemptiva presenta tiempos de ejecución menores respecto a la no preemptiva, aumentando esta mejora a medida que

-	P=1	P=1	P=3	P=4	P=5
TH=0	435,1	249,6	504,8	369,6	774,9
TH=2	345,1	292,4	605,0	295,1	615,0
TH=4	596,9	318,6	397,4	337,4	497,9
TH=6	402,2	373,3	295,3	317,3	395,3
TH=8	394,4	439,2	417,8	379,8	352,9

Tabla 4.3: Desviación estándar (σ) del tiempo necesario para transportar completamente un objeto en función de su prioridad, sin utilizar la estrategia preemptiva. P1, ..., P=5 representan los diferentes valores de prioridad.

-	P=1	P=1	P=3	P=4	P=5
TH=0	522,1	555,3	487,7	547,4	348,4
TH=2	532,1	749,9	446,6	332,4	403,1
TH=4	583,7	709,0	646,5	593,3	357,6
TH=6	319,6	496,5	665,8	448,5	330,8
TH=8	367,5	791,8	417,2	406,1	504,0

Tabla 4.4: Desviación estándar (σ) del tiempo necesario para transportar totalmente un objeto en función de su prioridad utilizando la estrategia preemptiva. P=1, ..., P=5 representan los diferentes valores de prioridad.

la entropía social disminuye. Por ejemplo, la configuración 1 ($SH = 0$) presenta mejoras en todos los casos gracias a la estrategia preemptiva, llegando a aumentar más de un 50 % el tiempo de ejecución debido al uso del método no preemptivo. En cambio, el uso de la estrategia preemptiva en la configuración con mayor entropía social (configuración 2) únicamente mejora los resultados cuando $TH = 6$ ó $TH = 8$. Las diferencias en cuanto a peso total transportado, utilizando y sin utilizar *preemption* no son significativas, siguiendo un patron similar al presentado en los experimentos sin prioridades. Por tanto se puede concluir, que la estrategia preemptiva es especialmente útil en tareas con prioridades asociadas, ya que disminuye el tiempo medio de ejecución de las tareas más prioritarias. Esta mejora es especialmente destacable cuando la entropía social del grupo es baja, por tanto una baja entropía social permite una distribución más eficaz de los robots entre las tareas.

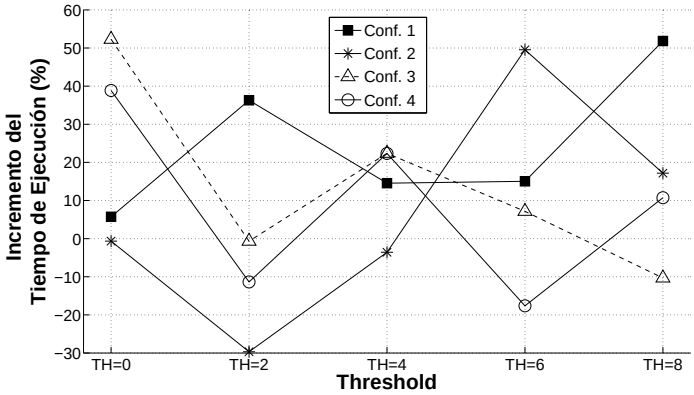


Figura 4.18: Incremento porcentual en el tiempo necesario para finalizar una tarea de prioridad 5 sin utilizar la estrategia peemptiva con respecto al tiempo necesario utilizando la estrategia preemptiva.

Experimentos utilizando tareas heterogéneas

Hasta ahora todos los objetos eran del mismo tipo y, por tanto, la capacidad de carga no dependía de la tarea. En los experimentos con tareas heterogéneas la capacidad de carga del robot dependerá de las características de los objetos. Para ello, se han agrupado las tareas en 5 tipos diferentes, de manera que, en función del tipo de objeto un robot tendrá mayor o menor capacidad de carga. El peso de todos los objetos, independientemente de su tipo, es siempre de 45 unidades. Se han realizado experimentos con 3 tipos diferentes de configuraciones o conjuntos de robots. En la configuración 1, todos los robots son iguales y tienen la misma capacidad de carga (3 unidades) para cualquiera de los 5 tipos de tareas. Las capacidades de carga para la configuración 2 y 3 de robots se puede ver en la tabla 4.5, donde, por ejemplo, en la configuración 2 el robot R_3 tiene una capacidad de carga de 10 unidades para las tareas de tipo 3, mientras que para las tareas de tipo 1 la capacidad es de sólo 1 unidad. De esta manera, se analiza el comportamiento de la estrategia de subastas en entornos con robots especializados (con una mayor capacidad de carga) en determinadas tareas.

La figura 4.19 muestra el incremento, en tanto por ciento, del peso total transportado por los robots para diferentes valores de TH, respecto a un sistema con un

-	R_1	R_2	R_3	R_4	R_5
T1	10	1	1	1	1
T2	1	10	1	1	1
T3	1	1	10	1	1
T4	1	1	1	10	1
T5	1	1	1	1	10

(a) Configuración 2.

-	R_1	R_2	R_3	R_4	R_5
T1	2	4	5	6	10
T2	2	12	15	1	9
T3	5	5	5	3	4
T4	8	13	14	2	3
T5	7	6	10	6	5

(b) Configuración 3

Tabla 4.5: Capacidad de carga de los robots para cada tipo de tarea. R1...R5 representan los robots y T1...T5 los 5 tipos de tareas.

umbral igual a 0 ($TH = 0$). Tal como se puede observar, en la mayoría de casos, cuando el valor de TH no es nulo, y por tanto cuando se utiliza la desigualdad 4.1, el peso total transportado se incrementa. Por ejemplo, cuando $TH = 4$ el peso total transportado se incrementa un 26,6 % en la configuración 3 y un 23,3 % en la configuración 2. Los peores resultados se obtienen en la configuración 1, aunque sólo cuando TH es igual a 2 disminuye el peso transportado, en todos los otros casos hay mejora. También se puede observar como existe un valor óptimo para TH, que en experimentos es aproximadamente igual a 4.

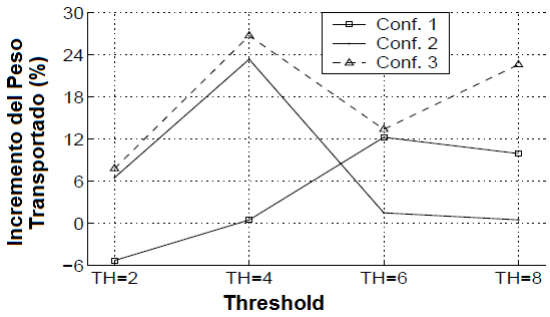


Figura 4.19: Incremento porcentual del peso transportado respecto a un sistema con $TH = 0$, al utilizarse tareas heterogéneas.

4.5.5. Resultados de la estrategia *lane keeping*

En este apartado se explican los experimentos llevados a cabo para validar la estrategia de coordinación de movimiento *lane keeping*. En el primer conjunto de

experimentos los robots deben transportar un único objeto situado en una posición conocida previamente sin que en el entorno haya ningún tipo de obstáculo, a excepción de los propios robots. El objeto tiene un peso infinito y los robots lo han de transportar durante 40.000 unidades de tiempo. Se han utilizado 4 distancias diferentes entre el objeto y el punto de descarga: $D_1 = 280$ unidades, $D_2 = 370$ unidades, $D_3 = 500$ unidades y $D_4 = 600$ unidades. La posición inicial de los robots para cada una de estas distancias siempre es la misma. El número de robots que han de transportar el objeto varía entre 2 y 8, todos ellos con la misma capacidad de carga (2 unidades). En ningún momento se ha limitado el número de robots asignados a la tarea, es decir el método de asignación de tareas utiliza un umbral TH igual a 0.

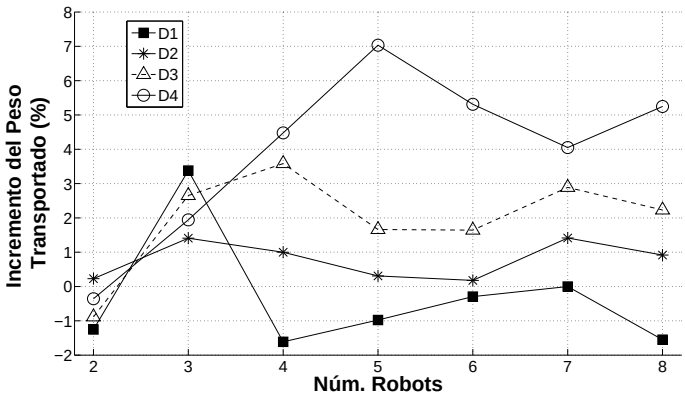


Figura 4.20: Incremento porcentual del peso total transportado utilizando *lane keeping* con respecto a un sistema que no utiliza ningún mecanismo de coordinación.

La figura 4.20 muestra el incremento porcentual en el peso transportado por los robots utilizando la estrategia *lane keeping*, con respecto a un sistema que no utiliza ningún mecanismo de coordinación del movimiento. Como se puede observar, los beneficios de la coordinación del movimiento aumentan a medida que la distancia entre objeto y punto de descarga es mayor. Por ejemplo, cuando la distancia es la mayor de todas (D_4), el beneficio también es el mayor (7 %). En cambio, cuando la distancia es la más pequeña (D_1), los resultados obtenidos son los peores. Este hecho es debido a que la utilización de la estrategia *lane keeping* también implica alargar el camino

entre objeto y punto de descarga, y para distancias cortas no compensa la reducción en la interferencia entre los robots. Si el número de robots es relativamente pequeño, el nivel de interferencia física también será bajo, por lo que tampoco compensará utilizar la estrategia de coordinación del movimiento.

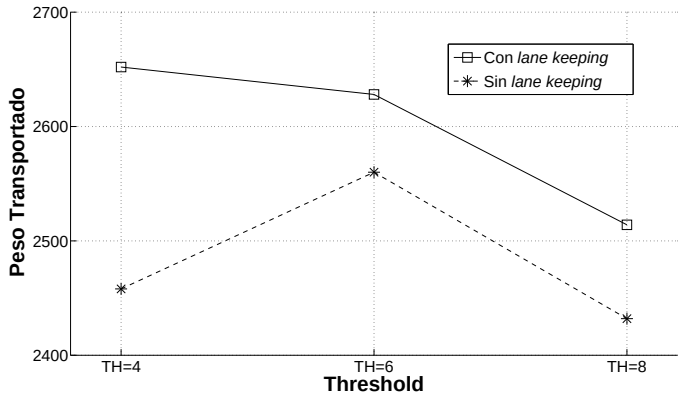


Figura 4.21: Resultados obtenidos utilizando y sin utilizar *lane keeping* con múltiples objetos y recogida continua.

En el siguiente conjunto de pruebas se ha ejecutado la tarea de recogida de objetos junto con la estrategia de subastas con negociación 'líder a robot'. En el entorno hay 10 objetos a recoger, todos ellos con la misma prioridad y un peso igual a 80 unidades, además de 5 robots, también homogéneos con una capacidad de carga de 3 unidades. Como ya se ha visto en la figura 4.20, la estrategia de *lane keeping* mejora los resultados si la distancia entre el objeto y el punto de descarga o el número de robots son lo suficientemente grandes. Por tanto, en los experimentos, los robots únicamente utilizan *lane keeping* si la distancia entre el objeto y el punto de descarga es mayor que 160 unidades y, además, el grupo de trabajo está formado por 3 o más robots. La figura 4.21 muestra el peso total transportado por los robots después de 42.000 unidades de tiempo. Las pruebas se han llevado a cabo tanto utilizando como sin utilizar *lane keeping* con diferentes valores del umbral TH. Como se puede observar, la estrategia *lane keeping* incrementa el peso transportado por los robots en todos los casos, especialmente cuando el valor de TH es bajo y, por tanto, el tamaño de los

grupos de trabajo es mayor.

4.6. Conclusiones

En este capítulo se ha explicado un nuevo método de asignación de tareas para problemas de tipo ST-MR-IA. Este método modifica los actuales de subastas utilizando ideas provenientes de las estrategias basadas en *response threshold*, donde, a diferencia de otros, el número de robots por tarea no está prefijado. Para ello, se utilizan conceptos como el umbral TH o la energía del grupo E_g. En el momento de su publicación éste ha sido uno de los pocos mecanismos de subastas en permitir múltiples robots por tarea y el único que no requiere tener que conocer previamente el número de robots necesarios para ejecutar cada una de ellas. Por lo que nosotros sabemos, actualmente aún se trata del único sistema de subastas que tiene en cuenta, en el proceso de asignación, la interferencia física entre robots y, de esta manera, permite en tiempo de ejecución, determinar el número de robots necesarios para ejecutar cada tarea. Otros mecanismos basados en negociación, como el propuesto por Paquet en [103], permiten no tener que prefijar el número de robots, pero no se pueden ejecutar *on-line*, debido a su elevado tiempo de ejecución, y tampoco tienen en cuenta la interferencia física.

Otra característica que diferencia nuestro método de otros de subasta ya existentes es el mecanismo de negociación entre líderes, que permite implementar la estrategia preemptiva. Mediante la negociación 'líder a líder' éstos negocian entre ellos directamente utilizando la información que tienen sobre los robots de su grupo. Otros mecanismos que también utilizan líderes, como por ejemplo el propuesto por Chaimowicz en [25] o el propuesto por Dias y Stenz en [37], no establecen este tipo de negociación. También cabe destacar, que nuestro método es uno de los pocos que utiliza conceptos de *swarm intelligence* junto con métodos de subastas.

Para validar los nuevos algoritmos, se ha presentado una variante de la tarea clásica de recogida de objetos, en la que múltiples robots pueden colaborar para transportar

un mismo objeto hasta la zona de descarga. Los resultados experimentales han demostrado que el uso del umbral (TH) permite controlar efectivamente el tamaño de las coaliciones y, como consecuencia, mejora los resultados del sistema, al aumentar el peso total transportado por los robots del grupo y disminuir el tiempo de ejecución de las tareas más prioritarias. Como también se ha podido comprobar, la estrategia preemptiva (negociación 'líder a líder') mejora los resultados si hay tareas con prioridades. Así, estos resultados complementan a los obtenidos por Østergaard et al. [102], donde se llevan a cabo tareas de tratamiento de emergencias y para las que estrategias preemptivas mejoran los resultados. Los autores afirman que si se necesitase más tiempo para finalizar las tareas, la estrategia preemptiva no sería conveniente. En el caso presentado en esta tesis, en el que las tareas requieren más tiempo que las propuestas por Østergaard, se ha visto que el mecanismo de intercambio de robots entre tareas es útil siempre que haya prioridades asociadas a los objetos a recoger.

En este capítulo también se ha presentado un nuevo mecanismo de coordinación de movimiento, llamado *lane keeping*, que permite afrontar las características especiales de la tarea de recogida de objetos utilizada y, en general, las de todas las tareas que presenten un alto nivel de interferencia física entre robots que pasan repetidas veces por un mismo camino. El presente estudio ha sido el primero en abordar el efecto de los mecanismos de coordinación en tareas de *foraging* similares a la aquí presentada. Los resultados de las pruebas realizadas para analizar el comportamiento de las estrategias de coordinación del movimiento, muestran que los mecanismos tipo *flocking*, como el de *follow the preceding*, no mejoran los resultados de los tareas de recogida en las que múltiples robots colaboran en la recogida de un mismo objeto. Debido al mal comportamiento de esta estrategia, en esta memoria se ha optado por no mostrar los resultados obtenidos. Estos resultados desmienten las afirmaciones, que sin pruebas experimentales, realizaron otros autores, como Mataric en [98], que mantenían que comportamientos de *flocking* también mejorarían los resultados en otros tipos de tareas de *foraging* diferentes a la clásica. Estos malos resultados se deben a que la mayor parte de la interferencia se produce entre los robots que van en

direcciones opuestas. El nuevo comportamiento de *lane keeping* mejora los resultados respecto a sistemas sin coordinación, especialmente si la distancia entre el objeto y el punto de descarga es grande.

Hasta ahora no se han tenido en cuenta las restricciones temporales de las tareas en el proceso de subasta para problemas ST-MR-IA. En el siguiente capítulo se modificará el sistema de subastas aquí propuesto para incluir estas restricciones. El principal problema que va a aparecer será saber cuánto tiempo va a necesitar un determinado grupo de trabajo para finalizar una tarea y, de esta manera, saber a priori si se van a poder cumplir las restricciones de tiempo. Este tiempo de ejecución dependerá, entre otros factores, del nivel de interferencia física entre los robots del grupo, por tanto, se propondrá un método para predecir esta interferencia y, por extensión, para conocer el tiempo de ejecución de las tareas.

Capítulo 5

Formación de coaliciones en entornos con restricciones temporales

En este capítulo se explican las modificaciones realizadas en el método de subasta presentado en el capítulo anterior para tener en cuenta tareas con restricciones temporales. Uno de los principales inconvenientes para asignar coaliciones de robots a tareas con *deadlines* es la predicción del tiempo de ejecución. Debido a múltiples factores, entre los que se encuentra la interferencia física entre robots, es muy difícil conocer a priori cuánto tiempo va a necesitar un grupo de robots para finalizar una determinada tarea. A tal fin, se ha propuesto un método basado en *Support Vector Regression* (SVR) para modelar y predecir el tiempo de ejecución en función de las características del grupo de trabajo. Se han utilizado dos versiones diferentes del modelo: SVR *off-line* en el que el modelo se obtiene antes del inicio de la misión y SVR *on-line* en la que el modelo se va actualizando durante la ejecución de la tarea. El método *on-line* tiene la ventaja de poder modelizar, tanto las interacciones entre robots del mismo grupo de trabajo, como las producidas con el resto del entorno, ya sean obstáculos o robots de diferentes grupos. También, se ha implementado un nuevo mecanismo de subastas, llamado subasta doble, extensión del propuesto en el capítulo 4, que tiene en cuenta las predicciones del modelo para decidir el número de robots necesarios para finalizar una tarea antes de su *deadline*. Los resultados experimentales

han mostrado la validez del nuevo método de subasta ya que mejora substancialmente los resultados respecto a mecanismos de subasta que no utilizan ningún modelo de interferencia. Además, se mostrará que si se dispone de una adecuada modelización de la interferencia no es necesario incluir mecanismos de monitorización en el progreso de la tarea, ya que se puede predecir el tiempo de ejecución. El método presentado es el primero basado en subastas, que permite ajustar las características del grupo, especialmente su tamaño, en función de la estimación del tiempo de ejecución de la tarea. Asimismo, es el primero que incluye en el proceso de subastas de manera explícita el fenómeno de la interferencia en tareas con restricciones temporales.

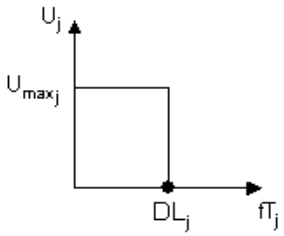
5.1. Definición del problema

La definición del problema a abordar en este capítulo es una extensión de la especificación dada en la sección 4.1. En esta ocasión, las tareas tienen restricciones temporales y funciones de utilidad asociadas a cada una de ellas. De esta manera, si T es el conjunto de tareas a realizar, cada $t_j \in T$ tiene las siguientes características: una carga de trabajo ($taskWorkLoad_j$) asociada, que representa la cantidad de trabajo necesario para ejecutarla; un tiempo límite de ejecución (*deadline*) $DL_j > 0$ y una función de utilidad $U_j(fT_j) \geq 0$, donde fT_j es el tiempo de finalización de t_j. Además, si R es el conjunto de robots, cada $r_i \in R$ tendrá una capacidad de trabajo $workCapacity_{i,j}$, que indicará lo adecuado que es el robot r_i para la tarea t_j. Para simplificar la notación se denotará por g a la coalición de robots R^j asociados a la tarea t_j. Así, cada grupo de robots g encargado de llevar a cabo una tarea t_j tendrá asociada una capacidad de trabajo $groupCapacity_{g,j}$. El valor de $groupCapacity_{g,j}$ dependerá de la capacidad de trabajo de cada uno de los robots de g e indicará lo adecuada que es la coalición para la tarea t_j. El objetivo del algoritmo de asignación de tareas es encontrar una asignación óptima TA^* que maximice la siguiente función objetivo:

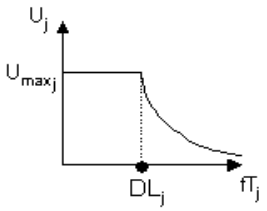
$$U = \sum_{t_j \in T} U_j(fT_j) \tag{5.1}$$

Dado que la utilidad de cada tarea es función del tiempo que se consume en su ejecución, la utilidad total U obtenida con la asignación dependerá de las características de las coaliciones de robots asignadas a cada tarea. Como en los casos analizados en capítulos anteriores, éste continua siendo un problema NP-hard. Se trata en esta ocasión de un caso particular del problema VRPSTW, visto en el capítulo 2.

Cuando el tiempo de ejecución de la tarea t_j excede el *deadline* DL_j, la función U_j empieza a decrecer, es decir la tarea empieza a perder utilidad. La figura 5.1(a) muestra un ejemplo de función de utilidad con *deadline* estricto (*hard deadline*), en la que únicamente se obtendrá utilidad no nula si la tarea finaliza antes de su *deadline*, en cuyo caso su utilidad será igual a U_{max_j}. En cambio, la figura 5.1(b) muestra un ejemplo de una función con un *deadline* no estricto (*soft deadline*), en la que la utilidad de la tarea decrece a partir de un determinado instante de tiempo. Igual que en el caso de la función de utilidad con *deadline* estricto, si la tarea finaliza antes de su tiempo límite, la utilidad obtenida será igual a U_{max_j}. La función de utilidad con *deadline* no estricto se podría ajustar a situaciones tales como el rescate de víctimas en situaciones de catástrofe, en las que a partir de un determinado instante de tiempo, la tasa de mortalidad empieza a crecer.



(a) *Deadline* estricto (*hard deadline*).



(b) *Deadline* no estricto (*soft deadline*).

Figura 5.1: Ejemplos de funciones de utilidad.

Para validar los métodos propuestos en este capítulo se ha utilizado una extensión de la tarea de recogida de objetos del capítulo anterior, permitiendo *deadlines* asociados a cada objeto a recoger. Así, la definición del problema dado anteriormente se ajusta de la siguiente manera a la nueva tarea. Cada robot r_i tiene una capacidad de carga $loadCapacity_i$, que únicamente depende de él y para cada objeto t_j tiene la capacidad de trabajo $workCapacity_{i,j}$. Cada objeto tiene un peso asociado, que se corresponde con la antes mencionada carga de trabajo ($taskWorkLoad_j$), un tiempo límite (DL_j), que empieza a contabilizarse desde el momento en que la tarea aparece en el entorno y una función de utilidad (U_j). La capacidad de trabajo de un robot representa la cantidad de trabajo (peso del objeto) que puede transportar por unidad de tiempo. Los objetos pueden ser transportados por un grupo de robots, g, con una capacidad de trabajo $groupCapacity_{g,j}$ que representa la cantidad de peso del objeto que puede transportar el grupo por unidad de tiempo. Así, el tiempo previsto para finalizar una tarea es igual a:

$$DL_{g,j} = \frac{taskWorkLoad_j}{groupCapacity_{g,j}} \tag{5.2}$$

Dado que en este trabajo se pretende analizar únicamente el proceso de asignación de tareas, se ha supuesto que los robots conocerán la posición de todos los objetos en el entorno, es decir no necesitarán buscarlos. Así pues, habrá un agente central, similar al visto en el capítulo 3, cuya única misión será comunicar a los robots la llegada de nuevas tareas, sin tomar ninguna decisión sobre qué robots las ejecutarán. El resto de características coinciden con las ya explicadas en la sección 4.1 para tareas sin restricciones temporales.

5.2. Modelización de la interferencia

El principal problema para saber qué utilidad (U_j) tendrá la tarea cuando ésta sea ejecutada por una coalición de robots es predecir su tiempo de ejecución, dado por la ecuación 5.2. En esta ecuación, el valor de $taskWorkLoad_j$ puede ser conocido a priori. En cambio, $groupCapacity_{g,j}$ será más difícil de calcular, debido, principalmente, a

la interferencia entre los robots del grupo de trabajo y otros elementos externos del grupo. Se supondrá que el valor real de $groupCapacity_{g,j}$ es igual a:

$$groupCapacity_{g,j} = idealCapacity_{g,j} - I \tag{5.3}$$

donde $idealCapacity_{g,j}$ es la capacidad 'ideal' del grupo g para llevar a cabo la tarea t_j y I es el factor de interferencia. El valor de $idealCapacity_{g,j}$, se puede calcular fácilmente de la siguiente forma:

$$idealCapacity_{g,j} = \sum_{1 \leq i \leq n_g} workCapacity_{i,j} \tag{5.4}$$

donde n_g es el número de robots que forma el grupo de trabajo. Tal como se verá más adelante, el valor de $workCapacity_{i,j}$, se puede calcular de manera relativamente fácil si se conocen las características de cada robot. En cambio, el valor de interferencia I depende de una gran variedad de factores que dificultan su cálculo, en consecuencia, será necesario crear modelos y estimar los valores de sus parámetros a partir de los datos experimentales. A pesar de que este modelo es relativamente simple, resulta muy eficiente, tal como se demostrará en las pruebas experimentales.

5.2.1. Cálculo de la capacidad de trabajo individual

La capacidad de trabajo de un robot, depende del tipo de tarea que se lleve a cabo y de las características del propio robot. En el caso de la misión de transporte de objetos, la capacidad de trabajo será la cantidad de peso que se puede transportar por unidad de tiempo. Por tanto, en condiciones ideales, esto es, asumiendo que no hay ningún obstáculo entre el robot y el punto de descarga, el valor de $workCapacity_{i,j}$ se puede calcular de la siguiente manera: sea V_i la velocidad máxima del robot y d_j la distancia entre el objeto y el punto de descarga. Así, el número de viajes que el robot deberá hacer entre el objeto y el punto de descarga para cargar todo el objeto será igual a $2 \cdot \frac{taskWorkLoad_j}{loadCapacity_i}$. Si se obvia el tiempo de aceleración y de desaceleración, para cada uno de estos viajes el robot necesitará $\frac{d_j}{V_i}$ unidades de tiempo. Sin perdida

de generalidad, se puede considerar que el robot necesita una unidad de tiempo para cargar o descargar cada unidad de peso. Por ejemplo, si un robot ha de cargar 2 unidades de peso, necesitará 2 unidades de tiempo para cargar todo este peso y 2 unidades de tiempo más para descargarlo al llegar al punto de descarga. Así, el robot necesitará $2 \cdot taskWorkLoad_j$ unidades de tiempo para cargar y descargar completamente el objeto. Por tanto, el tiempo total requerido para cargar el objeto será igual a $T_{i,j} = 2 \cdot \frac{taskWorkLoad_j}{loadCapacity_i}(loadCapacity_i + \frac{d_j}{V_i})$ y la capacidad de trabajo $(\frac{taskWorkLoad_j}{T_{i,j}})$ será igual a:

$$workCapacity_{i,j} = \frac{loadCapacity_i \cdot V_i}{2 \cdot (loadCapacity_i \cdot V_i + d_j)} \tag{5.5}$$

Como se puede observar, la capacidad de trabajo de un robot no depende del peso del objeto. Naturalmente, este valor es sólo una estimación, cuyos errores quedarán incluidos como parte del valor de I.

Para obtener el valor de I se han propuesto 2 aproximaciones: cálculo de I fuera del tiempo de ejecución (*off-line*) y cálculo durante el tiempo de ejecución (*on-line*). En el método *off-line* se calculará, en una serie de entornos muy simples el valor de I, que posteriormente se utilizará en la ejecución de la tarea. En cambio, en el método *on-line* los robots iniciarán la ejecución de la tarea con el valor de I calculado de manera *off-line* y en el transcurso de la misma se irá actualizando este valor. Para el cálculo *off-line* sólo se tienen en cuenta datos correspondientes a los robots de un grupo de trabajo, por lo tanto, la interferencia que se modeliza es aquella que se produce como consecuencia de la interacción de los robots que forman una coalición. En cambio, el método *on-line*, al ser actualizado permanentemente durante la ejecución, permite englobar en el modelo los efectos de la interferencia entre robots pertenecientes a diferentes coaliciones e, incluso, la interacción con los obstáculos del entorno, caso de que los hubiera.

5.2.2. Cálculo *off-line* de la interferencia

El modelo *off-line* de la interferencia depende del tipo de tarea a ejecutar. Para obtener el modelo *off-line* de la tarea de transporte de objetos se han llevado a cabo las mismas pruebas que en la sección 4.2 para mostrar los efectos de la interferencia: una serie de robots han de transportar un único objeto de peso infinito. Para llevar a cabo estos experimentos se ha utilizado el simulador RoboCoT y tras 40.000 unidades de tiempo se ha calculado el peso total transportado. Todos los robots tenían la misma capacidad de carga ($loadCapacity_i = 2$), la misma velocidad ($V_i = 3$) y, por tanto, todos tenían la misma capacidad de trabajo. Además, los únicos elementos que había en el entorno eran los robots, el objeto a transportar y la zona de descarga. Se probaron 7 distancias diferentes entre el objeto y el punto de descarga: $D_1 = 140$ unidades, $D_2 = 180$ unidades, $D_3 = 250$ unidades, $D4 = 280$ unidades, $D5 = 330$ unidades, $D_6 = 360$ unidades y $D_7 = 400$ unidades. La figura 5.2 muestra el peso total transportado en el transcurso de estas pruebas, en función del número de robots. Obsérvese que, la relación entre el número de robots y el peso total transportado no es lineal y se encuentra siempre por debajo de la recta de pendiente unidad. De esta manera, la diferencia entre el peso total esperado, calculado como la suma de las capacidades individuales de cada robot (ecuación 5.4), y el peso total realmente transportado, se atribuye a la interferencia I. Se constata experimentalmente que la interferencia merma notablemente la capacidad de trabajo de un grupo de robots. Por ejemplo, cuando la distancia es D_1 y hay 8 robots, $idealCapacity_{g,j} = 8 \cdot workCapacity_{i,j}$, donde $workCapacity_{i,j} = 0,0205479$ para todos los robots del grupo. Por tanto, en 40.000 unidades de tiempo, idealmente, el grupo habría podido transportar $40.000 \cdot idealCapacity_{g,j}$ =6.575,34 unidades de peso. En cambio, las simulaciones realizadas muestran que la cantidad total transportada por todos los robots es de 3.860 unidades de peso, lo que supone que la interferencia ha reducido en un 41,3 % el peso total transportado respecto al valor ideal. Para predecir el valor de I a partir de estos datos, se han usado 2 modelos diferentes: ajuste polinómico y *Support Vector Regression* (SVR).

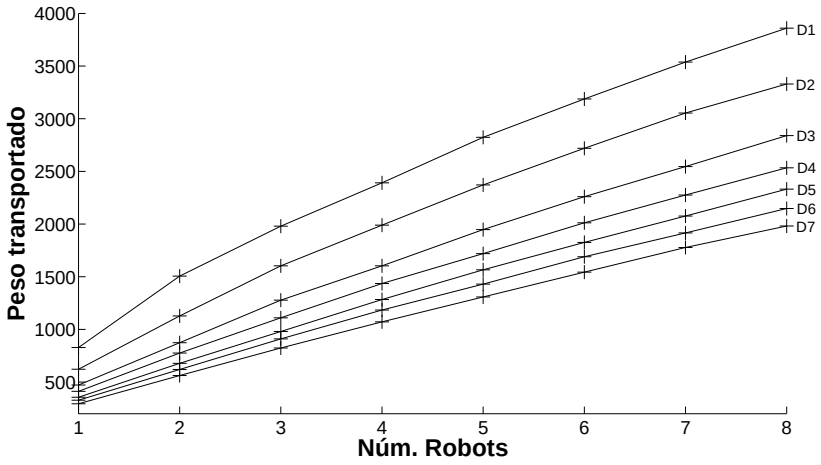


Figura 5.2: Peso total transportado en los experimentos para obtener el modelo *off-line* de la interferencia.

Ajuste polinómico

El modelo polinómico intenta ajustar los valores obtenidos de la interferencia, mediante un polinomio de grado 2, función del número de robots, tal como se puede ver en la siguiente expresión:

$$I(N) = \alpha N^2 + \beta N + \gamma \tag{5.6}$$

donde N es el número de robots del grupo de trabajo. Se ha usado un polinomio de grado 2 debido a su simplicidad y a que se ajusta con un error muy bajo, a los resultados reales. Se han probado otros polinomios de grado mayor, pero en ningún caso los resultados han sido sustancialmente mejores a los obtenidos con polinomios de segundo grado. El ajuste de los parámetros del polinomio $I(N)$ se ha realizado minimizando el error cuadrático medio respecto a los valores experimentales de I. La tabla 5.1 muestra los valores de estos parámetros para cada una de las distancias probadas. Estos parámetros tienen valores muy bajos, debido a que la capacidad de trabajo de cada robot es también muy baja, en comparación con el trabajo total realizado por el colectivo de robots. Además, debe tenerse en cuenta que el error entre los valores experimentales y la función de interferencia $I(N)$ es, en general, muy bajo,

aunque se incrementa a medida que el número de robots decrece. De hecho, $I(N)$ no es aplicable cuando el número de robots es igual a uno, algo lógico ya que con un único robot no hay interferencia física. La figura 5.3 muestra cómo el polinomio $I(N)$ obtenido se ajusta a los resultados experimentales, representados por las cruces que aparecen en el gráfico. Para mejorar la visualización de los resultados de la figura, únicamente se han representado las curvas para las distancias D_1, D_2, D_3 y D_7. En el eje y se presenta el valor de la interferencia multiplicado por 1.000 ($1.000 \cdot I(N)$).

-	α	β	γ
D_1	0,3589	7,5029	-12,3571
D_2	0,3476	2,9640	-4,2857
D_3	0,2432	1,3347	-2,0107
D_4	0,2006	1,0620	-1,6321
D_5	0,1619	0,4737	-0,6071
D_6	0,1494	0,3734	-0,5071
D_7	0,1071	0,4050	-0,5000

Tabla 5.1: Parámetros de la función de interferencia, $1.000 \cdot I(N)$, ajustada mediante un polinomio de grado 2.

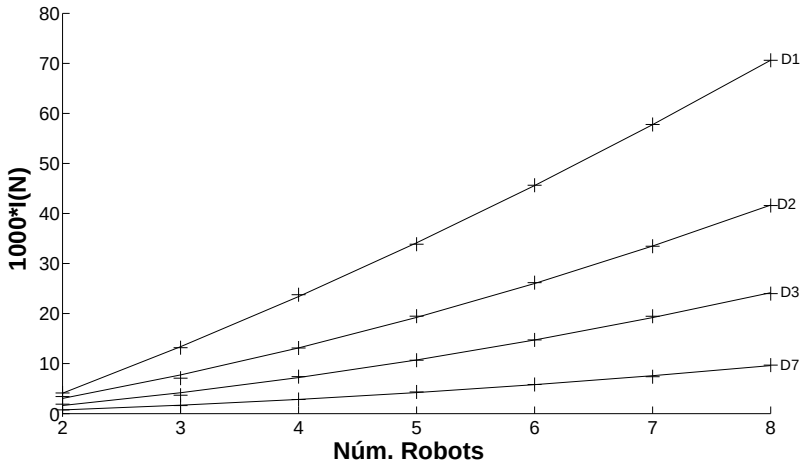


Figura 5.3: Ajuste de $I(N)$ mediante un polinomio de grado 2 para diferentes valores de distancia.

En este trabajo también se han probado formas alternativas de ajustar el valor de I a partir de la capacidad de trabajo que tiene el grupo de robots g para llevar a cabo la tarea t_j. En particular, una función probada ha sido la que se muestra a continuación:

$$groupCapacity_{g,j} = \sum_{1 \leq i \leq N} workCapacity_{i,j}(1 - e^{I(N)}) \tag{5.7}$$

donde $I(N)$ es un polinomio de grado 2, similar al expuesto en la ecuación 5.6, para el que se deben ajustar los parámetros α, β y γ. Los experimentos realizados mostraron que los resultados obtenidos a partir de la estimación de la ecuación 5.7 eran similares a los obtenidos utilizando el polinomio 5.6.

Support Vector Regression (SVR)

El último método estudiado para modelar la interferencia es el llamado *Support Vector Regression* (SVR) [11, 121, 120], basado en los sistemas de *Support Vector Machines* (SVM) [124]. El objetivo de este método es encontrar una función, $f(x)$ tan simple como sea posible, que se ajuste a un conjunto de datos de entrenamiento o muestras (x_i, y_i), $i = 1..l$, donde $x_i \in \Re^n$ representan los datos de entrada de $f(x)$ y $y_i \in \Re$ son los resultados esperados. Se pretende así mapear los valores de x en un espacio de características F de mayor dimensión mediante una función ϕ y resolver el problema como si fuese una regresión lineal sobre F. Así, $f(x)$ tendrá la siguiente forma:

$$f(x) = W^T \phi(x) + b \tag{5.8}$$

donde W y b son los coeficientes a estimar. Para ello, se ha de solucionar el siguiente problema de optimización:

$$
\begin{aligned}
& min\frac{1}{2}W^TW + C\sum_{1\leq i\leq l}(\xi_i + \xi_i^*) \\
& \text{sujeto a: } y_i - (W^T\phi(x_i) + b) \leq \varepsilon + \xi_i \\
& \qquad (W^T\phi(x_i) + b) - y_i \leq \varepsilon + \xi_i^* \\
& \qquad \xi_i, \xi_i^* \geq 0 \text{ para i=1...l}
\end{aligned}
\tag{5.9}
$$

donde la constante C es un valor positivo que representa el peso que se da al error respecto a lo 'plana' que queremos que sea la función $f(x)$, ε representa el error tolerado en $f(x)$ respecto a los datos de muestra, es decir, todo error inferior a ε se considerará como un valor correcto. Los valores ξ_i y ξ_i^* representan la diferencia entre el valor de la muestra con respecto a $f(x) + \varepsilon$ y $f(x) - \varepsilon$. La figura 5.4 presenta un ejemplo del significado de los valores ε, ξ_i y ξ_i^*, en la que los puntos de la gráfica simbolizan las muestras (y_i). Todos aquellos valores que se encuentren dentro del intervalo $[f(x) - \varepsilon, f(x) + \varepsilon]$ no se consideran errores.

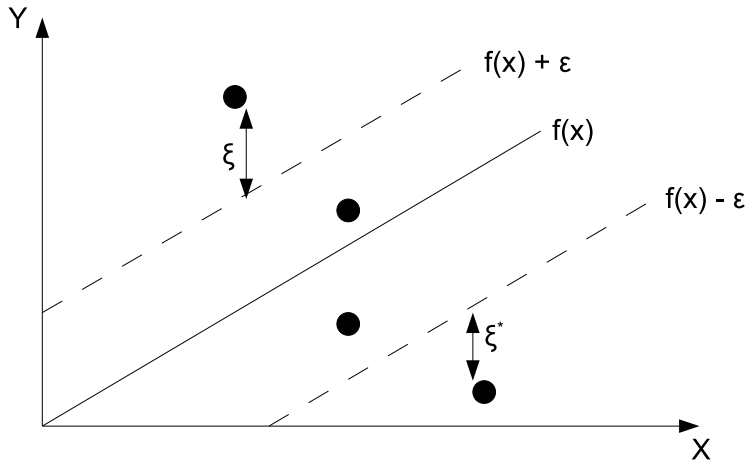


Figura 5.4: Ejemplo bidimensional del uso del parámetro ε y de ξ_i, ξ_i^*.

A partir de este conjunto de datos, y usando multiplicadores de Lagrange, se puede calcular la correspondiente función lagrangiana (L) y obtener su problema de optimización dual:

$$max_{\alpha,\alpha^*} -\frac{1}{2}\sum_{i=1}^{l}\sum_{i=1}^{l}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\phi(x_i)^T\phi(x_j)-\varepsilon\sum_{i=1}^{l}(\alpha_i+\alpha_i^*)+\sum_{i=1}^{l}y_i(\alpha_i-\alpha_i^*)$$
$$\text{sujeto a:}\sum_{i=1}^{l}(\alpha_i-\alpha_i^*)=0,\quad \alpha,\alpha_i\in[0,C] \tag{5.10}$$

donde α_i y α_i^* son multiplicadores de Lagrange. El producto $\phi(x_i)^T\phi(x_j)$ se puede calcular por medio de una función de *kernel* K, de manera que $K(x_i,x_j) = \phi(x_i)^T\phi(x_j)$. Existen múltiples funciones que cumplen las condiciones para ser *kernel*: *kernels* lineales, polinómicos, etc. De entre todos ellos el más usado y el que se utilizará en los experimentos es el *Radial Basis Function* (RBF) que tiene la siguiente expresión:

$$K(x_i,x_j)=e^{-\gamma\|x_i-x_j\|^2} \tag{5.11}$$

donde γ es el parámetro de este *kernel*. A partir de la función lagrangiana podemos saber que $w=\sum_{i=1}^{l}(\alpha_i+\alpha_i^*)\phi(x_i)$ ya que $\frac{\partial L}{\partial w}=0$. Entonces, la ecuación 5.8 se puede reescribir de la siguiente forma:

$$f(x)=\sum_{1\le i\le l}(\alpha_i-\alpha_i^*)K(x,x_i)+b \tag{5.12}$$

Finalmente, cabe destacar que se puede definir una función que penalize el error que aparece en la función 5.9. En las pruebas llevadas a cabo, se ha optado por una función de perdida insensible a ε (ε-*insensitive loss function*), que tiene la siguiente forma:

$$c(\xi_i)=\begin{cases}|\xi_i|-\varepsilon & \text{para } |\xi_i|>\varepsilon\\ 0 & \text{en caso contrario}\end{cases} \tag{5.13}$$

La información obtenida de los experimentos realizados *off-line* son los datos de entrenamiento del método SVR. Los vectores x_i representan las características de la

coalición de robots y tienen 2 componentes: el número de robots del grupo de trabajo, y la distancia entre el objeto y el punto de descarga. Los valores esperados de $f(x)$ (y_i) son los valores de la interferencia I. Estos valores se calculan como la diferencia entre la capacidad ideal de trabajo del grupo ($idealCapacity_{g,j}$), obtenida a partir de la ecuación 5.4, y la capacidad real, obtenida dividendo la carga de trabajo (peso del objeto) por el tiempo de ejecución de los experimentos. Para crear el modelo se han usado un total de 80 muestras (vectores (x_i, y_i), con $l = 80$), de manera que cada vez que un robot quiera conocer la interferencia estimada de un grupo de trabajo, y por tanto el tiempo previsto de finalización de una tarea, deberá aplicar la fórmula 5.12.

Para implementar la versión $off-line$ del método SVR se ha utilizado la librería *libsvm* desarrollada por Chang y Lin [26]. Esta librería usa una mejora del algoritmo *Sequential Minimal Optimization* (SMO) [43] para solucionar el problema de optimización de la función 5.10. Los valores de los parámetros del método SVR utilizados han sido los siguiente: $C = 300$, $\varepsilon = 0,1$ y $\gamma = 2,0$. Estos valores han sido obtenidos a partir de los datos experimentales. Otros autores, como Jones en [78], también han utilizado la librería *libsvm* para asignar tareas durante simulaciones de la liga *Robocup-Rescue*. En este caso, sólo se permitía un robot por tarea sin modelizar la interferencia y suponiendo tiempos de ejecución conocidos.

Tal como expone Juutilainen en [80], el ajuste mediante polinomios o con la función exponencial, no produce predicciones correctas cuando los datos están alejados de los valores de entrenamiento utilizados para obtener los parámetros. Por tanto, si durante la ejecución de la tarea la distancia o el número de robots no coincide con los utilizados para obtener las curvas, el error en la predicción será muy grande. Tampoco es adecuado utilizar este tipo de regresiones cuando la dimensión de los datos de entrada es muy elevada. Por ahora, únicamente se tiene en cuenta la distancia y el número de robots, pero en trabajos futuros se tendrán en cuenta otros parámetros, por ejemplo: capacidades de cada robot o las características del entorno, entre otras. Otros métodos, como el de regresión lineal local [17] de Bontempi et al. tampoco se

consideran convenientes cuando hay muchas dimensiones. Finalmente, según Juutilainen et al. [80], las redes neuronales muestran un comportamiento similar al de los métodos SVR siempre que el ruido en los datos de entrada sea bajo, en caso contrario los mecanismos SVR presentan mejores resultados. Es por todos estos motivos que, el método SVR ha sido el finalmente elegido para estimar el valor de la interferencia en nuestro sistema.

5.2.3. Cálculo *on-line* de la interferencia

Para el método de cálculo *on-line*, el modelo de interferencia no se fija previamente, sino que los robots pueden ir modificando y adaptando el modelo a las características específicas del entorno. Para implementar esta estrategia se ha optado por la versión *on-line* del método SVR, donde nuevos datos (vectores) pueden añadirse o eliminarse en tiempo de ejecución. Ma et al. propusieron en [97] un método *on-line* de SVR, en el cual cada vez que un nuevo dato es añadido o borrado, se modifican los multiplicadores de lagrange de la función SVR (ecuación 5.12) para asegurarse de que todo el sistema sigue verificando las condiciones de Karush-Kuhn-Tucker (KKT), condiciones que aseguran que la solución sigue siendo válida. Así, si las condiciones KKT se siguen cumpliendo, se puede asegurar que el error de la estimación dada por el modelo SVR sigue siendo mínimo. Las pruebas llevadas a cabo demostraron que este método es más rápido que la versión *off-line* implementada en *libsim* cuando los datos llegan uno a uno. Para nuestro sistema se ha utilizado la implementación del método de Ma realizada por Parrella en [105], con la misma función de *kernel* y parámetros C, ε y γ que en la versión *off-line*. La complejidad de este algoritmo es, en el peor de los casos, $O(kernel \cdot n_v^3) = O(n_v^3)$, donde n_v es el número de vectores en el modelo SVR y $kernel$ es el tiempo necesario para calcular el valor de la función de *kernel* 5.11.

En el cálculo *on-line* de la interferencia, los robots inician su ejecución utilizando el modelo obtenido de manera *off-line*. Cada vez que una tarea finaliza, el robot líder de la misma calcula el valor de la interferencia, de igual forma a como se obtenía de manera *off-line*, crea una nueva muestra (x_i, y_i) con estos valores y modifica el modelo

siguiendo el algoritmo *on-line* de SVR antes mencionado. Finalmente, comunica el nuevo vector (x_i, y_i) al resto de robots para crear un modelo global de interferencia, común a todo el sistema. El número de muestras n_v del modelo va incrementándose a lo largo del transcurso de las tareas, incrementando también el tiempo de ejecución del algoritmo *on-line* de SVR. Para evitar tener un excesivo número de muestras, se ha limitado su número a 120, ya que los datos experimentales mostraron que esta cantidad permite mantener la eficiencia del sistema sin requerir tiempos de ejecución excesivos. Si al añadir nuevas muestras su número supera las 120, se eliminarán siempre las más antiguas.

5.3. Proceso de subasta doble

En esta sección se explica el proceso de asignación de tareas propuesto para la creación de los grupos de trabajo. Para ello, se ha modificado el método de subasta explicado en el capítulo 4 para tener en cuenta las utilidades de las tareas y sobre todo sus restricciones temporales. El nuevo mecanismo de subastas, llamado método de subasta doble, divide el método anterior en dos fases: subasta de tareas y subasta de robots. A partir de ahora, y para distinguirlo de la subasta doble, al proceso de subastas visto en el capítulo 4 para tareas sin restricciones temporales se le denominará método de subasta simple.

5.3.1. Subasta de tareas

Cuando un robot no líder detecta una tarea no asignada, intenta convertirse en su líder. El proceso de creación de líder es exactamente igual al propuesto en las subastas simples, es decir sigue el algoritmo 9. Si el robot se convierte en líder, intenta crear un grupo de trabajo, mediante un proceso de subasta. El proceso es similar al visto para la subasta simple en la negociación 'líder a robot'. Sin embargo ahora, los robots pujan por la tarea utilizando su valor de capacidad de trabajo, $workCapacity_{i,j}$, obtenido a partir de la ecuación 5.5. El líder seleccionará los mejores robots (los que tengan

una mayor capacidad de trabajo), hasta detectar que el grupo es capaz de cumplir el *deadline* de la tarea, esto es, hasta que se cumpla la siguiente condición:

$$DL_{g,j} \leq DL_j \tag{5.14}$$

A tal fin, para cada puja se estimará la capacidad de trabajo del grupo creado hasta el momento aplicando la ecuación 5.3. El líder no incluirá ningún robot seleccionado hasta que no reciba su confirmación, mediante el proceso de subasta de robots, que se explicará en la siguiente sección.

5.3.2. Subasta de robots

Una vez el líder ha seleccionado los robots con los que pretende formar su grupo de trabajo, se inicia la subasta de robots. Para reclutar a los robots seleccionados en la subasta de tareas, el líder pujará por ellos enviándoles un mensaje ($AWARD$) con la utilidad estimada de la tarea (U_j), que depende del tiempo de ejecución previsto (véase ecuación 5.2). Cuando un robot recibe una puja de un líder, espera un determinado período de tiempo ($TIME\_BID\_ACCEPTED$) para recibir más pujas de otros líderes. Una vez transcurrido este tiempo, el robot selecciona el grupo de trabajo con la mayor utilidad de todas las solicitudes recibidas y envía un mensaje de confirmación ($ROBOT\_ALIVE$) al líder correspondiente. Al resto de líderes no seleccionados se les envía un mensaje de rechazo ($REFUSE$). Si el líder, tras haber transcurrido un determinado período de tiempo, no recibe ningún mensaje de confirmación de un robot o si recibe un mensaje de rechazo, eliminará al robot remitente de la lista de pre-seleccionados iniciando, si es necesario, una nueva subasta de tareas. En caso contrario, el robot será incluido en la lista definitiva de miembros del grupo. Además, durante toda la ejecución de la tarea, el líder irá calculando el valor de la capacidad de trabajo del grupo sin tener en cuenta la interferencia y el número de robots que lo forman. Al finalizar la tarea, se calcularán los valores medios de estas cantidades para crear el vector que modificará el modelo SVR y que posteriormente será enviado al resto de robots.

El proceso que ejecuta el líder, tanto en las subastas de tareas como en las de robots, se puede ver en el algoritmo 12. Como se puede observar, la principal diferencia respecto a la subasta simple, es que ahora se utiliza la expresión 5.2 para obtener el valor de $D_{g,j}$ (ver línea 15) y que los mensajes de confirmación a los robots no se envían mientras no se haya calculado calculado el valor de la utilidad de la tarea. La complejidad algorítmica de este proceso se puede calcular de la siguiente manera: sea n el número de pujas realizadas, la complejidad del algoritmo de ordenación de pujas (línea 9), utilizando los métodos *marge sort* o *binary tree sort*, es $O(n \cdot log(n))$. El cálculo del valor estimado de la interferencia con el modelo SVR realizado en la línea 15, tiene una complejidad de $O(n_v^3)$. Por tanto, la complejidad del bucle para seleccionar los robots (líneas 13-17) será $O(n \cdot n_v^3)$. Finalmente, el último bucle (líneas 18-20) tiene una complejidad $O(n)$. Por tanto la complejidad algorítmica de todo el proceso es igual a $O(n \cdot log(n) + n \cdot n_v^3 + n) \subset O(max(n \cdot log(n), n \cdot n_v^3))$. Lógicamente, esta complejidad es mayor que la calculada para la subasta simple, ya que se ha de utilizar el modelo SVR para tomar las decisiones. En lo que respecta a las comunicaciones, el líder ha de enviar un mensaje solicitando pujas (línea 1) más, otros n para confirmar los robots seleccionados. Por tanto, el número de mensajes a enviar por el líder será $O(2 \cdot n) = O(n)$. A estos se le ha de añadir, que en el caso de utilizar el modelo *on-line* del método SVR, el líder enviará un mensaje a todos los robots con la nueva muestra, resultante de la ejecución de la tarea.

El algoritmo 13 muestra el proceso que lleva a cabo cada uno de los robots, una vez recibida una petición ($AWARD$) por parte de un líder. Como se puede observar, para obtener el mejor líder (línea 9) se requiere una complejidad algorítmica de $O(n)$, dónde n es el número de mensajes recibidos de los líderes. El envío de los mensajes de rechazo (líneas 11-13) tiene una complejidad también igual a $O(n)$, por tanto la complejidad de todo el algoritmo será $O(2 \cdot n) = O(n)$. Este proceso es más complejo que el ejecutado por los robots en la subasta simple, donde la complejidad era $O(1)$. La complejidad de las comunicaciones también ha aumentado respecto a la subasta simple, ahora los robots han de enviar un mensaje de aceptación al líder seleccionado,

Algoritmo 12 Algoritmo de subasta ejecutado en el líder para la tarea t_j

Enviar solicitud de pujas para la tarea t_j
$B_r \leftarrow \emptyset$
$tiempo_{inicio} \leftarrow tiempo$
mientras $(tiempo - tiempo_{inicio}) \leq TIME\_AUCTION$ **hacer**
  si puja b_i recibida del robot r_i **entonces**
    $B_r \leftarrow B_r \bigcup b_i$
  fin si
fin mientras
Ordenar B_r de mayor a menor capacidad de trabajo
$DL_{g,j} \leftarrow \infty$
$k \leftarrow 0$
$R_p \leftarrow \emptyset$
mientras $i \leq |B_r|$ **y** $DL_{g,j} \geq DL_j$ **hacer**
  $R_p \leftarrow R_p \bigcup$ (robot que ha realizado la puja $B_r[k]$)
  Actualizar $DL_{g,j}$ utilizando las fórmulas 5.2 y 5.12
  $i \leftarrow i + 1$
fin mientras
para todo $r \in R_p$ **hacer**
  Enviar solicitud a robot r (mensaje $AWARD$) con la utilidad de la tarea $U_j(DL_{g,j})$
fin para

Algoritmo 13 Algoritmo de subasta de robots ejecutado en cada uno de los robots no líderes

si mensaje b_0 tipo $AWARD$ recibido de un líder **entonces**
  $B_l \leftarrow b_0$
  $tiempo_{inicio} \leftarrow tiempo$
  mientras $(timpo - tiempo_{inicio}) \leq TIME\_BID\_ACCEPTED$ **hacer**
    si mensaje b_i tipo $AWARD$ recibido de un líder **entonces**
      $B_l \leftarrow B_l \bigcup b_i$
    fin si
  fin mientras
  $l_{mejor} \leftarrow$ mejor líder de B_l
  Enviar mensaje ROBOT_ALIVE a l_{mejor}
  para todo líder l_i en B_l y $l_i \neq l_{mejor}$ **hacer**
    enviar mensaje $REFUSE$ a l_i
  fin para
fin si

más un mensaje de rechazo a todos aquellos líderes no seleccionados, por tanto la complejidad es $O(n)$. Este mensaje de rechazo no sería estrictamente necesario, ya que utilizando el mecanismo de detección de robots obsoletos en el grupo, explicado en la sección 4.3.6, se podría saber cuando un robot no forma parte de un grupo. A pesar de ello, las pruebas experimentales mostraron que el mensaje de rechazo reducía los tiempos de espera.

Método	Computacional	Comunicaciones
Subasta doble	A: $O(max(n \cdot log(n), n \cdot n_v^3))$ B: $O(n)$	A: $O(m)$ B: $O(m)$
[115] (Service et al. 2010)	$O(n^{2j} \cdot m)$	-
[128] (Vig et al. 2006)	$O(n^k m)$	-
Dynamic Role [25] (Chaimowicz et al. 2002)	A: $O(n)$ B: $O(1)$	A: $O(n)$ B: $O(1)$

Tabla 5.2: Complejidad computacional y del sistema de comunicaciones de diversos mecanismos de asignación de tareas. Donde m es el número de tareas y n el número de robots.

La tabla 5.2 muestra la complejidad computacional y del mecanismo de comunicaciones del algoritmo de subasta doble, así como la complejidad de algunos de los principales métodos de creación de coaliciones vistos en el capítulo 2. En esta tabla, m es el número de tareas, n el número de robots, A es la complejidad en el subastador, B es la complejidad de los robots que envían las pujas, k es el número máximo de robots en una coalición para el algoritmo de Vig et al. [128] y j es el número de grupos homogéneos en los que los robots se pueden agrupar en el algoritmo de Service et al. [115]. Los algoritmos de Service y Vig no están basados en subastas, por tanto no es necesario especificar la complejidad del subastador ni del pujador, ni tampoco mostrar la complejidad del mecanismo de comunicaciones. También se ha de notar que, como se dijo en el capítulo 2, para el problema propuesto en este trabajo, puede suceder que k y j sean igual al número de robots. Así pues, únicamente el método

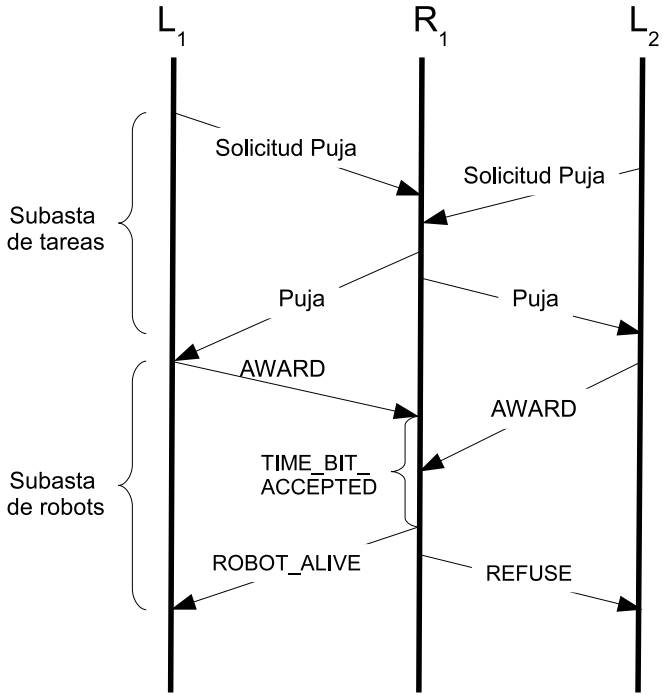


Figura 5.5: Esquema de los mensajes enviados durante el proceso de subasta doble.

de *dynamic role* propuesto por Chaimowicz tiene una complejidad algo inferior a la del algoritmo de subasta doble, aunque el tamaño de las coaliciones en el método de Chaimowicz está prefijado y no permite llevar a cabo tareas con *deadlines*.

La figura 5.5 muestra un ejemplo del conjunto de mensajes enviados entre los robots en el proceso de subasta doble. En este caso el robot R_1 ha enviado una puja a 2 líderes diferentes (L_1 y L_2). Durante el proceso de subasta de tareas, R_1 decide entrar a formar parte del grupo de trabajo de L_1.

La figura 5.6 muestra el autómata con las transiciones entre los diferentes estados durante una subasta doble. El significado de cada uno de estos estados es el siguiente:

- **Libre**: el robot no tiene ninguna tarea asignada.
- **Subasta Miembro**: el robot ha recibido ofertas de varios líderes y está ejecutando el algoritmo 13.
- **Miembro**: forma parte de un grupo de trabajo.

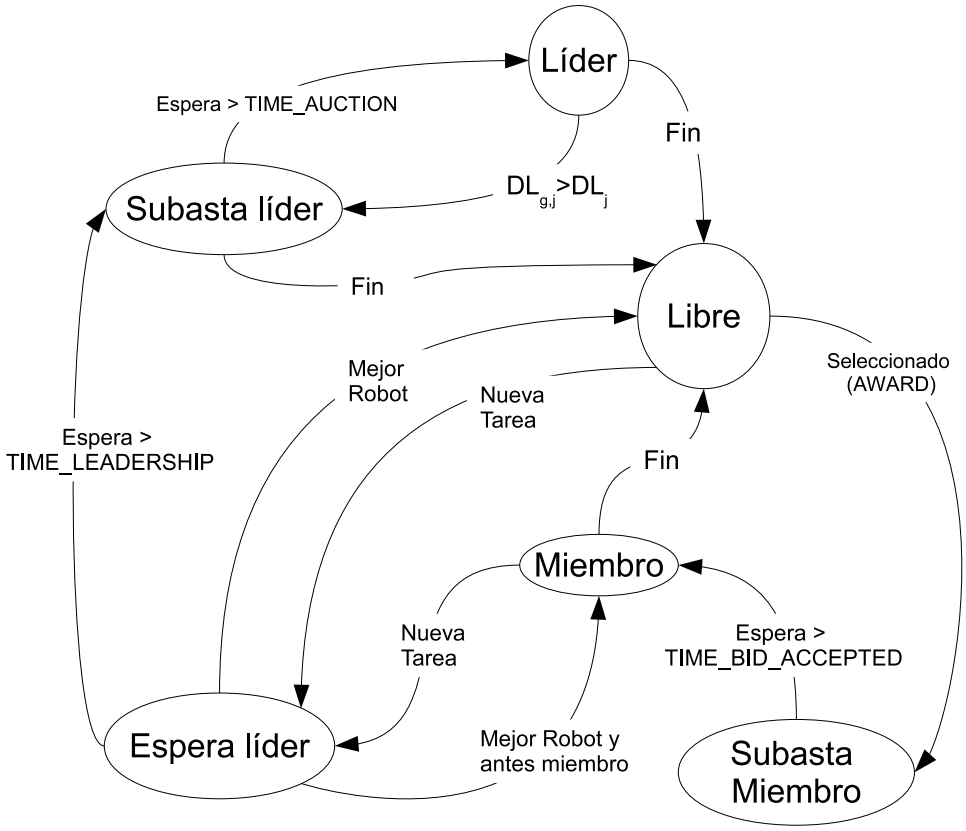


Figura 5.6: Autómata con las transiciones entre estados durante el método de subasta doble.

- **Espera Líder**: espera obtener el liderazgo de una tarea.
- **Subasta líder**: es líder y está ejecutando la subasta de tareas.
- **Líder**: es líder de la tarea y no está ejecutando en este momento ningún proceso de subasta.

Además de las transiciones que aparecen en el autómata, desde cualquier estado se puede pasar al estado *libre* en el momento de finalización de la tarea (transición Fin). Para facilitar la visualización del autómata, tampoco aparecen en la figura las transiciones debidas a la duplicidad de líderes o a robots obsoletos, representadas en el autómata de la figura 4.8, correspondiente a tareas sin restricciones temporales.

Durante todo este proceso también se han utilizado los mecanismos destinados a evitar la duplicación de líder y robots obsoletos, así como los procesos de agregación y segregación ya utilizados en las tareas sin restricciones temporales. Debido a que el objetivo del sistema es mostrar como se puede predecir el tiempo de ejecución utilizando mecanismos de subasta, no se ha implementado la estrategia preemptiva (negociación 'líder a líder') ni los mecanismos de substitución de líder.

5.3.3. Proceso de monitorización

Algunas de las soluciones propuestas anteriormente usan un agente central cuya función es monitorizar continuamente la evolución de la tarea. Así pues, durante la ejecución de una tarea, su líder podrá recibir periódicamente de dicho agente información sobre la carga de trabajo que queda por realizar, es decir se podrá monitorizar la evolución de la tarea. El sistema presentado en este capítulo puede adaptarse de forma sencilla a la presencia de un agente monitor. En este caso, el líder utiliza la información del estado de su tarea para predecir el valor actual de la carga de trabajo usando la siguiente ecuación:

$$WL(t) = WL(t_m) - groupCapacity_{g,j} \cdot (t - t_m) \tag{5.15}$$

donde t_m es el instante de tiempo en que el líder recibió la información sobre el progreso de la tarea y $WL(t)$ es la carga de trabajo estimada en el instante de tiempo t, $t \geq t_m$. De esta manera, cuando el líder detecta que, para el valor estimado de la carga de trabajo, el grupo de trabajo no puede finalizar la tarea antes de su *deadline*, se inicia un nuevo proceso de subasta doble con el fin de captar más capacidad de trabajo.

El proceso de monitorización puede ser una tarea muy compleja que requiere de sofisticados recursos sensoriales y de comunicación. Se han realizado pruebas utilizando tanto monitorización continua como sin usar ningún tipo de monitorización del progreso de la tarea. En el proceso de monitorización continua, el líder conoce en cada momento el valor exacto de la carga de trabajo restante, mientras que en el

sistema sin monitorización, el líder sólo conoce el valor de la carga de trabajo al inicio de la ejecución de la tarea, utilizando la ecuación 5.15 con un valor constante de t_m durante todo el proceso. Como se demostrará mediante las pruebas experimentales, el modelo de interferencia que se ha utilizado permite predecir de tal manera el tiempo de ejecución de la tarea que hace prácticamente innecesaria la monitorización.

5.4. Resultados experimentales

En esta sección se expondrán los resultados de las pruebas experimentales llevadas a cabo para analizar el impacto que tienen sobre la utilidad total del sistema los diferentes conceptos presentados en este capítulo: subasta doble, modelo de la interferencia (*on-line* SVR y *off-line* SVR), proceso de monitorización y tipo de función de utilidad asociada a cada tarea (U_j). En todas las pruebas se ha utilizado el simulador RoboCoT, usado en los experimentos sin restricciones temporales del capítulo 4. Los robots han de ejecutar la tarea de recogida de objetos ya explicada, de manera que, para preservar las condiciones iniciales del entorno, cada vez que un objeto es totalmente recogido, inmediatamente aparece otro con las mismas características situado en una posición aleatoria. El tiempo límite de un objeto (DL_j) empieza a contar a partir del momento en el que éste aparece en el entorno. Además, todos los objetos deben ser transportados totalmente aunque su *deadline* haya expirado y su utilidad sea nula. En cuanto al algoritmo de asignación de tarea, se han llevado a cabo 3 tipos de subastas:

- Subasta doble: utiliza el mecanismo de subasta doble explicado en la sección 5.3 y que incluye la subasta de tareas y de robots.
- Subasta única (*Single Auction*-SA): en esta estrategia únicamente se utiliza la subasta de tareas, de manera que un robot libre entra inmediatamente a formar parte de un grupo de trabajo tras recibir la confirmación del líder.
- Estrategia voraz: en estos experimentos, los líderes intentarán crear grupos de

trabajo con tantos robots como sea posible, sin tener en cuenta el valor del *deadline* de la tarea ni ninguna otra característica. Esta estrategia es similar a las subastas SA, pero ahora el líder envía un mensaje de aceptación a todos los robots de los que ha recibido una puja, independientemente del valor de éstas.

En función del mecanismo de monitorización y de si se utiliza o no modelo de interferencia (*off-line* o *on-line*), los 3 mecanismos de asignación descritos anteriormente pueden usarse conjuntamente con las siguientes estrategias:

- M_I: utiliza monitorización continua junto con el modelo de interferencia propuesto.
- NM_I: utiliza el modelo de interferencia, pero sin ningún tipo de monitorización del progreso de la tarea.
- M_NI: utiliza la estrategia de monitorización continua pero sin ningún modelo para la interferencia. Esto supone que el tiempo estimado de ejecución se calculará usando únicamente la capacidad ideal del grupo de trabajo.
- NM_NI: no utiliza ningún tipo de monitorización ni modelo de interferencia, esto es, únicamente se utiliza la ecuación 5.4.

La tabla 5.3 muestra un resumen de los diferentes tipos de experimentos llevados a cabo, como resultado de la combinación del método de subasta, del mecanismo de monitorización y del modelo de interferencia usado. La estrategia voraz sólo tiene sentido si se ejecuta conjuntamente con la estrategia NM_NI, ya que no implementa ningún mecanismo para limitar el tamaño del grupo de trabajo. A partir de ahora cada vez que se haga referencia a la asignación voraz se entenderá que se utiliza junto con la estrategia NM_NI.

5.4.1. Pruebas con el modelo *off-line* de SVR

En esta sección se explican los experimentos realizados para validar los métodos de subasta única y subasta doble con el modelo *off-line* de SVR. Para ello, se han

-	M_I	NM_I	M_NI	NM_NI
Subasta doble	X	X	X	X
SA	X	X	X	X
Voraz				X

Tabla 5.3: Resumen de los tipos de experimentos llevados a cabo.

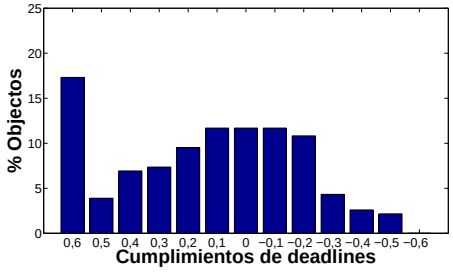
llevado a cabo experimentos con funciones de utilidad homogéneas y heterogéneas. En el primer caso todas las tareas tienen la misma utilidad, mientras que en el segundo el valor de la función de utilidad varía de unas tareas a otras.

Función de utilidad homogénea con *deadline* estricto

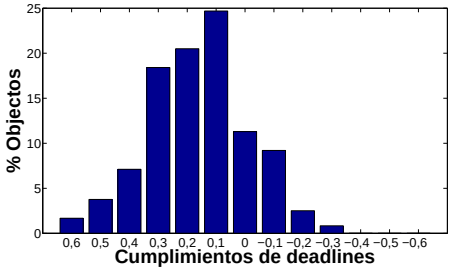
A continuación se explican los experimentos llevados a cabo utilizando tareas que tienen asignada una función de utilidad con *deadline* estricto y utilidad homogénea, es decir, todas las tareas tienen la misma utilidad máxima. De esta manera, el objetivo del sistema es finalizar tantas tareas como sea posible antes de su *deadline* y, en caso de no lograrlo, finalizarlas en el menor tiempo posible. En el proceso de subasta doble únicamente se tiene en cuenta la utilidad de la tarea en la etapa de subasta de robots, ya que en la subasta de tareas el objetivo es captar suficientes robots para cumplir el *deadline*. Si todas las tareas tienen la misma utilidad, en la subasta de robots todos los líderes que puedan cumplir el *deadline* de su tarea también tendrán la misma utilidad y no se podrá discriminar entre ellos. Así pues, para este tipo de tareas el algoritmo de subasta doble se comporta igual que el de subasta única y, por tanto, en este apartado únicamente se analiza el método de subasta única. La simplicidad de la función de utilidad facilita el análisis del modelo *off-line* de SVR, pudiendo comprobar su validez independientemente de si se utiliza subasta única o doble.

El modelo de interferencia utilizado en las pruebas descritas en este apartado ha sido siempre el método SVR *off-line*. En todos los experimentos se han utilizado 10 robots y 3 objetos a recoger. Todos los robots tienen las mismas características: capacidad de carga igual a 2 unidades de peso y velocidad máxima de 3 unidades de distancia por unidad de tiempo. Todas las tareas tienen un peso igual a 40 unidades, el

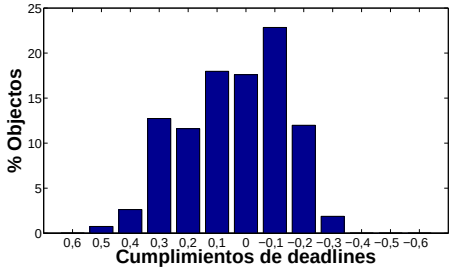
mismo tiempo límite de ejecución y la misma función de utilidad. Los tiempos límite probados ha sido los siguientes: 1.200 unidades de tiempo y 900 unidades de tiempo. Los robots llevan a cabo la misión durante un tiempo total de 30.000 unidades de tiempo, trascurrido el cual se calcula el tiempo invertido en el transporte de todos los objetos, tanto si se ha verificado su *deadline* como si no. Cada uno de los experimentos se ha repetido 4 veces.



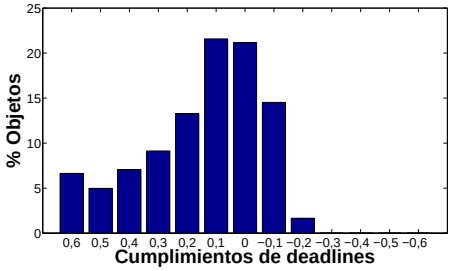
(a) Resultados con la estrategia voraz.



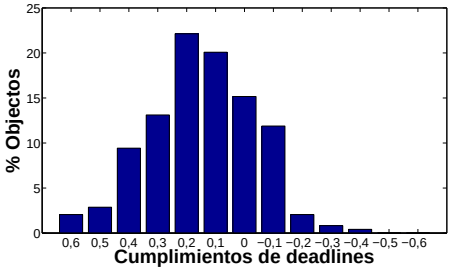
(b) Resultados con la estrategia NM_NI.



(c) Resultados con la estrategia NM_I.



(d) Resultados con la estrategia M_NI.



(e) Resultados con la estrategia M_I.

Figura 5.7: Porcentaje de cumplimientos de *deadlines* con un tiempo límite de 900.

La figura 5.7 muestra los resultados obtenidos cuando el *deadline* de las tareas es igual a 900 unidades utilizando las siguientes estrategias: voraz, NM_NI, NM_I,

M_NI y M_I. La barra etiquetada con 0,6 corresponde al porcentaje de tareas cuyo tiempo de ejecución excede en más de un 60 % su *deadline*, es decir cuyo tiempo de ejecución es mayor a $1,6 \cdot DL_j$. La barra etiquetada con 0,5 representa el porcentaje de tareas cuyo tiempo de ejecución es mayor o igual a un 50 % del *deadline* e inferior a un 60 % del mismo, y así sucesivamente. Los valores negativos, representan aquellas tareas que han cumplido su *deadline*. Por ejemplo, la barra etiquetada con -0,2 representa el porcentaje de tareas cuyo tiempo de ejecución es entre un 10 % y un 20 % inferior a su tiempo límite. La tabla 5.4 muestra el porcentaje de tareas finalizadas antes de su *deadline* (columna $t \leq DL$) y el porcentaje de tareas que cumplen su *deadline* o, exceden en un máximo del 10 % de este tiempo para finalizar (columna $t < (DL + 10\,\%)$).

-	$t \leq DL$	$t < (DL + 10\,\%)$
Voraz	30,87	42,57
NM_NI	12,56	23,86
NM_I	36,67	54,27
M_NI	16,16	37,36
M_I	15,18	30,38

Tabla 5.4: Porcentaje de tareas que ejecutadas antes de su *deadline* (columna $t \leq DL$, siendo t el tiempo de ejecución) o que exceden menos de un 10 % de este tiempo máximo, con un *deadline* de 900 unidades.

Como se puede observar en los gráficos, para la estrategia voraz la correlación entre el tiempo de ejecución de la tarea y su *deadline* es muy baja. En cambio, en los experimentos realizados con el mecanismo de subastas única, las tareas claramente tienden a agruparse alrededor del tiempo máximo de ejecución. Por ejemplo, en la estrategia voraz, el 17,31 % de las tareas necesitan más del 60 % del tiempo de *deadline* para finalizar, porcentaje que en el peor de los casos con el mecanismo de subasta única (estrategia M_NI) es del 6,64 %. Por otra parte, tal como muestra la tabla 5.4, en algunos casos el porcentaje de tareas que cumplen estrictamente su tiempo de ejecución es mayor en la estrategia voraz que en las estrategias que utilizan subastas. Como se verá en la sección 5.4.1, este problema se soluciona gracias al uso de subasta

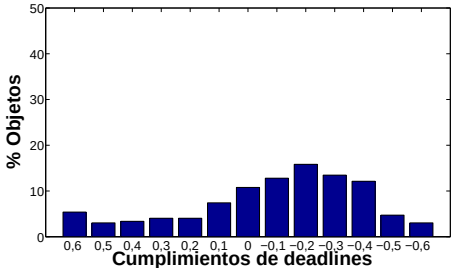
doble.

El uso de un modelo de interferencia también mejora los resultados, especialmente cuando se usa sin monitorización (estrategia NM_I), en este caso, el porcentaje de tareas que cumplen su *deadline* es el mayor de todos. Finalmente, los resultados obtenidos con las estrategia M_NI y M_I muestran como una monitorización continua puede incluso empeorar los resultados. Este efecto es debido a que la interferencia puede ralentizar momentáneamente el progreso de la tarea. En este caso, el tiempo necesario para finalizar la tarea es estimado erróneamente. Si el líder utiliza una estimación errónea, entonces la condición 5.14 no se cumplirá y acaparará un número demasiado grande de robots. Por tanto, gracias al uso del modelo de interferencia *off-line* se han mejorado los resultados del sistema de subasta.

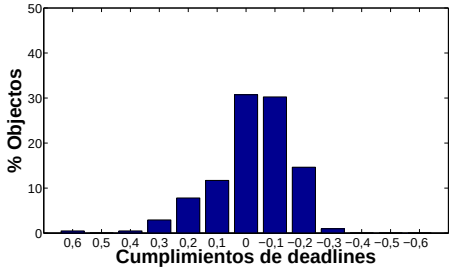
La figura 5.8 muestra los resultados obtenidos utilizando tareas con un *deadline* igual a 1.200 unidades de tiempo y las estrategias: voraz, NM_NI, NM_I, M_NI y M_I. La tabla 5.5 muestra el porcentaje de tareas que cumplen su *deadline* o que necesitan hasta un 10 % más de este tiempo para finalizar. Al igual que pasaba en los experimentos con *deadlines* iguales a 900 unidades, el mecanismo de subastas permite que el tiempo de finalización de las tareas se concentre más alrededor del tiempo máximo de ejecución. Además, excepto para la estrategia NM_NI, en todos los casos se incrementa el número de tareas que han cumplido su *deadline* respecto a la estrategia voraz. Al contrario de lo que pasaba con un valor de *deadline* menor, ahora utilizar monitorización junto con el modelo de interferencia (M_I) sí que mejora los resultados respecto al método NM_I, al contrario de lo que sucedía con un *deadline* de 900 unidades. Esto se debe a que es menos probable que la condición 5.14 deje de cumplirse debido a estimaciones erróneas en el tiempo de ejecución.

Función de utilidad heterogénea

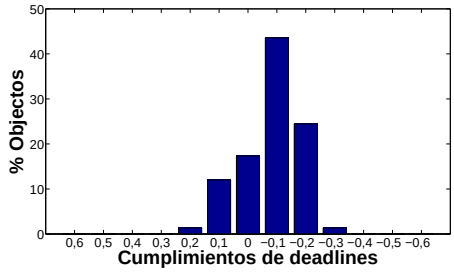
A continuación se exponen los resultados obtenidos utilizando funciones de utilidad con *deadlines* estrictos y no estrictos. La utilidad máxima de las tareas es heterogénea, esto es, puede variar de una tarea a otra. Además, ahora el objetivo del



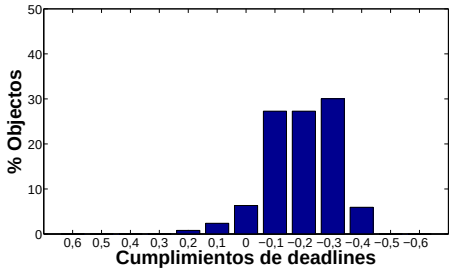
(a) Resultados con la estrategia voraz.



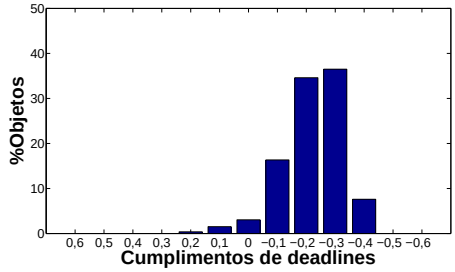
(b) Resultados con la estrategia NM_NI.



(c) Resultados con la estrategia NM_I.



(d) Resultados con la estrategia M_NI.



(e) Resultados con la estrategia M_I.

Figura 5.8: Porcentaje de cumplimientos de *deadlines* con un tiempo límite de 1.200.

-	$t \leq DL$	$t < (DL + 10\,\%)$
Voraz	61,94	72,71
NM_NI	45,78	76,48
NM_I	69,33	86,63
M_NI	90,53	96,85
M_I	95,00	98,04

Tabla 5.5: Porcentaje de tareas ejecutadas antes de su *deadline* (columna $t \leq DL$, siendo t el tiempo de ejecución) o que exceden menos de un 10 % de este tiempo máximo, con un tiempo *deadline* de 1.200 unidades.

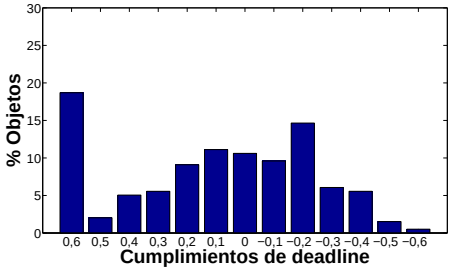
sistema es maximizar la utilidad total obtenida, dada por la ecuación 5.1. Mediante este tipo de tareas se demuestra que la subasta doble incrementa la utilidad total del sistema con respecto a la estrategia voraz o la subasta única.

La tarea a realizar es similar a la ya utilizada para la estrategia de subasta única: en el entorno habrá 10 robots y 3 objetos a recoger todos ellos con el mismo peso (40 unidades de peso). Cuando uno de los 3 objetos del entorno es totalmente transportado inmediatamente aparece en una posición aleatoria otro con las mismas características. La misión finalizará una vez los robots hayan transportado 400 objetos. Limitar el número de objetos, y no el tiempo de ejecución, como sucedía en en los experimentos anteriores, permite comparar la utilidad total obtenida en las diferentes estrategias. Todas las tareas tienen un *deadline* igual a 900 unidades de tiempo, pero su utilidad máxima (U_{max_i}) es diferente para cada una de ellas. En los experimentos se han usado dos tipos diferentes de funciones de utilidad: funciones de utilidad con *deadlines* no estrictos y funciones de utilidad con *deadline* estrictos. En la función con *deadlines* estrictos la utilidad de la tarea t_j es igual a U_{max_j} si se ha podido ejecutar antes de su límite temporal y 0 en caso contrario. La figura 5.1(a) es un ejemplo de este tipo de funciones. En el caso de funciones de *deadlines* no estrictos, la utilidad de una tarea siempre es mayor que 0, tal como muestra la figura 5.1(b). En los experimentos presentados en esta sección se ha usado la siguiente función:

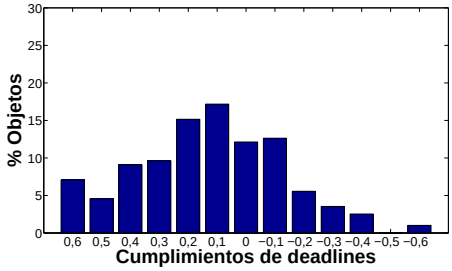
$$U_j = \begin{cases} U_{max_j} & \text{if } fT_j < DL_j \\ U_{max_j} \frac{0{,}07 \cdot DL_j}{(fT_j - DL_j) + 0{,}07 \cdot DL_j} & \text{en caso contrario} \end{cases} \tag{5.16}$$

donde fT_j es el tiempo que se ha necesitado para finalizar la tarea. En todos los casos siempre se ha utilizado la versión *off-line* del método SVR.

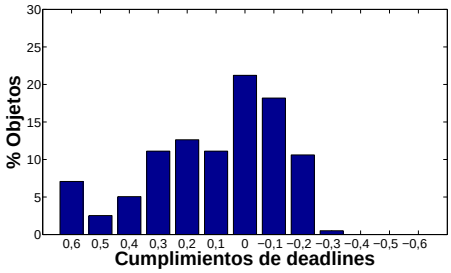
La figura 5.9 muestra el porcentaje de tareas finalizadas en función del tiempo requerido utilizando el método de subasta doble y con la función de utilidad dada por la expresión 5.16. El significado de cada una de las barras es el mismo ya explicado para la figura 5.7 en entornos con subasta única y funciones de utilidad homogéneas. Como se puede observar, la estrategia voraz sigue sin mostrar ninguna correlación entre el



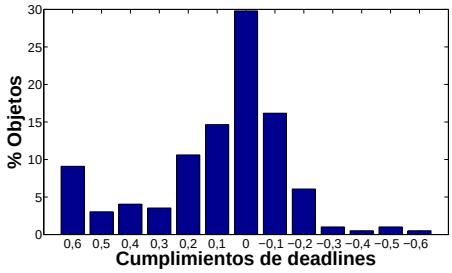
(a) Resultados con la estrategia voraz.



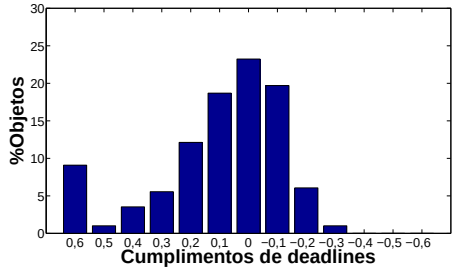
(b) Resultados con la estrategia NM_NI.



(c) Resultados con la estrategia NM_I.



(d) Resultados con la estrategia M_NI.



(e) Resultados con la estrategia M_I.

Figura 5.9: Porcentaje de cumplimientos de *deadlines* utilizando la función de utilidad 5.16 y subastas dobles.

tiempo de ejecución de la tarea y su *deadline*. En cambio, el método de subasta doble agrupa las tareas alrededor de su *deadline*, principalmente si se utiliza monitorización o el modelo de interferencia. La tabla 5.6 muestra el porcentaje de tareas finalizadas antes de su *deadline* (columna $t \leq DL$) o, que exceden un máximo del 10 % de este tiempo para finalizar. Al igual que en los resultados obtenidos con subasta única, nuevamente la estrategia voraz y la NM_I son las que tienen un mayor porcentaje de tareas que cumple estrictamente su *deadline*. A pesar de ello, se ha de recordar que el objetivo del sistema es maximizar la utilidad y no únicamente incrementar el número tareas finalizadas antes de su tiempo límite. Tal como se demuestra a continuación, el sistema de subasta doble junto con el modelo de interferencia incrementa claramente la utilidad del sistema. Los resultados obtenidos en entornos con tareas que tienen *deadlines* estrictos tienen una forma similar a los ya mostrados con el método de subasta única.

-	$t \leq DL$	$t < (DL + 10\,\%)$
Voraz	37,88	48,48
NM_NI	25,30	37,40
NM_I	29,30	50,50
M_NI	25,25	55,05
M_I	26,76	50,00

Tabla 5.6: Porcentaje de tareas ejecutadas antes de su *deadline* (columna $t \leq DL$, siendo t el tiempo de ejecución) o que exceden menos de un 10 % de este tiempo máximo, utilizando subastas dobles y la función de utilidad dada por la expresión 5.16.

La figura 5.10 muestra la utilidad total U del sistema, obtenida a partir de la ecuación 5.1, al usar la función de utilidad con *deadlines* no estrictos (función 5.16). Tanto la estrategia voraz como la SA con NM_I se han incluido en el gráfico 5.10 como elementos de comparación con las estrategias de subasta doble. Como se puede observar, las estrategias que utilizan subasta doble junto con el modelo de interferencia (M_I y NM_I) son las que presentan los mejores resultados, produciendo un incremento en la utilidad de aproximadamente un 40 % respecto al sistema voraz. En cambio, los métodos que no tienen en cuenta la interferencia (NM_NI y M_NI)

muestran resultados similares a los del sistema voraz. De esta manera, se demuestra que el modelo de interferencia permite incrementar la utilidad total del sistema. La diferencia entre M_I y NM_I también es muy pequeña, tal como ya sucedía en las anteriores pruebas con el método de subasta única. Por tanto, se puede afirmar que el modelo de interferencia predice correctamente la evolución de la tarea, sin que sea necesario utilizar mecanismos de monitorización. Finalmente, el método de subasta única muestra en esta ocasión valores similares a los de estrategia voraz, por tanto, la subasta de robots ha sido capaz de mejorar claramente los resultados.

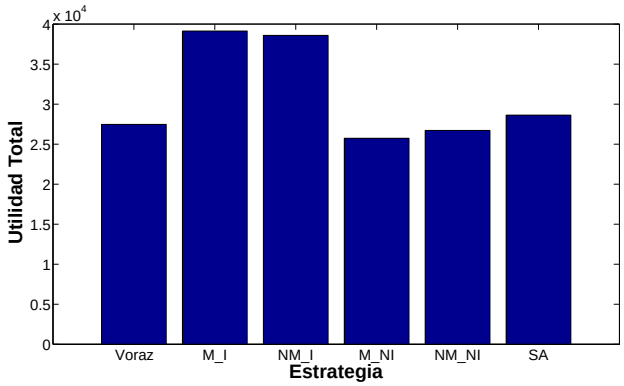


Figura 5.10: Utilidad total obtenida utilizando la función de *deadlines* no estrictos (función 5.16).

La figura 5.11 muestra la utilidad total obtenida cuando se utiliza una función de utilidad con *deadline* estricto. La estrategia NM_I es la que presenta los mejores resultados, incrementando en aproximadamente un 43 % la utilidad total respecto a la estrategia voraz. De igual manera a como ya sucedía con *deadlines* no estrictos, los mecanismos de subasta doble que utilizan el modelo de interferencia mejoran los resultados respecto a aquellos que no lo utilizan. La estrategia de monitorización con interferencia (M_I) muestra aproximadamente un 11 % menos de utilidad total que el método NM_I, esta reducción se debe a que, al igual que en pruebas con la estrategia de subasta única, el progreso de una tarea puede que no sea uniforme a lo largo de su ejecución. Finalmente, como sucedía en la sección anterior, la utilidad de la estrategia de subasta única es menor que la obtenida con la estrategia voraz. A

pesar de ello, y tal como ya se pudo ver en la figura 5.7(c), la estrategia de subasta única reduce notablemente el número de tareas que requieren gran cantidad de tiempo para finalizar.

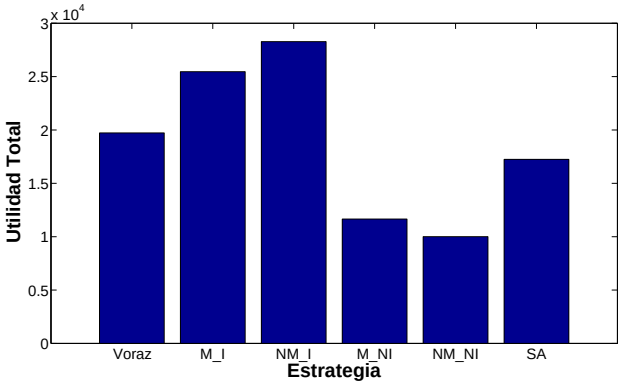


Figura 5.11: Utilidad total obtenida utilizando la función de *deadlines* estrictos.

5.4.2. Pruebas con el modelo *on-line* de SVR

A continuación se analizan los resultados del método de aprendizaje SVR *on-line* para predecir el tiempo de ejecución. Debido a restricciones computacionales se ha decidido reducir tanto el número de robots como el de tareas respecto a las pruebas realizadas hasta ahora. Así, en todas las pruebas realizadas se han utilizado 7 robots homogéneos que deben transportar un total de 200 objetos, de los que siempre hay 3 simultáneamente en el entorno. Para ello, se ha usando la función de utilidad 5.16 y la estrategia NM_I. Se han llevado a cabo dos tipos diferentes de experimentos:

- Experimentos sin errores: en este caso los robots conocen sin error alguno todos los parámetros del entorno y de sí mismos: velocidad, distancia al punto de descarga, etc.

- Experimentos con errores: en estas pruebas los robots desconocen su velocidad máxima. La velocidad máxima real del robot es de 3,0 unidades de distancia/unidad de tiempo, pero creen que es igual a 0,8. Esta velocidad errónea es

la utilizada para calcular el valor de la capacidad de trabajo $workCapacity_{i,j}$ en la ecuación 5.5. De esta manera, se podrá analizar lo que sucede al haber errores en el modelo cinemático del robot.

La tabla 5.7 muestra los resultados con y sin errores utilizando los algoritmos *on-line* y *off-line* del método SVR. Cuando no hay errores en el sistema, el método *on-line* incrementa en un 9,7 % la utilidad respecto a la estrategia *off-line*. Por su parte, cuando hay errores en el modelo del robot, la estrategia *on-line* incrementa en un 62,5 % la utilidad total obtenida respecto al cálculo *off-line* de la interferencia. Así, se puede afirmar que el sistema propuesto es capaz de modelar la interferencia física entre robots, además de otras fuentes de error, como errores en la cinemática de los robots, incrementando la robustez de la solución propuesta.

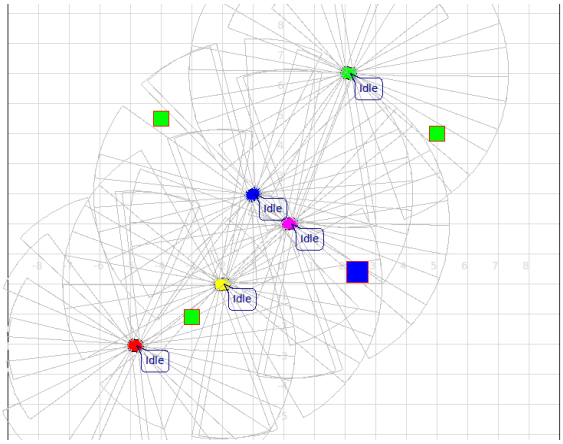
	off-line SVR	*on-line* SVR
Sin errores	10.802	11.851
Con errores	6.290	10.226

Tabla 5.7: Utilidad total alcanzada en los experimentos con los algoritmos *on-line* y *off-line* del método SVR.

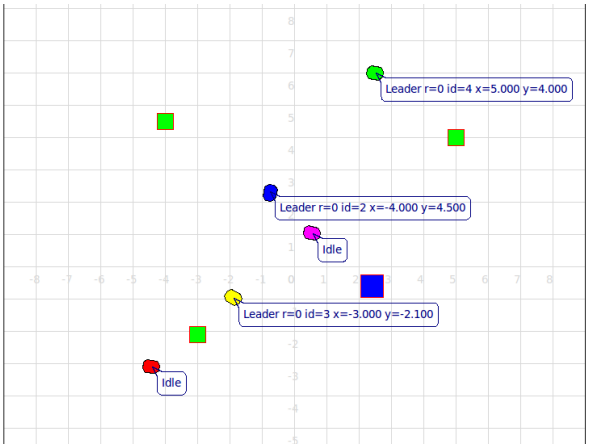
5.5. Pruebas con robots reales

En esta sección se explican las pruebas llevadas a cabo tanto con el simulador Player/Stage como con robots reales, usando el algoritmo de subastas doble junto con el modelo de interferencia *on-line*.

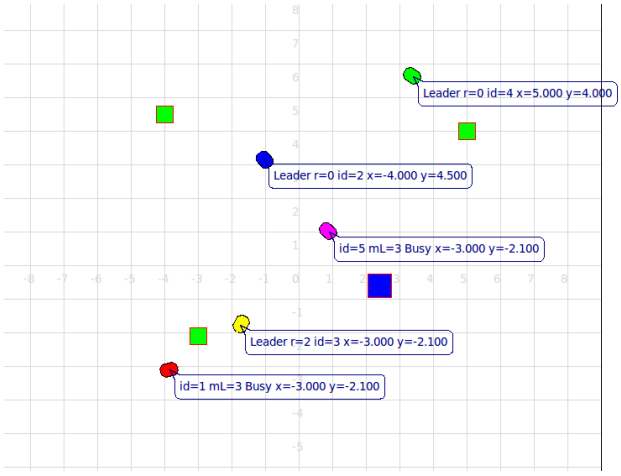
Como paso previo a las pruebas sobre robots reales se han llevado a cabo una serie de simulaciones en el simulador Player/Stage utilizando 5 robots robots Pioneer 3DX, con las mismas características que los reales. El código utilizado para el control de los robots (mecanismos de evitación de obstáculos, arquitectura de los robots, algoritmos de asignación de tareas, etc.) es exactamente el mismo que posteriormente se utilizará sobre los robots físicos. Desde el punto de vista de las aplicaciones programadas es indiferente actuar sobre robots reales o sobre el simulador Player/Stage. Además,



(a) Instante inicial. Todos los robots están libres.



(b) Finalización del proceso se selección de líder.



(c) Finalización del proceso de creación de grupos.

Figura 5.12: Ejemplo del proceso de creación de grupos de trabajo.

existe un programa, llamado *monitor*, que se encarga de informar a todos los robots de las nuevas tareas que aparecen en el entorno y de indicar cuando éstas ya han sido transportadas totalmente.

La figura 5.12 muestra el proceso de creación de los grupos de trabajo con 5 robots y tres objetos a recoger. Los cuadrados de menor tamaño representan los objetos a recoger y el de mayor dimensión el punto de descarga. Junto a cada robot aparece la información de su estado: si no tiene ninguna tarea asignada (*idle*), si se trata de un líder (*Leader*), la posición del objeto que recoge el robot (valores x e y), además del identificador del robot (*id*) y, si es necesario, el identificador de su líder (*ml*). Si el robot es un líder, el valor r indica el número de robots que tiene a su cargo, de manera que si r es igual a 0, el grupo de trabajo únicamente estará formado por el líder. En la figura 5.12(a) se puede ver el estado inicial del sistema y el alcance de los sensores de ultrasonidos que tienen los robots. La 5.12(b) muestra los tres líderes, uno por cada tarea, aún sin ningún otro robot asignado. Finalmente, en la figura 5.12(c) se puede observar la asignación definitiva de los robots a las tareas.

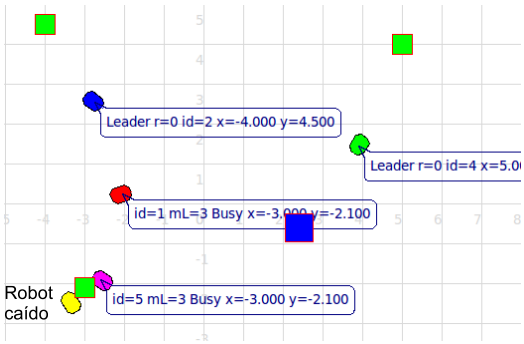
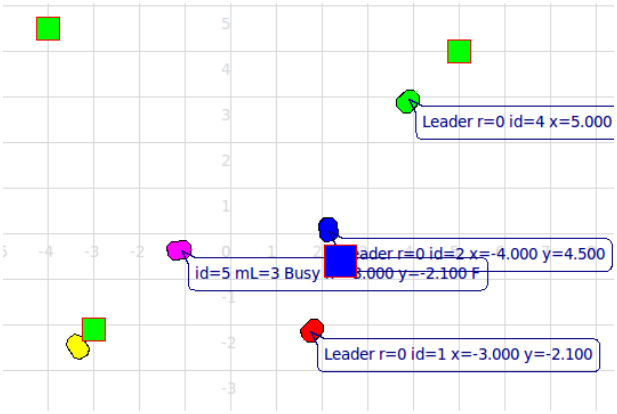
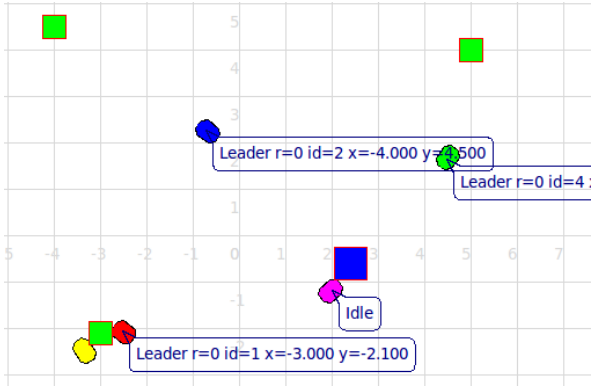


Figura 5.13: El robot 3, antes líder de una tarea, deja de funcionar.

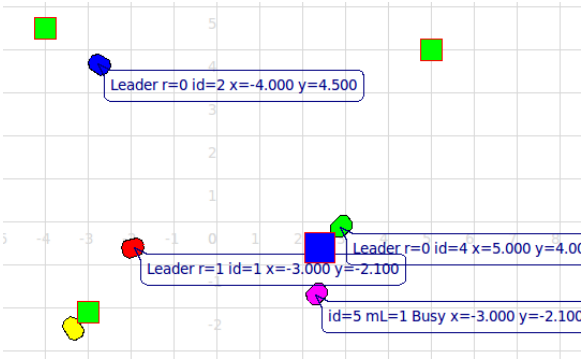
Las figura 5.13 y 5.14 muestran cómo, durante la misma misión explicada para la figura 5.12, el sistema se recupera después de que un robot líder deje de funcionar. En la figura 5.13 se puede ver como el robot 3, que antes era líder de la tarea situada en la posición (-3.0, -2.1), deja de funcionar. La figura 5.14(a) muestra como, 30s después de la caída del líder, el robot 1 substituye al antiguo líder de la tarea. Nueve segundos más tarde, el robot 5, que antes formaba parte del grupo de líder que ha dejado de



(a) El robot 1 substituye al líder caído (30s después de la caída del líder).

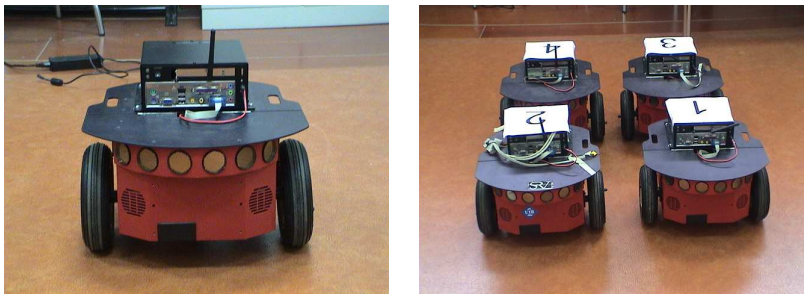


(b) Se libera el robot del líder caído. (39s después de la caída del líder).



(c) Recuperación total del sistema (43s tras la caída del líder).

Figura 5.14: Recuperación del sistema ante la caída de un líder.



(a) Pioneer-3DX. (b) Imagen con los 4 robots utilizados.

Figura 5.15: Imágenes de los robots Pioneer-3DX.

funcionar, queda libre, tal como se puede observar en la figura 5.14(b). El sistema se recupera totalmente cuando el nuevo líder (robot 1) selecciona como miembro de su grupo de trabajo al robot que había quedado libre (robot 5). De esta manera, tras haber transcurrido 43s. de la caída del líder, el sistema se recupera totalmente.

Una vez verificados los distintos procesos diseñados sobre el simulador Player/Stage, se ha procedido a realizar diversas pruebas sobre robots reales. En estas pruebas se han utilizado hasta 4 robots Pioneer-3DX, iguales al que se puede ver en la figura 5.15, con una velocidad máxima de 0,25m/s. Para la detección de obstáculos se utilizan 2 anillos de 8 sensores de ultrasonidos cada uno. El posicionamiento de los robots se realiza mediante odometría. Cada uno de estos robots está equipado con una placa basada en un procesador Eden a 600MHz y 512MB de RAM, además de con una tarjeta *wireless* para comunicación. En el anexo B se puede encontrar una explicación más detallada sobre el hardware utilizado. Las pruebas han sido realizadas en un entorno con unas dimensiones de 4m de ancho por 9m de largo, en el que los robots debían recoger 2 objetos y un tercero aparecía en el transcurso de la ejecución. La figura 5.16 muestra la disposición inicial de los 4 robots en una de las pruebas. Los puntos marcados como *O3*, *O2* y *O1* representan los objetos a recoger, mientras que el punto *D* (en el centro de la imagen) representa la zona de descarga. Para considerar que un robot ha llegado a un objeto y por tanto lo puede recoger, éste debe situarse a menos de 0,4m del mismo. Para que se pueda realizar la descarga el vehículo ha de

estar a menos de 0,7m del punto *D*.

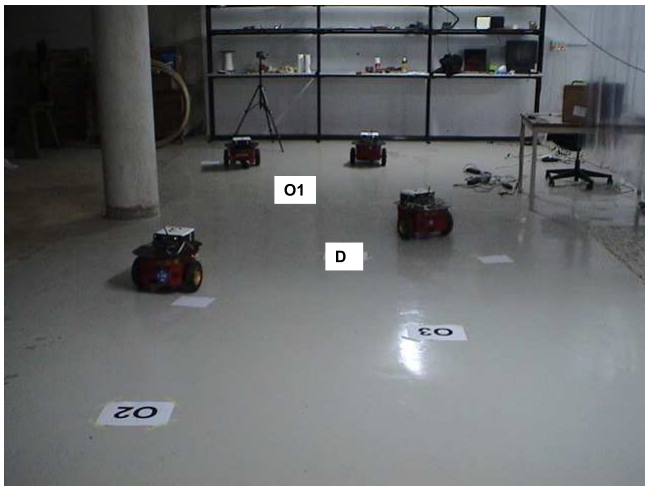


Figura 5.16: Situación inicial en los experimentos con robots reales.

La figura 5.17 muestra dos imágenes tomadas durante la ejecución de uno de los experimentos. En la figura 5.17(a) se ve como el robot 1 se acerca al punto de descarga (*D*) mientras que el robot 2 se aproxima al objeto 1. En la figura 5.17(b) se puede observar como los robos 4 y 3 colaboran en la recogida del objeto *O2*. Por otra parte, la figura 5.18 muestra como el sistema se recupera tras el fallo de uno de sus líderes. En la imagen 5.18(a) se puede ver la situación inicial del sistema, en la que 3 robots están asignados a la tarea *O2* y uno sólo a la tarea *O1*. La figura 5.18(b) muestra como el líder de *O1* falla (es retirado del entorno). Finalmente, en 5.18(c) se ve que se ha asignado un nuevo líder a *O1* sólo 15s después de haber retirado al anterior líder. El comportamiento del sistema en todas las pruebas realizadas con robots reales ha sido en todo momento el esperado e igual al ya probado sobre los simuladores Player/Stage y RoboCoT. De esta manera, se demuestra la validez de nuestro sistema en un entorno real, en el que además la capacidad de procesamiento de los robots es relativamente baja. En el anexo B se encuentra una definición detallada del entorno utilizado, tanto hardware como software, así como el valor del resto de parámetros del sistema.

(a) El robot 1, se acerca al punto de descarga y el robot 2 al objeto 1.

(b) Los robots 4 y 3 se encaminan hacia el objeto 2.

Figura 5.17: Ejemplo de ejecución sobre un entorno real.

(a) Situación inicial.

(b) El líder de $O1$ es retirado del entorno (falla).

(c) Recuperación del sistema.

Figura 5.18: Ejemplo del fallo de un líder en un entorno real.

5.6. Conclusiones

En este capítulo se ha ampliado el método de subastas para su uso en tareas con restricciones temporales y diferentes tipos de funciones de utilidad. Para ello, se ha utilizado un modelo basado en SVR para estimar el nivel de interferencia física y el tiempo que necesita un grupo de trabajo para finalizar una tarea. Las pruebas experimentales muestran que el modelo SVR ajusta notablemente bien el tiempo de ejecución y, de esta manera, nuestro sistema permite ajustar el número de robots necesarios en cada grupo de trabajo sin necesidad de costosos mecanismos de monitorización del progreso de la tarea. Además, la versión *on-line* permite tener en cuenta múltiples fuentes de error, como errores en la cinemática de los robots o interacciones entre robots pertenecientes a diferentes coaliciones. Se trata del primer sistema que tiene en cuenta el nivel de interferencia para crear un modelo del tiempo de ejecución y utilizar este modelo en un sistema de subastas. Además, a diferencia de otros trabajos, como el realizado por Fua y Ge [47], la utilidad de la coalición de robots no sigue una fórmula predeterminada, sino que se puede ir aprendiendo mediante el modelo SVR. Otros autores, como por ejemplo Vig [127, 128] permiten múltiples robots por tarea, pero sin utilizar *deadlines* asociados, ni realizar ningún tipo de predicción sobre el tiempo que necesitará una coalición para finalizar una determinada misión. Así, Vig asume que las capacidades de las coaliciones de robots son conocidas a priori, en nuestro caso, estas capacidades o utilidades se obtienen en buena parte mediante el modelo SVR.

Autores como Jones et al. [76, 79] tienen, en cierta manera, en cuenta las interferencias entre robots, mediante las llamadas restricciones intra-camino. No obstante, no se utilizan ni robots reales ni simuladores realistas para obtener los resultados. Tampoco se realiza ningún tipo de previsión sobre el tiempo de ejecución de las tareas. En otro trabajos de Jones et al. [77] se utilizan también modelos basados SVR en procesos de subasta para calcular el valor de las pujas, pero en este sistema únicamente se puede asignar un robot por tarea y tampoco se tiene en cuenta la interferencia física.

En [129, 131] Viguria et al. proponen un sistema en el que el número de robots asignados a cada tarea no está prefijado, sino que depende de las características sensoriales y de capacidad de cálculo de los robots. En cambio, no se hace ninguna previsión sobre el tiempo de ejecución, no se tiene en cuenta la interferencia física entre robots, ni tampoco se incluyen restricciones temporales. Finalmente, otros mecanismos de modelización de la interferencia, como por ejemplo los presentados por Lerman y Galstyan en [92, 91], sólo permiten un robot por tarea y están ligados a una arquitectura de control concreta.

Capítulo 6

Conclusiones y trabajo futuro

En este capítulo se resumen las principales contribuciones y objetivos alcanzados en esta tesis. También se presentan futuras líneas de investigación a desarrollar a partir de los trabajos realizados hasta el momento.

6.1. Resumen y contribuciones

Esta tesis se ha centrado en el estudio de los mecanismos de asignación de tareas en sistemas multi-robot, especialmente cuando estas tareas han de ser ejecutadas antes de un determinado instante de tiempo y cuando los robots pueden formar coaliciones para llevarlas a cabo. En primer lugar, se ha llevado a cabo un estudio comparativo entre sistemas basados en subasta y mecanismos de *swarm*, cuando un único robot puede ser asignado a cada tarea. Se tarta del primer estudio realizado de este tipo, ampliando los trabajos de autores como Kalra y Martinoli [81]. Como resultado de este análisis se han propuesto 3 nuevos algoritmos: PSW-R, PSW-D y EDFBP. Los algoritmos PSW-R y PSW-D se basan en un nuevo concepto llamado *swarm* pseudo-aleatorio, que combina un proceso de decisión basado en probabilidades con mecanismos de tipo determinista. Los resultados experimentales muestran como ambos sistemas mejoran los resultados de los algoritmos de *swarm* clásicos, basados sobre todo en *response threshold*. Por otra parte, el sistema EDFBP obtiene resultados

similares a los obtenidos con el método de 'mejor pareja', pero con una complejidad de computo, en general, muy inferior.

En esta tesis también se han tenido en cuenta por primera vez los efectos de la interferencia física entre robots en un proceso de subasta. Los resultados experimentales han demostrado que incluir este factor permite mejorar de manera notable los resultados. Para ello, se ha definido un nuevo sistema de subastas que utiliza conceptos heredados de los mecanismos basados en *response threshold* y que permite asignar múltiples robots a una misma tarea. También se ha demostrado que uno de los principales problemas que tienen los mecanismos de subastas, y también los basados en *swarm*, para llevar a cabo tareas con restricciones temporales es la predicción del tiempo de ejecución de estas tareas. Este problema es especialmente importante cuando múltiples robots pueden ser asignados a una misma tarea. De entre los múltiples factores que afectan al tiempo de ejecución, nos hemos centrado en la interferencia física entre robots. Para ello se ha propuesto un modelo basado en SVR, que predice el tiempo que necesita una coalición de robots para finalizar una tarea, junto con un nuevo mecanismo de subasta, llamado subasta doble. Los experimentos realizados, con diferentes tipos de funciones de utilidad, entornos, etc., han demostrado que este nuevo sistema ofrece mejores resultados que los métodos clásicos de subastas. Además, el sistema de predicción de tiempos de ejecución evita la necesidad de utilizar sistemas para monitorizar el progreso de la tarea y permite, al contrario de lo que sucede en otros métodos, no tener que prefijar el número de robots de cada coalición.

Por tanto, a la pregunta planteada inicialmente: "¿Pueden los sistemas actuales de asignación de tareas basados en *swarm* y subastas afrontar tareas con restricciones temporales?", podemos constatar que los mecanismos actuales, tanto de *swarm* como de subastas, aún tienen muchas limitaciones al respecto. Esta tesis ha presentado varias aportaciones encaminadas a solucionar la que, desde nuestro punto de vista, es una de las principales limitaciones: la predicción en el tiempo de ejecución. Más concretamente, las soluciones aportadas se encaminan a modelizar la interferencia

física, a pesar de ello, existen aún otros factores que podrían mejorar el modelo de predicción que se dejan como trabajo futuro.

6.2. Trabajo futuro

A continuación se explican algunas de las líneas de trabajo futuro que se prevé desarrollar a partir de los métodos propuestos en esta tesis:

- **Mejoras en el modelo del tiempo de ejecución**: el modelo de previsión del tiempo de ejecución basado en SVR únicamente tiene en cuenta el número de robots y la distancia entre el objeto y punto de descarga. Este modelo necesita ser ampliado para poder llevar a cabo otro tipo de tareas. En primer lugar se han de tener en cuenta las capacidades individuales de cada robot y las características del entorno (densidad de obstáculos, tamaño de éstos, etc.). La capacidad de los sistemas basados en SVR para tratar vectores con gran cantidad de características simplificará notablemente todo el proceso. En [78] Jones et al. también plantean el mismo tipo de mejoras sobre su modelo SVR, pero sin tener en cuenta las restricciones temporales y permitiendo tan sólo un robot por tarea.

- **Planificación de las tareas**: todos los mecanismos de asignación de tareas aquí propuestos son, siguiendo la taxonomía de Gerkey, de tipo IA. El modelo de predicción del tiempo de ejecución, propuesto en esta tesis, puede servir de base para mejorar la planificación temporal de las tareas y, de esta manera, mejorar el rendimiento de los actuales sistemas que solucionan problemas de tipo TE. Se está analizando el uso del método de subastas aquí propuesto junto con mecanismos de planificación como los propuestos por Lemaire et al. en [90] o por Zlot y Stentz en [141].

- **Nuevos tipos de tareas fuertemente acopladas**: las misiones propuestas en esta tesis permiten múltiples robots por tarea, pero pueden considerarse como

débilmente acopladas. En un futuro se pretende utilizar los mecanismos aquí propuestos para llevar a cabo misiones más complejas, en las que robots con diferentes capacidades sensoriales se vean obligados a cooperar para llevar a cabo una tarea. Entre estos métodos cabe destacar el propuesto por Viguria et al. en [130], el cual se podría utilizar, junto con el sistema de subastas dobles y el mecanismo de predicción del tiempo de ejecución aquí propuesto.

- **Mejoras en los mecanismos de subastas**:
  - Permitir que los robots decidan cuándo pujar por una tarea y cuándo esperar a tener una mejor opción, tal como proponen otros autores como Busquets y Simmons en [20]. De esta manera, se reduciría la cantidad de mensajes necesarios para implementar los métodos de subastas.
  - Incluir subastas combinatorias, en las que los líderes y demás robots pujen por un conjunto de tareas.
  - Utilizar nuevos mecanismos de planificación de caminos para solucionar el problema TSP en los algoritmos de tipo LP planteados en el capítulo 3. La solución propuesta actualmente para estos problemas es mejorable. Actualmente existen múltiples alternativas, como las explicadas por Böckenhauer et al. en [13] o por Lawler et al. en [88].
- **Mecanismos para reducir la interferencia física**:
  - Nuevas estrategias de evitación de obstáculos y coordinación del movimiento: analizar el comportamiento del sistema usando otras estrategias para evitar obstáculos que pudiesen disminuir la interferencia física. Entre ellas cabe destacar la estrategia de competición agresiva propuesta por Vaughan et al. en [126].
  - Mecanismos de división territorial: en el capítulo 3 se han presentado algunos métodos de división del espacio para reducir la interferencia. Hay

otras estrategias, como la propuesta por Pini et al. en [107] que también se pueden incluir en los procesos de subastas explicados en este documento.

6.3. Publicaciones relacionadas

Como resultado de los trabajos de investigación descritos a lo largo de esta memoria, se han realizado las publicaciones que se detallan a continuación, junto con las principales contribuciones realizadas en cada una de ellas:

- J. Guerrero y G. Oliver. 2010. "Auction and swarm multi-robot task allocation algorithms in real time scenarios", capítulo en *Multi-Robot Systems, Trends and Development*, *Intech publisher* (En prensa). Estudio comparativo de los sistemas de subastas y *swarm* con restricciones temporales. Propuesta del algoritmo EDFBP.

- J. Guerrero y G. Oliver. 2010. "A multi-robot auction method to allocate tasks with deadlines" en *7th IFAC Symposium on Intelligent Autonomous Vehicles*. Nuevo mecanismo de subasta doble y modelo *on-line* de SVR.

- J. Guerrero y G. Oliver. 2007. "Interference modelization in Multi-Robot Auction Methods" en *6th IFAC Symposium on Intelligent Autonomous Vehicles*. Utilización del modelo SVR para modelizar la interferencia.

- J. Guerrero y G. Oliver. 2006. "Monitoring and interference impact in multi-robot auction methods" en *Workshop on Auction Mechanisms for Robot Coordination* como parte del *21th National Conference on Artificial Intelligence*. Nuevo mecanismo de monitorización mecanismos de subastas con restricciones temporales.

- J. Guerrero y G. Oliver. 2006. "Physical interference impact in multi-robot task allocation auction methods" en *IEEE Workshop on Distributed Intelligent Systems*. Ajuste polinómico para modelizar la interferencia en tareas con restricciones temporales.

- J. Guerrero y G. Oliver. 2005. "Auction like task allocation and motion coordination strategies for multi-robot transport tasks" en *ICINCO International Workshop on Multi-Agent Robotic Systems*. Comportamiento *lane keeping* para la coordinación del movimiento.

- J. Guerrero y G. Oliver. 2004. "Multi-Robot Task Allocation Method for Heterogeneous Tasks with Priorities" en 7^{th} *International Symposium on Distributed Autonomous Robotic Systems*, publicado en *Distributed Autonomous Robotic Systems 6*, Springer-Verlag. Mecanismos de subastas sin restricciones temporales con prioridades.

- J. Guerrero y G. Oliver. 2003. "Multi-robot task Allocation strategies using auction-like mechanisms" en *Sisè Congrés Català d'Intel.ligència Artificial*, publicado en *Artificial Research and Development* de la serie *Frontiers in Artificial Intelligence and Applications*, Vol. 100. IOS Press. Estrategia preemptiva basada en subastas para la creación de coaliciones de robots con tareas sin restricciones temporales.

- J. Guerrero y G. Oliver. 2003. "A Multi-robot task allocation method to regulate working group sizes" en 1^{st} *EURON European Conference on Mobile Robots*. Mecanismos de subastas para la creación de coaliciones, concepto de TH.

Anexo A

Arquitectura de los robots

En este anexo se explica la arquitectura implementada en los robots utilizados durante las pruebas experimentales. Esta arquitectura fue, en su mayor parte, desarrollada por los autores como trabajo previo a esta tesis y publicada en [62, 60]. Se trata de una arquitectura basada en comportamientos [3, 4]. Este tipo de arquitecturas están inspiradas en la fisiología animal, según la cual un estímulo externo provoca una determinada respuesta refleja. En el caso de sistemas artificiales, los sensores proporcionan los estímulos y las respuestas se convierten en las consignas que recibirán los actuadores del robot. Un comportamiento reactivo primitivo supone una toma de decisión instantánea, siendo sólo función del estímulo que la genera. Los comportamientos primitivos se pueden componer mediante operadores de cancelación y agregación aplicados sobre las respuestas. Estos sistemas se caracterizan por su bajo coste computacional y modularidad intrínseca. Por contra, la complejidad de las tareas que pueden llevar a cabo las arquitecturas basadas únicamente en comportamientos reactivos es muy limitada. Para realizar misiones más complejas es necesario utilizar las llamadas arquitecturas híbridas, que implementan módulos de más alto nivel encargados de planificar la ejecución de la tarea.

A.1. Estructura funcional

La arquitectura implementada en este trabajo se engloba dentro del grupo de arquitecturas híbridas, donde las funciones a realizar están divididas en los siguientes 3 niveles o módulos: sensorial, adaptación y misión, cada uno de los cuales se comunica únicamente con sus inmediatos vecinos, tal como se puede ver en la figura A.1. La funcionalidad de cada módulo es la siguiente:

- Nivel Sensorial: este módulo ejecuta las tareas de más 'bajo nivel' interactuando directamente con el hardware del vehículo. Concretamente, se encarga de la lectura de los sensores, enviando los datos al nivel de adaptación. También se encarga de adaptar las ordenes procedentes de niveles superiores para que puedan ser enviadas directamente a los actuadores del robot.

- Nivel de Adaptación: recibe del nivel de misión un punto objetivo hacia el que el robot se ha de dirigir. El nivel de adaptación se encarga de decidir los movimientos que ha de realizar el vehículo para llegar hasta el punto objetivo de manera segura, esto es, evitando los obstáculos que puedan aparecer en el camino. La dirección que ha de seguir el robot es enviada al nivel sensorial para que, a su vez, sea transmitida a los actuadores del robot. Para implementar este nivel se han utilizado una serie de comportamientos, explicados en la sección A.2.

- Nivel de Misión: éste es el módulo implementado de mayor nivel, en él se decidirán los puntos intermedios por los que ha de pasar el robot para finalizar su misión. También es el nivel encargado de la comunicación entre robots y de implementar los algoritmos de subastas o de *swarm* explicados a lo largo de esta tesis. La misión realizada durante las pruebas experimentales ha sido la de recogida de objetos, la qual se ha implementado mediante la máquina de estados que se puede ver en la figura A.2. En el estado *Libre* el robot no tiene ningún objeto asignado y se encuentra a la espera de encontrar uno nuevo. Cuando lo encuentra o se le asigna uno nuevo, pasa al estado *Aproximar*, durante el cual

el robot se aproxima al objeto a recoger. Una vez el robot alcanza esa posición, pasa al estado *Cargar* en el que acumulará el peso del objeto hasta cargarlo totalmente o hasta llegar a su límite de capacidad de carga. Luego, pasará al estado *Ir* para llevar la carga transportada al punto de descarga. Una vez el robot está situado a una determinada distancia del punto de descarga, pasará al estado *Descargar*, en el que se libera el peso transportado. Si después de este proceso aún queda peso para transportar, el robot vuelve al estado *Ir*. Además, desde cualquier estado el robot puede pasar a *Libre* si recibe un mensaje de finalización de tarea. Si un robot que está en el estado *Ir* recibe un mensaje de finalización, éste no quedará libre hasta que no haya dejado su carga en el punto de descarga.

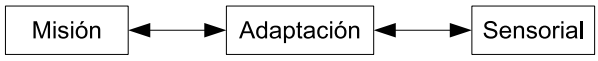


Figura A.1: Niveles funcionales de la arquitectura.

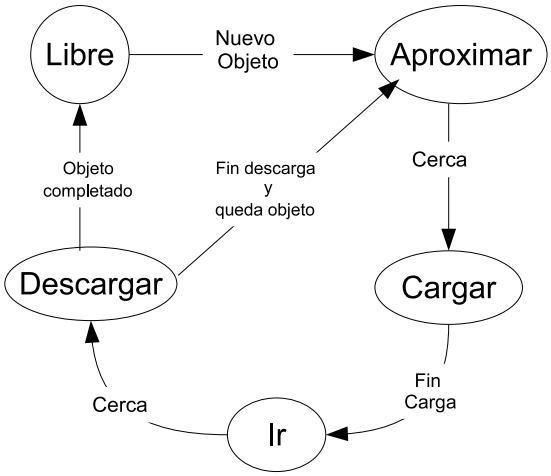


Figura A.2: Máquina de estados de la tarea de recogida de objetos.

A.2. Comportamientos implementados

El nivel de adaptación ha sido implementado mediante una serie de comportamientos que combinados permiten al robot alcanzar un punto objetivo de manera segura. Estos comportamientos están basados en una evolución de las técnicas de campos de potencial llamada *motor schemas*, propuesta por Arkin [3]. Cada *motor schema* se encarga de generar una fuerza de repulsión o de atracción, que indicará la dirección ($\vec{v}_i$) que ha de tomar el robot desde su punto de vista. Los vectores resultantes de cada *motor schema* se combinan mediante una suma ponderada para dar como resultado el vector de dirección ($\vec{V}$) que ha de tomar el robot, tal como se puede ver en la siguiente ecuación:

$$\vec{V} = \sum_{i=1}^{N} w_i \cdot \vec{v}_i \tag{A.1}$$

donde w_i es el peso o valor relativo dado a cada *motor schema* y N es el número total de *motor schemas*. Se han implementado un total de 5 *motor schemas* diferentes: *ir hacia objetivo*, *evitar obstáculo*, *inercia*, *evitar el pasado* y *lane keeping*. La figura A.3 muestra el esquema general del nivel de adaptación y la manera en que interacciona con los demás niveles de la arquitectura.

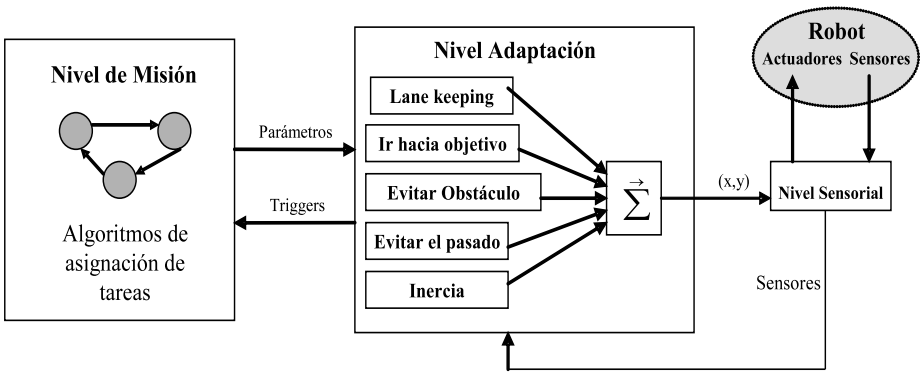


Figura A.3: Visión general de la arquitectura implementada en los robots.

El *motor schema ir hacia objetivo* es el encargado de generar una fuerza de atracción ($\vec{v}_1$) en dirección al punto objetivo del robot, mediante la siguiente ecuación:

$$\vec{v}_1 = \frac{\vec{x}_g - \vec{x}}{\| \vec{x}_g - \vec{x} \|} \tag{A.2}$$

donde $\vec{x}_g$ es la posición del punto objetivo y $\vec{x}$ es la posición del robot.

El *motor schema evitar obstáculos* genera una fuerza de repulsión por cada obstáculo o_i detectado, cuyo módulo es inversamente proporcional al cuadrado de la distancia entre el robot y el obstáculo. El vector final de *evitar obstáculos* ($\vec{v}_2$) se obtiene mediante la suma de las fuerzas de repulsión generadas por cada uno de los obstáculos detectados, utilizando la siguiente ecuación:

$$\vec{v}_2 = \sum_{i=1}^{m} \frac{1}{d_i^2} \frac{\vec{x} - \vec{x}_{oi}}{\| \vec{x} - \vec{x}_{oi} \|} \tag{A.3}$$

donde m es el número de obstáculos detectados, $\vec{x}_{oi}$ es la posición donde se encuentra el obstáculo y d_i es la distancia entre este obstáculo y el robot. Si d_i es mayor a un determinado valor, entonces se supone que el obstáculo no es un peligro y no se tiene en cuenta su interacción con el robot.

El *motor schema evitar el pasado* implementa el algoritmo *avoiding the past* propuesto por Balch y Arkin en [8] para evitar los problemas de *trapping* presentes en las técnicas basadas en campos de potencial. Este problema se debe a que el movimiento del robot depende de un campo de fuerzas que pueden provocar que quede atrapado en un mínimo local, dando lugar un camino cerrado del que no puede salir. Para implementar este *motor schema* se necesita utilizar una determinada cantidad de memoria para recordar los puntos ya visitados por el robot y poderlos evitar posteriormente. La memoria consiste en un mapa métrico del entorno del robot organizado en celdas, en el que en cada una de ellas se almacena el número de veces que ha sido visitada por el robot. La dirección de la fuerza se calcula como la opuesta a la dirección del gradiente del mapa. La magnitud es proporcional al número total de veces que el robot ha visitado los puntos contenidos en la ventana (entorno alrededor del robot) utilizada para calcular el gradiente. De esta manera, el valor de la fuerza se calcula utilizando la siguiente expresión:

$$\vec{v}_3 = -\nabla M \cdot \frac{\sum_{i=-n,j=-n}^{i=n,j=n} M(x-i, y-j)}{4n^2 \cdot max\{M\}} \tag{A.4}$$

donde M es el mapa, $2n$ es su dimensión, (x, y) es la posición del robot y $Max\{M\}$ es el valor máximo que puede almacenar una celda de M.

El *motor schema inercia* genera un vector ($\vec{v}_4$) que tiene la misma dirección y sentido que la velocidad del robot. Su objetivo es suavizar la trayectoria generada por el resto de *motor schemas*.

Finalmente, el *motor schema lane keeping* genera un vector ($\vec{v}_5$) cuyo cálculo ya ha sido descrito en el capítulo 4.

Anexo B

Entorno de desarrollo y experimentación

En este anexo se explican las herramientas y el entorno de desarrollo usados para esta tesis. Se describen las características más importantes de los simuladores utilizados: RoboCoT, RoboSim y Player/Stage, así como la infraestructura usada en las pruebas con robots reales.

B.1. *Robot Colonies Tool* (RoboCoT)

En esta sección se explicará el simulador *Robot Colonies Tool* (RoboCoT) utilizado en parte de las pruebas experimentales de esta tesis. Se trata de un simulador inicialmente desarrollado por Guerrero, Oliver y Ortiz [60, 61] como trabajo previo a esta tesis y que en el transcurso de la misma ha sido ampliado y mejorado. RoboCoT permite simular el comportamiento de colonias de robots, con diferentes características, entornos, y misiones. La principal ventaja que ofrece RoboCoT respecto a otros simuladores, como Player/Stage, es su rapidez, ya que el comportamiento de todos los robots siempre se ejecuta en una misma máquina, sin necesitad de complejos mecanismos de comunicación entre módulos. Como contrapartida a esta rapidez, RoboCoT es menos flexible y genérico que el entorno Player/Stage.

RoboCoT ha sido desarrollado en C++ sobre plataforma Windows. Su sencilla estructura permite que se pueda ejecutar sobre máquinas con prestaciones reducidas, como por ejemplo ordenadores Pentium, con una frecuencia de 200MHz y 32MB de memoria. La figura B.1 muestra la pantalla principal de este simulador con un único robot en el entorno. Entre las diferentes características del simulador RoboCoT cabe destacar la posibilidad de modificar el entorno (añadir nuevos objetos, paredes, etc.) y la de crear robots con diferentes capacidades cinemáticas y sensoriales. La figura B.2(a) muestra la pantalla de configuración de los sensores de un robot, que permite incluir sensores de contacto, de distancia y de presencia. La pantalla muestra la configuración de sensores utilizada en las pruebas experimentales realizadas en esta tesis: 4 sensores de distancia con un alcance 45 unidades y sensores de contacto alrededor del robot. Los sensores de distancia permiten obtener la posición exacta a la que se encuentra un obstáculo respecto al robot, mientras que los de contacto son de tipo binario, indicando únicamente si hay, o no, un obstáculo.

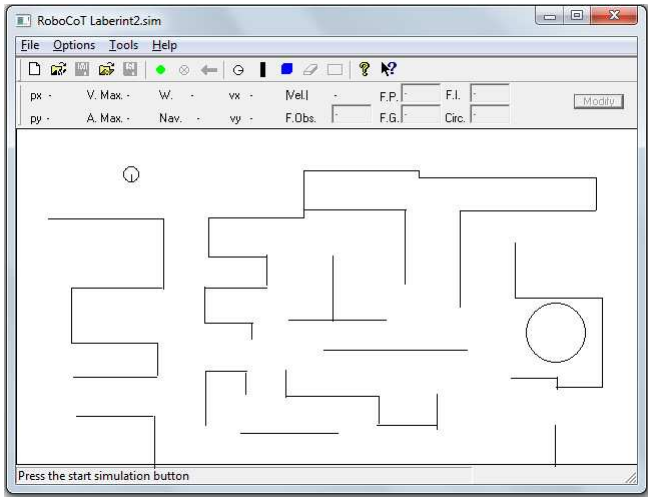
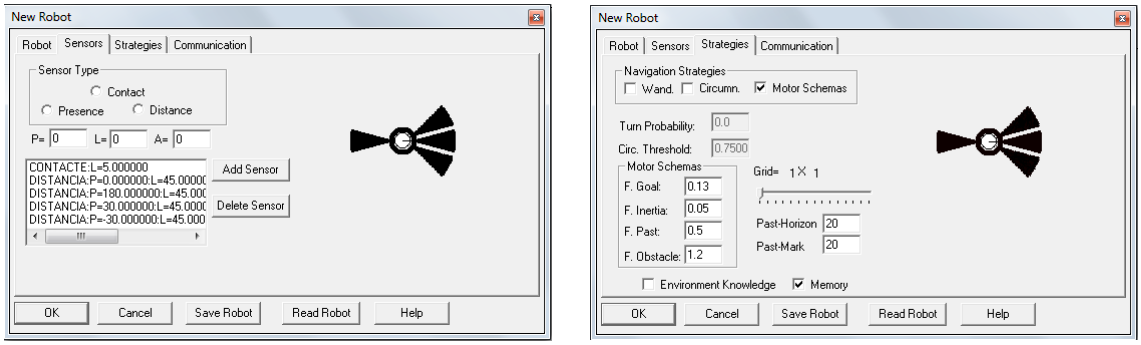


Figura B.1: Pantalla principal del simulador RoboCoT.

La figura B.2(b) muestra la pantalla de configuración de los diferentes comportamientos (*motor schemas*) y estrategias de evitación de obstáculos implementadas en RoboCoT. Los robots pueden implementar las técnicas de evitación de obstáculos de vagar, circunvalar, así como, los métodos basados en *motor schemas*, vistos en el

anexo A. En el caso de este ejemplo, el robot únicamente implementa la estrategia de *motor schemas*, donde *F. Goal* es el peso del vector de *ir hacia objetivo*, *F. Inertia* el del *motor schema inercia*, *F. Past* el peso de *evitar el pasado* y *F. Obstacle* es el peso de *evitar obstaculos*. Los valores de *Past-Horizon* y *Past-Mark* son el tamaño de la ventana (número de celdas) del algoritmo *avoiding the past*. Los valores de los diferentes parámetros que se pueden ver en la figura son iguales a los utilizados en las pruebas experimentales explicadas en los capítulos anteriores.



(a) Pantalla de configuración de los sensores de un robot.

(b) Pantalla de los comportamientos (*motor schemas*) del robot.

Figura B.2: Ejemplos de pantallas del simulador RoboCoT

El simulador permite definir objetos con diferentes características: tamaño, peso, tipo, etc. y varios tipos de tareas. Las tareas definidas en RoboCoT son de tres tipos: ir a un determinado punto del espacio, recogida de objetos, localización de objetos. La tarea de recogida de objetos es la utilizada en las pruebas experimentales explicada en los capítulos 4 y 5. La tarea de localización de objetos es similar a la de recogida, pero ahora no es necesario llevar el peso hasta un punto de descarga, sino que los robots se quedan cerca del objeto cargándolo hasta haber finalizado. Finalmente, la tarea de ir a un determinado punto del espacio, consiste en que todos los robots de la colonia han de pasar, secuencialmente, por una serie de posiciones del espacio establecidas por el usuario.

B.2. RoboSim

El simulador RoboSim ha sido desarrollado durante esta tesis para permitir la simulación del comportamiento de un gran número de robots y tareas. Este simulador únicamente implementa el nivel de misión de la arquitectura, que incluye los algoritmos de asignación de tareas y comunicación entre robots. En cada unidad de tiempo de simulación se calcula la siguiente posición de cada robot a partir de sus características cinemáticas y de su posición actual. El cálculo no incluye los algoritmos de evitación de obstáculos ni navegación, ya que se supone que los robots no pueden colisionar con ningún elemento del entorno. De esta manera, se puede analizar como influyen los mecanismos de asignación de tareas sobre el comportamiento del sistema de manera aislada de otros factores. Además, esta simplicidad permite reducir de manera muy substancial el tiempo de simulación con respecto a otros entornos más realistas.

RoboSim ha sido desarrollado en C++ y probado sobre un ordenador Pentium IV a 2GHz con 512MB de memoria utilizando la distribución de linux Ubuntu 8.04. Permite la ejecución de tareas de recogida de objetos, tipo *foraging* e implementa todos los algoritmos explicados en el capítulo 3, tanto los basados en *swarm* como los de subastas. Este simulador permite como parámetros de entrada, un fichero con el conjunto de objetos a recoger y los tiempos de aparición de cada uno de estos objetos en el entorno. Este último fichero indica los instantes de tiempo, generados siguiendo un proceso de Poisson, en los que han de aparecer los diferentes objetos. De esta manera, se asegura que los instantes de aparición de tareas siempre son los mismos para las diferentes pruebas realizadas.

B.3. Pruebas con Player/Stage y robots reales

El último de los simuladores o entornos utilizados es el Player/Stage implementado por Gerkey et al. [29] y ampliamente usado por otros autores. Player es un servidor que permite el control de diferentes tipos de robots y sensores, utilizando el protocolo

TCP/IP. De esta manera, los clientes que se comunican con el servidor Player pueden controlar diferentes tipos de dispositivos sin necesidad de conocer detalladamente su hardware. Además, Player puede actuar indistintamente sobre el sistema real o sobre un simulador, que en nuestro caso es el simulador Stage. La aplicación Stage permite simular un amplio número de dispositivos y robots de forma realista, de manera que cualquier programa probado sobre Stage pueden utilizarse, sin prácticamente ninguna modificación, sobre los robots reales. La figura B.3, muestra un ejemplo de la pantalla principal del simulador Stage en la que aparecen 5 robots con una serie de sensores de ultrasonidos. En todos los casos, la versión ha sido al 2.0.3, tanto de Player como del simulador Stage.

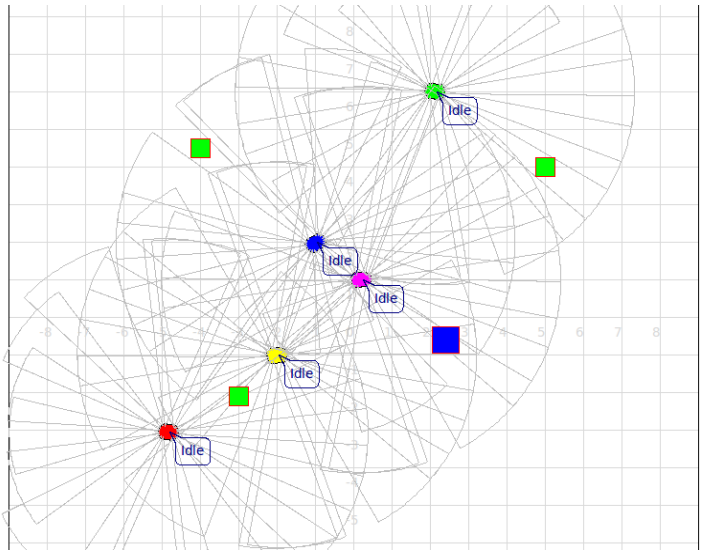


Figura B.3: Pantalla principal del simulador Stage.

La figura B.4 muestra el esquema general utilizado tanto en las pruebas con robots reales como en las simulaciones con Player/Stage. Los diferentes componentes del sistema se comunican mediante TCP/IP y en cada robot se ejecuta tanto el servidor Player como la aplicación cliente, que incluye toda la arquitectura, algoritmos de MRTA, etc. En la figura B.5 se pueden ver las diferentes unidades funcionales en las que se divide el programa cliente: *Arquitectura*, *Comunicaciones* y *MRTA*. Los módulos del programa cliente se comunican entre si mediante memoria compartida y

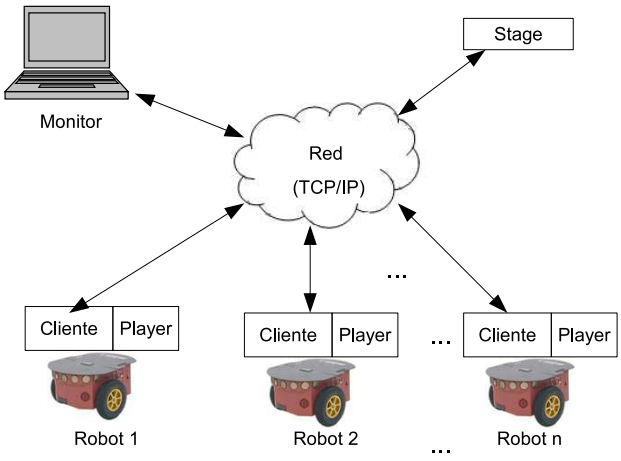


Figura B.4: Comunicaciones entre los componentes del sistema en el entorno Player/Stage.

cada uno de ellos se ejecuta en un hilo de ejecución diferente. El módulo *Arquitectura* implementa parte de la arquitectura vista en el anexo A: nivel sensorial, de adaptación y la máquina de estados que ejecuta la tarea de nivel de misión. Este módulo se comunica mediante TCP con el servidor Player para solicitar al robot información sobre las lecturas de los sensores, enviar las ordenes a los actuadores, etc. En el caso de utilizar robots reales, el servidor Player también se comunica mediante una conexión RS-232 con el robot real. El módulo *Comunicaciones* se encarga de las comunicaciones con el resto de robots del sistema y con el programa *monitor* mediante el protocolo UDP. Este módulo únicamente leerá los diferentes mensajes y los irá almacenando, de manera secuencial, en una cola, para su posterior tratamiento. Finalmente, el módulo funcional *MRTA* implementa los mecanismos de asignación de tareas explicados a lo largo de esta tesis. Entre otras tareas, este módulo irá leyendo los mensajes recibidos a través del módulo *Comunicaciones* y actuará en consecuencia. La tabla B.1 muestra el listado de los parámetros utilizados en los algoritmos de MRTA junto con el valor que se les ha asignado en las pruebas experimentales.

En las pruebas experimentales con robots reales se han utilizado 4 robots Pioneer 3DX equipados con dos anillos de 8 sensores de ultrasonidos cada uno. Los sensores de ultrasonidos tienen un alcance de entre 10cm y 4m y el posicionamiento de los

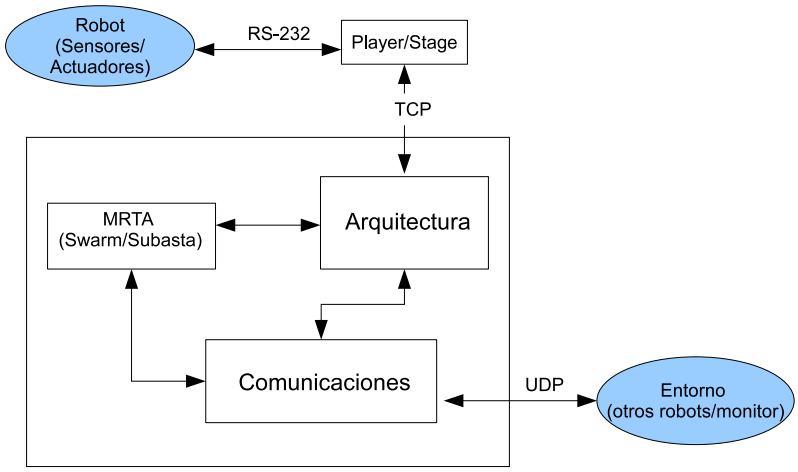


Figura B.5: Módulos de la aplicación cliente.

$TIME\_LEADERSHIP$	2s
$TIME\_AUCTION$	2s
$TIMEBIDACCEPTED$	2s
$TIME\_LEADER\_ALIVES$	10s
$TIME\_BETWEEN\_ALIVES$	10s
$TIME\_BETWEEN\_REMOVE$	20s

Tabla B.1: Valores de los parámetros de los algoritmos de asignación de tareas.

robots se realiza únicamente mediante odometría. Cada robot se comunica a través de su puerto serie con un ordenador que lleva abordo, el cual tiene una placa Via Epia ME6000 con un procesador EDEN a 600MHz, 512MB de memoria RAM y 40GB de disco duro y en el que se ejecuta la distribución de GNU/linux Ubuntu 6.10, junto con el servidor Player y la aplicación cliente. Los diferentes robots se comunican a través de un punto de acceso utilizando las tarjetas wireless conectadas a ellos. En la figura B.6 se puede ver una imagen del ordenador con el que están equipados los robots.

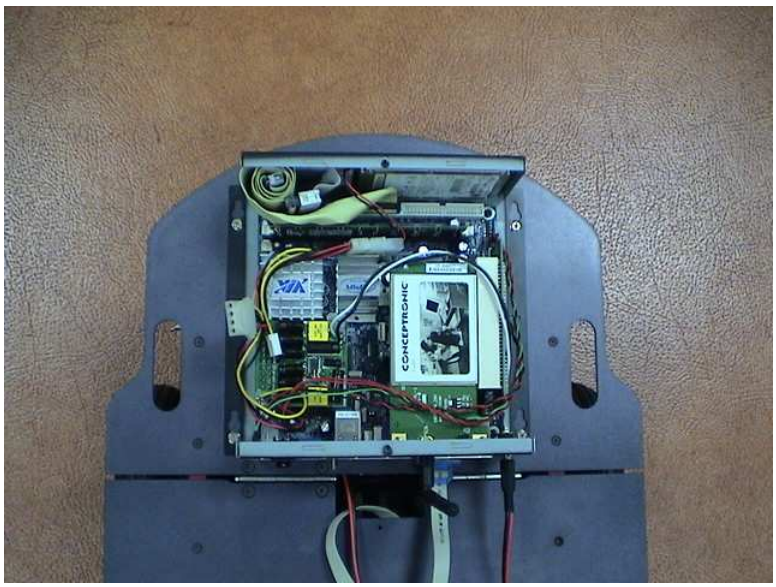


Figura B.6: Ordenador con el que están equipados los robots.

Glosario

CNP *Contract Net Protocol*, pág. 26.

deadline Tiempo límite para finalizar una tarea, pág. 5.

$\Delta - D_L TSP$ Problema TSP con *deadlines* en espacios métricos, pág. 19.

EDF *Earliest Deadline First*, pág. 27.

EDFA *Earliest Deadline First Auctions*, pág. 36.

EDFBP *Earliest Deadline First Best Pair*, pág. 36.

Estrategia voraz Estrategia de subastas en la que el líder intenta crear un grupo de trabajo con tantos robots como sea posible., pág. 147.

follow the preceding Estrategia de coordinación de movimiento en la que cada robot sigue al robot que le precede, pág. 107.

foraging Tarea de búsqueda y recolección de objetos, pág. 2.

grupo de trabajo Conjunto de robots encargados de realizar una determinada tarea, pág. 87.

IA *Instantaneous Assignment*. Problemas en los que no se permite realizar ningún tipo de planificación sobre futuras asignaciones de tareas, pág. 12.

JSS *Job Shop Scheduling Problem*, pág. 16.

lane keeping Estrategia de coordinación del movimiento propuesta en esta memoria, pág. 7.

líder Robot encargado de formar el grupo de trabajo, pág. 87.

LP *Local planning*. Conjunto de métodos que permiten a los robots decidir el orden de ejecución de las tareas asignadas, incluye los siguientes algoritmos: SUA-LP, EDFA-LP y SBPA-LP, pág. 53.

M_I Conjunto de experimentos que utilizan monitorización y el modelo de interferencia, pág. 148.

M_NI Conjunto de experimentos que utilizan monitorización pero no el modelo de interferencia, pág. 148.

MR *Multi-robot tasks*. Problemas en los que varios robots pueden ser asignados a una misma tarea, pág. 12.

MRTA *Multi-Robot Task Allocation* (Asignación de tareas en sistemas multi-robot). Mecanismos que permiten decidir qué robots han de ser asignados a cada tarea, pág. 12.

MT *Multi-Task*. Problemas en los que cada robot puede ejecutar múltiples tareas al mismo tiempo, pág. 12.

NFS Algoritmo *Nearest First Task*, pág. 39.

NM_I Conjunto de experimentos que no utilizan monitorización pero sí el modelo de interferencia, pág. 148.

NM_NI Conjunto de experimentos que no utilizan monitorización ni el modelo de interferencia, pág. 148.

NP Conjunto de problemas que pueden solucionarse en un tiempo polinómico con una máquina de Turing no determinista, pág. 14.

NP-completo Un problema X es NP-completo si X es NP y NP-hard, pág. 14.

NP-hard Un problema X es de tipo NP-hard si cualquier otro problema NP se puede convertir en un tiempo polinómico a un problema de tipo X, pág. 11.

OAP *Optimal Assignment Problem*, pág. 13.

P Conjunto de problemas que pueden solucionarse en un tiempo polinómico con una máquina de Turing determinista.

PSW Métodos de *Pseudo-random Swarm*, pág. 41.

PSW-D Algoritmo de Pseudo-random *Swarm Distance*, pág. 35.

PSW-R Algoritmo de *Pseudo-random Swarm Robot*, pág. 35.

RBF *Radial Basis Function*, pág. 136.

RCPSP *Resource Constrained Project Scheduling Problem*, pág. 18.

RoboCoT Simulador *Robot Colonies Tool*, pág. 83.

RoboSim Simulador RoboSim, pág. 54.

RTH Algoritmo de *response threshold*, pág. 40.

SA *Single Auction*. Véase definición de subasta única, pág. 147.

SBPA *Sequential Best Pair Auctions*, pág. 36.

SCP *Set Covering Problem*, pág. 14.

SMO *Sequential Minimal Optimization*, pág. 137.

solución k-óptima Solución a un problema que es k veces peor que su solución óptima, pág. 20.

SPP *Set Partitioning Problem*, pág. 14.

SR *Single-Robot*. Problemas en los que cada tarea únicamente puede ser ejecutada por un robot, pág. 12.

SSA *Sequential Single-item Auctions*, pág. 26.

ST *Single-Task*. Problemas en los que cada robot únicamente puede ejecutar una tarea simultáneamente, pág. 12.

SUA *Sequential Unordered Auctions*, pág. 36.

subasta de robots Parte del proceso de subasta doble en la que los líderes de una tarea pujan por los robots seleccionados, pág. 139.

subasta de tareas Parte del proceso de subasta doble en la que los robots pujan por una tarea, pág. 139.

subasta doble Estrategia de subastas en la que el proceso se divide en dos partes: subasta de tareas y subasta de robots, pág. 139.

subasta única Estrategia de subastas en la que únicamente se utiliza la subasta de robots, pág. 148.

subasta simple Proceso de subasta utilizado en entornos con tareas sin restricciones temporales, pág. 139.

SVR *Support Vector Regression*, pág. 7.

swarm intelligence Mecanismo por el que surge un comportamiento colectivo complejo a partir de la interacción entre los miembros de un sistema formado por múltiples agentes simples, pág. 21.

TE *Time-Extendent Assignment*. Problemas en los que se permite realizar una planificación temporal de las tareas a asignar, pág. 12.

TSP *Traveling Salesman Problem*, pág. 19.

VRPSTW *Vehicle Routing Problem with Soft Time Windows*, pág. 18.

VRPTW *Vehicle Routing Problem with Time Windows*, pág. 17.

Notación

A	Distancia para pre-seleccionar robots en el algoritmo PSW-R, pág. 42.
B	Valor de una puja durante un proceso de subasta, pág. 95.
C	Peso que se da al error de las muestras en el método SVR, pág. 135.
C^i	Representa la asignación de un conjunto de robots R^i a una tarea t_i o la asignación de un conjunto de tareas T^i a un robot r_i, pág. 38.
D	Distancia a partir de la cual un robot que ejecute el algoritmo NFS no tendrá en cuenta un obstaculo, pág. 39.
d_j	Distancia a la que se encuentra el objeto t_j del punto de descarga, pág. 129.
$DL_{g,j}$	Tiempo de ejecución estimado de una tarea t_j cuando es llevada a cabo por un grupo de robots g, pág. 152.
DL_j	*Deadline* de la tarea t_j, pág. 127.
E_g	Energía del grupo de trabajo g, pág. 93.
E'_{g1}	Energía estimada del grupo de trabajo $g1$, pág. 95.
fT_j	Tiempo de finalización de la tarea t_j, pág. 126.
g	Grupo de robots asignados a la tarea t_j. Dambien denotado por R^j, pág. 126.

$groupCapacity_{g,j}$ Capacidad de trabajo del grupo g para llevar a cabo la tarea t_j, pág. 126.

H Hentropía del sistema, pág. 110.

I Interferencia física, pág. 129.

$idealCapacity_{g,j}$ Capacidad de trabajo ideal del grupo g para llevar a cabo la tarea t_j, pág. 129.

K Función de *kernel* en el método SVR, pág. 136.

λ Frecuencia de llegadas de las tareas, cuando éstas siguen un proceso de Poisson, pág. 37.

$loadCapacity_i$ Capacitad de carga del robot r_i, pág. 128.

Na_r Número de robots cuya distancia al robot r es menor que A. Valor utilizado en el algoritmo PSW-R, pág. 42.

n_g Número de robots en un grupo de trabajo, pág. 91.

n_v Número de vectores en el modelo SVR *on-line*, pág. 138.

O_v Sobrecarga (*overhead*) porcentual consecuencia del intercambio de robots entre grupos de traabjo, pág. 95.

P Planificación. Conjunto ordeanado de puntos del espacio por los que ha de pasar un robot, pág. 51.

P' Planificación estimada tras añadir a una planificación inicial P una nueva tarea, pág. 51.

Pri_j Prioridad de la tarea $t_j \in [1..,5]$, pág. 81.

$P_{r,t}$ Probabilidad de que el robot r ejecute la tarea t, pág. 40.

R^j Conjunto de robots asignados a la tarea t_j, llamado coalición o grupo de robots. También se denota por g, pág. 81.

$R = \{r_1, ..., r_n\}$ Conjunto de robots, pág. 38.

SH Hentropía social, pág. 111.

TA Asignación válida de un conjunto de un conjunto de tareas a un conjunto de robots.

TA^* Asignación óptima de tareas a un conjunto de robots. La asignación TA^* maximiza una determinada función de utilidad U, pág. 38.

$taskWorkLoad_j$ Carga de trabajo. Trabajo necesario para finalizar la tarea t_j, pág. 91.

TH Valor umbral del grupo de trabajo en problemas MR sin restricciones temporales, pág. 91.

TH_g Valor utilizado para limitar el tamaño de los grupos de trabajo en problemas MR sin restricciones temporales. El líder de un grupo selecciona a los robots hasta que $TH_g < TH$, pág. 91.

T^i Subconjunto de tareas asignadas al robot r_i, pág. 38.

$T = \{t_1, ..., t_m\}$ Conjunto de tareas, pág. 38.

U Función de utilidad del sistema. Depende de la asignación de tareas TA, pág. 38.

U_j Valor de la función de utilidad de la tarea t_j, pág. 127.

ε Error tolerado respecto a los datos de la muestra en el método SVR, pág. 135.

V_i Velocidad del robot r_i, pág. 129.

$workCapacity_{i,j}$ Capacitad de trabajo del robot r_i para realizar la tarea t_j, pág. 81.

Bibliografía

[1] W. Agassounon y A. Martinoli, "Efficiency and robustness of threshold-based distributed allocation algorithms in multi-agent systems," en *1st Int. Joint Conference on Autonomous Agents and Multi-Agents Systems*, Bolonia, Italia, Julio 2002, pp. 1090–1097.

[2] W. Agassounon, A. Martinoli y R. Goodman, "A scalable, distributed algorithm for allocating workers in embedded systems in embedded systems," en *IEEE International Conference on Systems, Man and Cybernetics*, vol. 5, Tucson, EUA, Octubre 2001, pp. 3367–3373.

[3] R. C. Arkin, *Behavior-Based Robotics*. The MIT press, 1998.

[4] R. C. Arkin y G. A. Bekey, Eds., *Robot Colonies*. Kluwer Academic Publishers, 1997.

[5] E. Balas y M. W. Padberg, "Set partitioning: A survey," *Siam Review*, vol. 18, no. 4, pp. 710–760, Octubre 1976.

[6] E. Balas, N. Simonetti y A. Vazacopoulos, "Job shop scheduling with setup times, deadlines and precedence constraints," *Journal of Scheduling*, vol. 11, no. 4, pp. 253–262, 2008.

[7] T. Balch, "Social entropy: A new metric for learning multi-robot teams," en *10th international Florida Artificial Intelligence Research Society Conference*, Daytona, EUA, Mayo 1997, pp. 272–277.

[8] T. Balch y R. C. Arkin, "Avoiding the past: A simple but effective strategy for reactive navigation," en *International Conference on Robotics and Automation (ICRA)*, Atlanta, EUA, Mayo 1993, pp. 678–685.

[9] J. Baltes, M. G. Lagoudakis, T. Naruse y S.-S. Ghidary, Eds., *RoboCup 2009: Robot Soccer World Cup XIII*, ser. Lecture Notes on Artificial Intelligence series, vol. 5949. Springer, Heidelberg, 2010.

[10] N. Bansal, A. Blum, S. Chawla y A. Meyerson, "Approximation algorithms for deadline-tsp and vehicle routing with time-windows," en *Annual ACM Symposium on Theory of Computing*, Chicago, EUA, Junio 2004, pp. 166–174.

[11] D. Basak, S. Pal y D. C. Patranabis, "Support vector regression," *Neural Information Processing, Letters and Reviews*, vol. 11, no. 10, pp. 203–224, Octubre 2007.

[12] L. Bayindir y E. Sahin, "A review of studies in swarm robotics," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 15, no. 2, pp. 115–147, 2007.

[13] H. Böckenhauer, J. Kneis y J. Kupke, "Approximation hardness of deadline-tsp reoptimization," *Theoretical Computer Science*, vol. 410, pp. 2241–2249, 2009.

[14] J. C. Beck, P. Prosser y E. Selensky, "Vehicle routing and job shop scheduling: What's the difference?" en *13^{th} International Conference on Automated Planning and Scheduling*, Trento, Italia, Junio 2003, pp. 267–276.

[15] M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin y A. Kleywegt, "Robot exploration with combinatorial auctions," en *International Conference on Intelligent Robots and Systems (IROS)*, vol. 2, Las Vegas, EUA, Octubre 2003, pp. 1957–1962.

[16] E. Bonabeau, A. Sobkowski, G. Theraulaz y J.-L. Deneubourg, "Adaptive task allocation inspired by a model of division of labor in social insects," en *Bio*

Computation and Emergent Computing, D. Lundh, B. Olsson y A. Narayanan, Eds. World Scientific, 1997, pp. 36–45.

[17] G. Bontempi, M. Birattari y H. Bersini, "Lazy learning for local modelling and control design," *International Journal of Control*, vol. 72, no. 7/8, pp. 643–658, Mayo 1999.

[18] S. Botelho y R. Alami, "Cooperative plan enhancement in multi-robot context," en *6th International Conference on Intelligent Autonomous Systems*, Venecia, Italia, Julio 2000, pp. 131–138.

[19] F. Bullo, E. Frazzoli, M. Pavone, K. Savla y S. L. Smith, "Dynamic vehicle routing for robotic systems," *Proceedings of the IEEE*, Mayo 2010.

[20] D. Busquets y R. Simmons, "Learning when to auction and when to bid," en *8th International Symposium on Distributed Autonomous Robotic Systems*, Minneapolis, EUA, Julio 2006, pp. 21–30.

[21] H. I. Calvete, C. Gale, M. J. Oliveros y B. Sanchez-Valverde, "A goal programming approach to vehicle routing problems with soft time windows," *European Journal of Operational Research*, vol. 177, no. 3, pp. 1720–1733, Marzo 2007.

[22] A. Campbell, A. Wu y R. Shumaker, "Multi-agent task allocation: Learning when to say no," en *10th annual conference on Genetic and evolutionary computation*, Atlanta, EUA, Julio 2008, pp. 201–208.

[23] Y. U. Cao, A. S. Fukunaga y A. B. Kahng, "Cooperative mobile robotics: Antecedents and directions," *Autonomous Robots*, vol. 4, no. 1, pp. 7–23, Marzo 1997.

[24] L. Chaimowicz, "Dynamic coordination of cooperative robots: A hybrid system approach," Tesis doctoral, Universidade Federal de Minas Gerais, Belo Horizonte, Brasil, Junio 2002.

[25] L. Chaimowicz, M. F. Campos y V. Kumar, "Dynamic role assignment for cooperative robots," en *International Conference on Robotics and Automation (ICRA)*, Washington, EUA, Mayo 2002, pp. 292–298.

[26] C. Chang y C. Lin, *LIBSVM: a library for support vector machines (http://www.csie.ntu.edu.tw/ cjlin/libsvm)*, 2001.

[27] K. Cheng y P. Dasgupta, "Coalition game-based distributed coverage of unknown environments by robot swarms," en 7^{th} *international joint conference on Autonomous agents and multiagent systems*, vol. 3, Estoril, Portugal, Mayo 2008, pp. 1191–1194.

[28] V. A. Cicirello y S. F. Smith, "Distributed coordination of resources via wasp-like agents," en 1^{st} *International Workshop on Radical Agent Concepts*, McLean, EUA, Enero 2002, pp. 71–80.

[29] T. J. Collett, B. A. MacDonald y B. P. Gerkey, ""player 2.0: Toward a practical robot programming framework"," en *Australasian Conference on Robotics and Automation*, Sydney, Australia, 2005.

[30] T. S. Dahl, M. Mataric y G. S. Sukhatme, "Multi-robot task allocation through vacancy chain scheduling," *Robotics and Autonomous Systems*, vol. 57, pp. 674–687, 2009.

[31] P. Dasgupta y M. Hoeing, "Task selection in multi-agent swarms using adaptive bid auctions," en 1^{st} *International Conference on Self-Adaptive and Self-Organizing Systems*, Cambridge, EUA, Julio 2007, pp. 307–310.

[32] P. Dasgupta y M. Hoeing, *Massively Multi-Agent Technology*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2008, vol. 5043, cap. Dynamic Pricing Algorithms for task Allocation in multi-agent Swarms, pp. 64–79.

[33] D. de Oliveira, P. R. Ferreira y A. L. Bazzan, "A swarm based approach for task allocation in dynamic agents organizations," en *3th International Joint Conference on Autonomous Agents and Multiagent Systems*, vol. 3, Nueva York, EUA, Julio 2004, pp. 1252–1253.

[34] E. del Acebo y J. L. de-la Rosa, "Introducing bar systems: A class of swarm intelligence optimization algorithms." en *AISB Convention Communication, Interaction and Social Intelligence*, Aberdeen, Escocia, Abril 2008, pp. 18–23.

[35] M. B. Dias, "Traderbots: A new paradigm for robust and efficient multirobot coordination in dynamic environments," Tesi Doctoral, The Robotics Institute, Carnegie Mellon University, Pittsburgh, EUA, Enero 2004.

[36] M. B. Dias y A. Stentz, "Traderbots: A market-based approach for resource, role, and task allocation in multirobot coordination," Carnegie Mellon University, Pittsburgh, EUA, Reporte técnico CMU-RI-TR-03-19, Agosto 2003.

[37] M. B. Dias y A. Stentz, "Opportunistic optimization for market-based multirobot control," en *International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, Pittsburgh, EUA, Septiembre 2002, pp. 2714–2720.

[38] M. B. Dias, R. Zlot, N. Kalra y A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257–1270, Julio 2006.

[39] M. Djadane, G. Gonclaves, T. Hsu y R. Dupas, "Dynamic vehicle routing problems under flexible time windows and fuzzy travel times," en *International Conference on Service Systems and Service Management*, Troyes, France, Octubre 2006, pp. 1519–1524.

[40] W. Drozd, "Automatically defined swarms for task allocation," en *International Conference on Intelligent Agent Technology*, San Francisco, EUA, Noviembre 2007, pp. 67–71.

[41] F. Ducatelle, A. Förster, G. D. Caro y L. Gambardella, "Task allocation in robotic swarms: new methods and comparisons," IDSIA - Dalle Molle Institute for Artificial Intelligence, Lugano, Suiza, Reporte técnico IDSIA-01-09, Enero 2009.

[42] G. Dudek, M. Jenkin y E. Milos, *Robot Teams: From Diversity to Polymorphism*. A K Peters, Ltd., 2002, cap. Taxonomies of Multirobot Task and Reward, pp. 3–35.

[43] R. E. Fan, P. H. Chen y C. J. Lin, "Working set selection using second order inforamtion for training support vector machines," *Journal of Machine Learning Research*, vol. 6, pp. 1889–1918, 2005.

[44] A. Farinelli, L. Iocchi y D. Nardi, "Multi-robot systems: A classification focused on coordination," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, pp. 2015–2028, 2004.

[45] P. R. Ferreira, F. dos Santos, A. L. Bazzan, D. Epstein y S. J. Waskow, "Robocup rescue as multiagent task allocation among teams: experiments with task interdependencies," *Autonomous Agents and Multi-Agent Systems*, vol. 20, pp. 421–443, 2010.

[46] V. Frias-Martinez, E. Sklar y S. Parsons, "Exploring auction mechanisms for role assignment in teams of autonomous robots," en *Proceedings of the 2004 RoboCup Symposium*, Lisboa, Portugal, Junio 2004, pp. 532–539.

[47] C. H. Fua y S. S. Ge, "Cobos: Cooperative backoff adaptive scheme for multi-robot task allocation," *IEEE Transactions on Robotics*, vol. 1, no. 5, pp. 1168–1178, Diciembre 2005.

[48] J. Gancet, G. Hattenberger, R. Alami y S. Lacroix, "Task planning and control for a multi-uav system: architecture and algorithms," en *International Conference on Intelligent Robots and Systems (IROS)*, Edmonton, Canadá, Agosto 2005, pp. 1017–1022.

[49] M. R. Garey y D. S. Johnson, "'strong' np-completeness results: Motivation, examples, and implications," *Journal of the ACM (JACM)*, vol. 25, no. 3, pp. 499–508, Julio 1978.

[50] M. R. Garey, D. S. Johnson y R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 117–129, Mayo 1976.

[51] M. Gendreau y C. D. Tarantilis, "Solving large-scale vehicle routing problems with time windows: The state-of-the-art," Interuniversity Research Center on Enterprise Networks, Logistic and Transportation, Montréal, Canadá, Reporte técnico CIRRELT 2010 04, Febrero 2010.

[52] B. P. Gerkey, "On multi-robot task allocation," Tesis doctoral, Center of Robotics and Embedded Systems, University of Southern California, Los Ángeles, EUA, Agosto 2003.

[53] B. P. Gerkey y M. Mataric, "Sold!: Auction methods for multi-robot coordination," *IEEE Transactions on robotics and Automation, Special Issue on Multi-robot Systems*, vol. 18, no. 5, pp. 758–768, Octubre 2002.

[54] B. P. Gerkey y M. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *International Journal of Robotics Research*, vol. 23 (9), pp. 939–954, 2004.

[55] B. P. Gerkey y M. J. Mataric, *Multi-Robot Systems: From Swarms to Intelligent Automata*. Holanda: Kluwer Academic Publishers, 2003, vol. 2, cap. A Framework for Studying Multi-Robot Task Allocation, pp. 15–26.

[56] D. Goldberg y M. J. Mataric, "Interference as a tool for designing and evaluating multi-robot controllers," en *14^{th} National Conference on Artificial Intelligence (AAAI'97)*, Providence, EUA, Julio 1997, pp. 637–642.

[57] D. Goldberg y M. J. Mataric, "Robust behavior-based control for distributed multi-robot collection tasks," USC Institute for Robotics and Intelligent Systems, Los Ángeles, EUA, Reporte técnico IRIS-00-387, 2000.

[58] D. Goldberg y M. J. Mataric, *Robot Teams: From Diversity to Polymorphism*. Ak Peters, 2002, cap. Design and Evaluation of Robust Behavior-Based Controllers for Distributed Multi-Robot Collection Tasks, pp. 315–344.

[59] J. Goossens, S. Funk y S. Baruah, "Edf scheduling on multiprocessor platforms: some (perhaps) counterintuitive observations," en *Real-Time Computing Systems and Applications Symposium*, Tokyo, Japón, Marzo 2002, pp. 321–330.

[60] J. Guerrero, G. Oliver y A. Ortiz, "On simulating behaviour-based robot colonies in the classroom," en *1st EURON Workshop on Robotics Education and Training*, Weingarden, Alemania, Julio 2001, pp. 91–98.

[61] J. Guerrero, A. Ortiz y G. Oliver, "Robocot: Simulador de colonias de robots móviles," en *VII Jornadas de Enseñanza Universitaria de la Informática*, Palma, España, Julio 2001, pp. 470–473.

[62] J. Guerrero, A. Ortiz y G. Oliver, "Experiments design using a mobile robots simulator," en *IEEE-TTTC International Conference on Automation, Quality and Testing Robotics*, Cluj-Napoca, Rumania, Mayo 2002, pp. 29–34.

[63] J. Guerrero y G. Oliver, *Artificial Research and Development*, ser. Frontiers in Artificial Intelligence and Applications. IOS press, 2003, vol. 100, cap. Multi-Robot Task Allocation Strategies Using Auction-Like Mechanisms, pp. 111–122.

[64] J. Guerrero y G. Oliver, "A multi-robot task allocation method to regulate working group sizes," en *1st EURON European Conference on Mobile Robots*, Radziejowice, Polonia, Septiembre 2003, pp. 27–32.

[65] J. Guerrero y G. Oliver, *Distributed Autonomous Robotic Systems 6.* Springer, 2004, cap. Multi-Robot Task Allocation Method for Heterogeneous Tasks with Priorities, pp. 181–190.

[66] J. Guerrero y G. Oliver, "Auction like task allocation and motion coordination strategies for multi-robot transport tasks," en *1st ICINCO International Workshop on Multi-Agent Robotic Systems*, Barcelona, Spain, Septiembre 2005, pp. 80–87.

[67] J. Guerrero y G. Oliver, "Monitoring and interference impact in multi-robot auction methods," en *Workshop on Auction Mechanisms for Robot Coordination*, Boston, EUA, Julio 2006, pp. 16–22.

[68] J. Guerrero y G. Oliver, "Physical interference impact in multi-robot task allocation auction methods," en *IEEE Workshop on Distributed Intelligent Systems*, Praga, República Checa, Junio 2006, pp. 19–24.

[69] J. Guerrero y G. Oliver, "Interference modelization in multi-robot auction methods," en *6th IFAC Symposium on Intelligent Autonomous Vehicles*, Toulouse, Francia, Septiembre 2007.

[70] J. Guerrero y G. Oliver, "A multi-robot auction method to allocate tasks with deadlines," en *7th IFAC Symposium on Intelligent Autonomous Vehicles*, Lecce, Italia, Septiembre 2010.

[71] J. Guerrero y G. Oliver, *Multi-Robot Systems, Trends and Development (En prensa).* InTech, 2010, cap. Auction and Swarm Multi-Robot Task Allocation Algorithms in Real Time Scenarios.

[72] W. S. Haboush, "Adaptive task selection using threshold-based techniques in dynamic sensor networks," Tesis doctoral, School of Computing, University of Kent, Canterbury, Reino Unido, Junio 2008.

[73] A. T. Hayes, "How many robots? group size and efficiency in collective search tasks," en *6th Int. Symp. on Distributed Autonomous Robotic Systems*, Fukuoka, Japón, Junio 2002, pp. 289–298.

[74] A. J. Ijspeert, A. Martinoli, A. Billard y L. M. Gambardella, "Collaboration through the exploitation of local interaction in autonomus collective local interactions in autonomous collective robotics: the stick pulling experiment," *Autonomous Robots*, vol. 11, no. 2, pp. 149–171, 2001.

[75] A. Jain y S. Meeran, "Deterministic job-shop scheduling: Past, present and future," *European Journal of Operational Research*, vol. 113, pp. 390–434, 1999.

[76] E. G. Jones, "Multi-robot coordination in domains with intra-path constraints," Tesis doctoral, The Robotics Institute, Carnegie Mellon University, Pittsburg, EUA, Agosto 2009.

[77] E. G. Jones, M. Dias y A. Stentz, "Learning-enhanced market-based task allocation for disaster response," Carnegie Mellon University, Pittsburgh, EUA, Reporte técnico CMU-RI-TR-06-48, Octubre 2006.

[78] E. G. Jones, M. B. Dias y A. Stentz, "Learning-enhanced market-based task allocation for oversubscribed domains," en *International Conference on Intelligent Robots and Systems (IROS)*, San Diego, EUA, Noviembre 2007, pp. 2308–2313.

[79] E. G. Jones, M. B. Dias y A. Stentz, "Time-extended multi-robot coordination for domains with intra-path constraints," en *Robotics: Science and Systems (RSS)*, Seattle, EUA, Julio 2009.

[80] I. Juutilainen, J. Röning y P. Laurinen, "A sutdy on differences in interpolation capabilities of models," en *IEEE Mid-Summer Workshop on Soft Computing in Industrial Applications*, Espoo, Finlandia, Junio 2005, pp. 202–207.

[81] N. Kalra y A. Martinoli, "A comparative study of market-based and threshold-based task allocation," en *8th International Symposium on Distributed Autonomous Robotic Systems*, Minneapolis, EUA, Julio 2006, pp. 91–102.

[82] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou y S. Shimada, "Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research," en *IEEE international conference on systems, man and cybernetics*, vol. 6, Tokyo, Japón, Octubre 1999, pp. 739–743.

[83] S. Koenig, C. Tovey, M. G. Lagoudakis, E. Markakis y D. Kempe, "The power of sequential single-item auctions for agent coordination," en *21st National Conference on Artificial Intelligence*, Boston, EUA, Julio 2006, pp. 1625–1629.

[84] M. Koes, I. Nourbakhsh y K. Sycara, "Heterogeneous multirobot coordination with spatial and temporal constraints," en *20th National Conference on Artificial Intelligence (AAAI)*, Boston, EUA, Junio 2005, pp. 1292–1297.

[85] B. Korte y J. Vygen, *Combinatorial Optimization: Theory and Algorithms*. Springer-Verlag, Berlin, 2000.

[86] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1, pp. 83–97, 1955.

[87] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, S. Koenig, C. Tovey, A. Meyerson y S. Jain, "Auction-based multi-robot routing," en *Robotics: Science and Systems (RSS)*, Boston, EUA, Junio 2005, pp. 343–350.

[88] E. L. Lawler, J. K. Lenstra, A. R. Kan y D. B. Shmoys, Eds., *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley-Interscince, 1985.

[89] A. Lein y R. T. Vaughan, "Adaptive multi-robot bucket brigade foraging," en *11st International Conference on the Simulation and Synthesis of Living Systems*, Winchester, Reino Unido, Agosto 2008, pp. 337–342.

[90] T. Lemaire, R. Alami y S. Lacroix, "A distributed tasks allocation scheme in multi-uav context," en *International Conference on Robotics and Automation (ICRA)*, vol. 4, New Orleans, EUA, Mayo 2004, pp. 3622–3627.

[91] K. Lerman y A. Galstyan, "A general methodology for mathematical analysis of multi-agent systems," USC Information Sciences, Los Ángeles, EUA, Reporte técnico ISI-TR-529, 2001.

[92] K. Lerman y A. Galstyan, "Mathematical model of foraging in a group of robots: Effect of interference," *Autonomous Robots*, vol. 13, no. 2, pp. 127–141, Septiembre 2002.

[93] J. W. Liu, *Real-Time Systems*. Prentice-Hall, 2000.

[94] W. Liu, A. Winfield, J. Sa, J. Chen y L. Dou, "Strategies for energy optimisation in swarm of foraging robots," *Lecture Notes in Computer Science*, vol. 4433, pp. 14–26, 2007.

[95] K. H. Low, W. K. Leow y M. H. Ang, "Task allocation via self-organizing swarm coalitions in distributed mobile sensor network," en *19th National Conference on Artificial Intelligence*, San José, EUA, Julio 2004, pp. 28–33.

[96] B. López, S. Suárez y J. L. de la Rosa, "Task allocation in rescue operations using combinatorial auctions," en *Artificial Intelligence Research and Development*, ser. Frontiers in Artificial Intelligence and Applications, I. Aguiló, L. Valverde y M. T. Escrig, Eds., vol. 100. IOS Press, Octubre 2003, pp. 233–243.

[97] J. Ma, J. Theiler y S. Parkins, "Accurate on-line support vector regression," *Journal Computation*, vol. 15, pp. 2683–2703, 2003.

[98] M. J. Mataric, "Designing and understanding adaptive group behavior," *Adaptive Behavior*, vol. 4, no. 1, pp. 51–80, 1995.

[99] J. McLurkin y J. Smith, *Distributed Autonomous Robotic Systems 6.* Springer, 2007, cap. Distributed Algorithms for Dispersion in Indoor Environments using a Swarm of Autonomous Mobile Robots, pp. 399–410.

[100] J. Melvin, P. Keskinocak, S. Koenig, C. Tovey y B. Y. Ozkaya, "Multi-robot routing with rewards and disjoint time windows," en *International Conference on Intelligent Robots and Systems (IROS)*, San Diego, EUA, Marzo 2007, pp. 2332–2337.

[101] S. Momen y J. Sharkey, "An ant-like task allocation model for a swarm of heterogeneous robots," en *Swarm Intelligence Algorithms and Applications Symposium*, Edinburgh, Escocia, Abril 2009, pp. 31–38.

[102] E. H. Østergaard, M. J. Mataric, y G. S. Sukhatme, "Distributed multi-robot task allocation for emergency handling," en *International Conference on Intelligent Robots and Systems (IROS)*, Hawaii, EUA, Octubre 2001, pp. 821–826.

[103] S. Paquet, "Distributed decision-making and task coordination in dynamic, uncertain and real-time multiagent environments," Tesis doctoral, Faculté de Sciences et de Génie, Université Laval Québec, Quebec, Canadá, Enero 2006.

[104] L. E. Parker, "Alliance: An architecture for fault tolerant multi-robot cooperation," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, pp. 220–240, 1998.

[105] F. Parrella, "Online support vector regression," Tesis de master, Department of Information Science, University of Genoa, Génova, Italia, Junio 2007.

[106] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems. Tercera edición.* Springer, 2008.

[107] G. Pini, A. Brutschy, M. Birattari y M. Dorigo, "Interference reduction through task partitioning in a robotic swarm, or: Don't you step on my blue suede

shoes," en *6th International Conference on Informatics in Control, Automation and Robotics*, Milán, Italia, Julio 2009, pp. 52–59.

[108] S. D. Ramchurn, M. Polukarov, A. Farinelli y C. Truong, "Coalition formation with spatial and temporal constraints," en *International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Toronto, Canadá, Mayo 2010, pp. 1181–1188.

[109] A. Rosenfeld, G. A. Kaminka y S. Kraus, *Coordination of Large Scale Multiagent Systems*. Springer-Verlag, 2006, cap. A Study of Scalability Properties in Robotic Teams, pp. 27–51.

[110] E. Sahin, S. Girgin, L. Bayindir y A. E. Turgut, *Swarm Intelligence: Introduction and Applications*, ser. Natural Computing. Springer Verlag, 2008, cap. Swarm Robotics, pp. 87–100.

[111] S. Sariel, "An integrated planning, scheduling and execution framework for multi-robot cooperation and coordination," Tesis doctoral, Istanbul Technical University, Estambul, Turquía, Junio 2007.

[112] S. Sariel, T. Balch y N. Erdogan, "Multiple traveling robot problem: A solution based on dynamic task selection and robust execution," *IEEE/ASME Transactions on Mechatronics*, vol. 14, no. 2, pp. 198–209, Abril 2009.

[113] B. Sellner y R. Simmons, "Duration prediction for proactive replanning," en *International Conference on Robotics and Automation (ICRA)*, Pasadena, EUA, Mayo 2008, pp. 1365–1371.

[114] T. Service, "Theory and algorithms for coaltion formation among heterogeneous agents," Human-Machine Teaming Laboratory, Electrical Engineering and Computer Science Department, Vanderbilt University, Nashville, EUA, Reporte técnico HMT-09-01, 2009.

[115] T. Service y J. Adams, "Coalition formation for task allocation: theory and algorithms," *Autonomous Agents and Multi-Agent Systems (En prensa)*, 2010.

[116] O. Shehory y S. Kraus, "Methods for task allocation via agent coalition formation," *Artificial Intelligence*, no. 1-2, pp. 165–200, Mayo 1998.

[117] R. G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Transactions on Computers*, vol. 29, no. 12, pp. 1104–1113, Diciembre 1980.

[118] S. L. Smith, "Task allocation and vehicle routing in dynamic environments," Tesis doctoral, Mechanical Engineering Department, University of California, Santa Barbara, EUA, Agosto 2009.

[119] S. L. Smith y F. Bullo, "The dynamic team forming problem: Throughput and delay for unbiased policies," *Systems and Control Letters*, vol. 58, pp. 709–715, 2009.

[120] A. J. Smola y B. Schölkopf, "A tutorial on support vector regression," Royal Holloway College, University of London, Reino Unido, Reporte técnico NC-TR-98-030, Octubre 1998.

[121] A. J. Smola y B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, pp. 199–222, 2004.

[122] F. Tang y L. E. Parker, "Distributed multi-robot coalitions through asymtre-d," en *International Conference on Intelligent Robots and Systems (IROS)*, Edmonton, Canadá, Agosto 2005, pp. 2606–2613.

[123] R. Thamilselvan y P. Balasubramanie, "Integrating genetic algorithm, tabu search approach for job shop scheduling," *International Journal of Computer Science and Information Security*, vol. 2, no. 1, pp. 134–139, 2009.

[124] S. Theodoridis y K. Koutroumbas, *Pattern Recognition*. Elsevier Science, 2006.

[125] G. Thomas y A. B. Williams, "Sequential auctions for heterogeneous task allocation in multiagent routing domains," en *International Conference on Systems, Man, and Cybernetics*, San Antonio, EUA, Octubre 2009, pp. 1995–2000.

[126] R. T. Vaughan, K. Støy, G. S. Sukhatme y M. J. Mataric, "Go ahead, make my day: robot conflict resolution by aggressive competition," en *6th International Conference on the Simulation of Adaptive Behavior*, París, Francia, Septiembre 2000, pp. 491–500.

[127] L. Vig, "Multi-robot coalition formation," Tesis doctoral, Gratuate School of Vanderbilt University, Nashville, EUA, Diciembre 2006.

[128] L. Vig y J. A. Adams, "Market-based multi-robot coalition formation," en *8th International Symposium on Distributed Autonomous Robotic Systems*, Minneapolis, EUA, Julio 2006, pp. 227–236.

[129] A. Viguria, "Market-based distributed task allocation methodologies applied to multi-robot exploration," Ph.D. Thesis, Departamento de ingeniería de sistemas y automática, Escuela Superior de Ingenieros, Universidad de Sevilla, Sevilla, España, Noviembre 2009.

[130] A. Viguria, I. Maza y A. Ollero, "Set: An algorithm for distributed multirobot task allocation with dynamic negotiation based on task subsets," en *International Conference on Robotics and Automation (ICRA)*, Roma, Italia, Abril 2007, pp. 3339–3344.

[131] A. Viguria, I. Maza y A. Ollero, "S+t: An algorithm for distributed multirobot task allocation based on services for improving robot cooperation," en *International Conference on Robotics and Automation (ICRA)*, Pasadena, EUA, Mayo 2008, pp. 3163–3168.

[132] S. Wei, D. Lihua, F. Hao y Z. Haiqiang, "Task allocation for multi-robot cooperative hunting behavior based on improved auction algorithm," en *27th Chinese Control Conference*, Beijing, China, Julio 2008, pp. 435–440.

[133] B. B. Werger y M. J. Mataric, "Broadcast of local eligibility for multi-target observation," en *5th International Symposium on Distributed Autonomous Robotic Systems*, Knoxville, EUA, Octubre 2000, pp. 347–356.

[134] E. Wolfstetter, "Auctions an introduction," *Jornal of Economic Surveys*, vol. 10, no. 4, pp. 367–420, Abril 1996.

[135] B. Yang, J. Geunes y W. J. O'brien, "Resource-constrained project scheduling: Past work and new directions," Department of Industrial and Systems Engineering, University of Florida, Gainesville, EUA, Reporte técnico, Abril 2001.

[136] Y. Yang, C. Zhou y Y. Tin, "Swarm robots task allocation based on response threshold model," en *4th International Conference on Autonomous Robots and Agents*, Willengton, Nueva Zelanda, Febrero 2009, pp. 171–176.

[137] L. Yu y Z. Cai, "Robot exploration mission planning based on heterogeneous interactive cultural hybrid algorithm," en *5th International Conference on Natural Computation*, Tianjin, China, Agosto 2009, pp. 583–587.

[138] D. Zhang, G. Xie, J. Yu y L. Wang, "Adaptive task assignment for multiple mobile robots via swarm intelligence approach," *Robotics and Autonomous Systems*, vol. 55, no. 7, pp. 572–588, Julio 2007.

[139] X. Zheng y S. Koenig, "Greedy approaches for solving task-allocation problems with coalitions," en *AAMAS-08 Workshop on Formal Models and Methods for Multi-Robot Systems*, Estoril, Portugal, Mayo 2008, pp. 35–40.

[140] X. Zheng, S. Koenig y C. Tovey, "Improving sequential single-item auctions," en *International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, Octubre 2006, pp. pages 2238–2244.

[141] R. M. Zlot y A. Stentz, "Market-based multirobot coordination for complex tasks," *International Journal of Robotics Research, Special Issue on the 4th*

International Conference on Field and Service Robotics, vol. 25, no. 1, pp. 73–101, Enero 2006.