TESIS DOCTORAL

GENERACIÓN AUTOMÁTICA DE TRAYECTORIAS LIBRES DE COLISIONES PARA MÚLTIPLES ROBOTS MANIPULADORES

EDUARDO FERNÁNDEZ CAMACHO

MIGUEL ANGEL RIDAO

Autor: Miguel Ángel Ridao Carlini

Director: Eduardo Fernández Camacho

TESIS DOCTORAL

GENERACIÓN AUTOMÁTICA DE TRAYECTORIAS LIBRES DE COLISIONES PARA MÚLTIPLES ROBOTS MANIPULADORES

por

Miguel Ángel Ridao Carlini

Ingeniero Industrial por la E.T.S. de I.I.

de la Universidad de Sevilla

Presentada en la

Escuela Técnica Superior de Ingenieros Industriales

de la

Universidad de Sevilla

para la obtención del

Grado de Doctor Ingeniero Industrial

Sevilla, Octubre de 1994

A Carmen y Carlos

Agradecimientos

Deseo agradecer a todas aquellas personas que, con su ayuda, han contribuido a la realización de esta tesis, mencionando especialmente

- al profesor Eduardo Fernández Camacho, bajo cuya dirección ha sido realizada esta tesis, por todo lo que de él he aprendido y por todo el apoyo prestado;
- a los profesores José Riquelme Santos y Miguel Toro Bonilla, por su inestimable colaboración en la aplicación de algoritmos evolutivos;
- al profesor José Mariano González Romano por la cuidadosa revisión del manuscrito;
- a todos los compañeros y amigos del Departamento de Ingeniería de Sistemas y Automáticapor la ayuda prestada,
- A Carmen y Carlos, por cederme generosamente gran parte de su tiempo durante estos años;
- y muy especialmente, a mis padres y hermanas, por todo el apoyo prestado a lo largo de mi vida.

Resumen

En esta tesis se estudia el problema de la planificación del movimiento para múltipes robots manipuladores que operan en un mismo entorno de trabajo estructurado con obstáculos fijos. El objetivo es especificar caminos y trayectorias para cada uno de los robots, de forma que no se produzcan colisiones entre los robots ni con los obstáculos fijos del entorno. La planificación se realizará utilizando técnicas desacopladas, es decir, se planifican los caminos de cada robot independientemente de los demás, para posteriormente, utilizando el *diagrama de coordinación*, donde se reflejan las colisiones entre los robots, obtener trayectorias libres de colisiones. La metodología propuesta permite generar de forma automática programas escritos en lenguajes de programación para robots industriales donde se encuentran implementados tanto los caminos como las sincronizaciones necesarias para evitar colisiones entre los robots.

En primer lugar se realiza una revisión de los distintos métodos propuestos en la literatura para resolver el problema de la planificación del movimiento para robots manipuladores.

A continuación se presenta la problemática asociada a los métodos de planificación desacoplada, proponiéndose un método para la obtención del diagrama de coordinación. Se estudia el problema de los diagramas de coordinación sin solución, proponiéndose modificaciones locales de los caminos mediante técnicas heurísticas para solventar este problema.

Para la generación automática de programas se introduce el concepto de *punto de sincronización*, de forma que el problema a resolver es la obtención de una secuencia de puntos de sincronización que, evitando las colisiones entre los robots, minimice el tiempo de ejecución del movimiento coordinado. Este problema de optimización se resuelve de dos formas: mediante técnicas de búsqueda en grafos del tipo A^* y mediante algoritmos evolutivos.

Estos métodos se prueban tanto en simulación como en experiencias realizadas con distintos robots manipuladores.

Se finaliza la tesis exponiendo las conclusiones más importantes y proponiendo algunas líneas de investigación futura.

Índice

Lista de figuras

Lista de tablas

Capítulo 1

Introducción

En los sistemas de fabricación industriales se está realizado un importante esfuerzo que tiene como objetivo la obtención de una producción flexible, con la que se pretende reducir costes de producción y satisfacer las diversas necesidades de los clientes. Los *sistemas de fabricación flexibles*, que están cobrando un inusitado interés, están pudiendo ser implementados en la práctica con el reciente desarrollo de la tecnología en campos como la robótica, la informática y la automática industrial.

En este ámbito, la utilización de robots manipuladores[1] cobra un creciente interés, ya que permiten incrementar la productividad, reducir los costes de producción y mejorar la calidad final de los productos. De cualquier forma, en la mayoría de los casos, los robots únicamente realizan tareas repetitivas simples, como coger y colocar piezas, cargar y descargar máquinas, tareas de pintura y soldadura entre otras. Los avances producidos en los últimos años en aspectos como la potencia de cálculo de los ordenadores y el desarrollo de sistemas sensoriales y de visión, han permitido abordar tareas más complejas, en las que se hace necesario dotar a los robots de una cierta "inteligencia".

A pesar de todo, la operación con un único robot en un entorno de trabajo limita el tipo de tareas que pueden ser ejecutadas, de forma que la utilización de varios robots operando de forma coordinada se hace indispensable para trabajos como transportar una carga de gran tamaño. Por otra parte, el uso de más de un robot también puede mejorar la eficiencia y versatilidad en tareas complejas. Así, la tarea puede ser descompuesta en varias subtareas más simples, que pueden ser realizadas en paralelo por cada uno de los robots.

---

[1] A lo largo de esta tesis se les denominará de esta forma, o simplemente *robots* o *manipuladores*.

La utilización de varios robots trabajando de forma cooperativa crea una amplia problemática que ha sido motivo de investigación dentro de numerosos campos, entre los que se puede destacar [66]:

- Métodos de interconexión entre los robots y análisis de su eficiencia.
- Diseño de lenguajes de programación (normalmente concurrentes) adaptados a este tipo de sistemas.
- Comunicaciones y protocolos para sincronizar las actividades de los distintos robots.
- Planificación de tareas y del movimiento de los robots.
- Estrategias de control.
- Integración de sensores y sistemas de visión.

Uno de los problemas que presentan un mayor interés es la planificación en sistemas con múltiples robots. Tradicionalmente, en sistemas robotizados, el problema de la planificación se descompone en varias fases, que no necesariamente se resuelven por separado:

- Planificación de procesos de ensamblaje y de tareas [44]: Consiste en dotar al robot de un sistema al que se le suministren órdenes en forma de descripciones de objetivos de alto nivel y elabore automáticamente una sucesión de acciones ejecutables que permitan al robot o sistema de robots alcanzar su objetivo.
- Planificación del movimiento: La planificación anterior da lugar a movimientos de los robots descritos de forma abstracta (normalmente movimientos de punto a punto). El objetivo es formalizar dichos movimientos, encontrando un camino geométrico y una trayectoria asociada, de forma que no se produzcan colisiones con los obstáculos fijos del entorno, ni entre los robots que realicen el movimiento simultáneamente. También es frecuente descomponer esta fase en otras dos: planificación de caminos y de trayectorias.

Un problema fundamental que surge en sistemas donde varios robots operan cooperativamente es la necesidad de establecer una sincronización entre ellos, lo que obligará a fijar canales de comunicación y protocolos. La forma en que se realicen estas conexiones darán lugar a distintos tipos de arquitecturas. En [66] se realiza un estudio de algunos de los más importantes tipos de arquitecturas.

1.1 Objetivos de la tesis

El objetivo de esta tesis es el estudio, desarrollo y implementación de algoritmos para la generación automática de programas para robots, que permitan ejecutar el movimiento coordinado de múltiples robots manipuladores. Dichos programas estarán escritos en los lenguajes de programación para robots que actualmente se encuentran en la industria.

En esta tesis, se considera un robot manipulador como una cadena cinemática abierta formada por cuerpos rígidos (*enlaces* o *elementos*) conectados por articulaciones. El final de la cadena está unida a una base fija mientras que el otro extremo está libre y unido a una herramienta. Por simplicidad, se supondrá que las articulaciones son de revolución, aunque todos los algoritmos propuestos son aplicables a otros tipos de articulaciones.

El problema global que se aborda se puede describir del siguiente modo: Dado un entorno estructurado con obstáculos fijos donde operan varios robots y conocidas las configuraciones iniciales y finales para cada uno de los robots (o un camino para cada robot), se pretende obtener un programa para cada uno de los robots, de forma que evitando las colisiones con los obstáculos fijos y con el resto de los robots, se minimice el tiempo de ejecución del movimiento conjunto de los robots. La ejecución simultánea del movimiento obliga a la elaboración de un sistema de comunicación entre los robots que permita sincronizar sus movimientos. Se ha pretendido que tanto las conexiones físicas como los protocolos sean lo más simple posibles.

Para este problema de planificación del movimiento se considerarán técnicas desacopladas, es decir, se planifican por separado los caminos para cada uno de los robots, de forma que eviten los colisionar con los obstáculos fijos del entorno. Posteriormente, se realiza un proceso de integración del movimiento conjunto de los robots, con el que se pretenden evitar las colisiones entre los distintos robots.

Las trayectorias para cada uno de los robots se obtienen en este proceso de integración, modificando la forma en que cada robot sigue su camino con el fin de evitar las colisiones. Sin embargo, como se analizará posteriormente, el partir de caminos obtenidos de forma independiente, puede llevar a problemas en que no se encuentre ninguna forma de realizar el movimiento de los robots sin que se produzcan colisiones. Para solucionar este problema, en esta tesis se proponen métodos heurísticos para la modificación de los caminos originales, que permiten encontrar una solución de forma simple.

Para la obtención de los programas anteriormente descritos, se tendrán en cuenta principalmente dos criterios, que en cierta manera son antagónicos: el tiempo de ejecución

del movimiento coordinado y la seguridad ante la evitación de colisiones entre los robots. Incidiendo sobre uno u otro criterio se proponen métodos para la obtención de dichos programas, que van desde el más conservador, que garantiza que los robots no colisionarán independientemente de la forma en que realicen su movimiento (velocidad, aceleración, instante de arranque...), hasta los métodos más efectivos, en que las colisiones entre ellos sólo se evitan si el movimiento de cada robot sigue de forma aproximada al modelo utilizado en la etapa de planificación.

Estos métodos llevan a un problema de optimización en que se pretende minimizar el tiempo de ejecución del movimiento coordinado. Este problema ha sido resuelto con dos métodos distintos: mediante algoritmos heurísticos de búsqueda en grafos y mediante algoritmos evolutivos.

Los algoritmos propuestos han sido probados para sistemas formados por robots tipo PUMA y tipo SCORBOT, aunque son aplicables a la mayor parte de los robots industriales.

1.2 Estructura de la tesis

La tesis está organizada del siguiente modo:

- En el capítulo 2 se introduce el problema de la planificación del movimiento para múltiples robots, a la vez que se realiza un repaso a las principales técnicas y algoritmos propuestos en la literatura.

- El capítulo 3 introduce la técnica que se utilizará en esta tesis para la planificación del movimiento de robots manipuladores: la *planificación desacoplada*. Se definen los conceptos necesarios para su desarrollo, y se describen algoritmos para su implementación práctica. Finalmente se proponen dos algoritmos para la modificación de caminos cuando las técnicas de planificación desacopladas no encuentran solución al problema de la coordinación de dos robots.

- En el capítulo 4 se proponen métodos para la generación automática de programas que permitan ejecutar el movimiento coordinado de varios robots. que sean fácilmente implementables mediante programas escritos en los lenguajes de programación de los robots industriales existentes en la actualidad. Asimismo se propone un procedimiento basado en el algoritmo A^* para la obtención de dicho programa, que permite optimizar el tiempo de ejecución del movimiento de los manipuladores.

- En el capítulo 5 se realiza una breve descripción de los algoritmos genéticos y las estrategias evolutivas, para finalmente proponer un nuevo método para la obtención de los programas para la coordinación de los robots basado en este tipo de algoritmos.

- El capítulo 6 muestra la aplicación de los métodos desarrollados en esta tesis al caso del movimiento coordinado de dos robots SCORBOT.

- El capítulo 7 expone las conclusiones de esta tesis y las líneas futuras de investigación.

- En el apéndice A se describe brevemente el algoritmo A^*, ampliamente utilizado a lo largo de esta tesis, así como algunas de las más interesantes propiedades y variantes de dicho algoritmo.

- El apéndice B describe el método utilizado en esta tesis para la planificación del camino de un único robot manipulador, herramienta indispensable para la planificación de varios robots si se utilizan técnicas desacopladas.

- Finalmente, el apéndice C muestra un modelo general de las trayectorias de lo un robot en el espacio articular, utilizado en los capítulos 4 y 5 para la obtención de programas óptimos.

Capítulo 2

Planificación del movimiento de múltiples robots manipuladores

2.1 Introducción

La planificación del movimiento para varios robots que operan en un entorno común de trabajo consiste en formalizar los movimientos, mediante la especificación de caminos y trayectorias, de forma que éste se realice sin colisiones con los obstáculos del entorno ni entre los distintos robots. Este problema ha sido extensamente estudiado dentro del campo de la robótica, tanto en su aplicación a robots móviles como a manipuladores. En [62] y [42] se realiza un exhaustivo repaso a la problemática que se presenta en robots móviles, así como diversas técnicas y algoritmos utilizados para abordarla.

La planificación del movimiento para manipuladores presenta una problemática específica, debido principalmente a las restricciones cinemáticas existentes entre los distintos elementos del robot. Además, el mayor número de grados de libertad (la mayor parte de las aplicaciones para robots móviles suponen movimiento en el plano) hace que el espacio de búsqueda sea mucho mayor. Esto hace que una gran parte de los métodos para robots móviles no sean directamente aplicables, y otros, aunque lo sean, necesiten un tiempo inaceptable de cálculo. Este problema se complica cuando se estudia la planificación del movimiento para varios robots manipuladores que operan en un mismo entorno, debido a que las colisiones del robot no se producen solo con los objetos del entorno, sino también con el resto de los robots. Otro problema asociado es el de la necesidad de sincronizar los movimientos de los distintos robots.

La forma en que es necesario establecer esta sincronización entre los robots depende en gran medida de la tarea que lleven a cabo dichos manipuladores. Así, en general, se pueden distinguir dos tipos de coordinación: la *coordinación débil* o *discreta* y la

coordinación fuerte o *continua*. En la primera, la sincronización solo se produce en algunos puntos concretos y determinados del camino de cada robot. Un ejemplo sería el caso de dos robots cooperando en una misma tarea de montaje de modo que para realizar una tarea común se tienen que encontrar simultáneamente en un punto determinado, aunque el movimiento hasta dicha situación lo realicen de forma independiente. La coordinación fuerte se caracteriza porque el movimiento de cada robot está cinemáticamente ligado al de los demás de forma permanente, lo que requiere un sincronización más frecuente, prácticamente continua. Un ejemplo característico es el caso de varios robots transportando una misma carga, en el que es necesario que cada robot esté siempre en contacto con la carga durante su movimiento. Es frecuente que en la bibliografía se haga referencia al primer tipo de coordinación como *movimiento simultaneo* y al segundo como *movimiento coordinado* [121].

Cuando la coordinación es débil, la arquitectura, protocolos y comunicaciones no son un problema crítico, debido a que en primer lugar los requerimientos de sincronización son infrecuentes y en segundo la cantidad de información a transmitir es escasa. Esto no ocurre con la coordinación fuerte, donde la comunicación además de ser frecuente, debe suministrar una cantidad importante de información (en el caso de transporte de una carga entre varios robots, la posición de cada uno de los ejes). Debido a este hecho, hasta los últimos años, no ha sido frecuente la aparición de sistemas capaces de realizar este tipo de coordinación.

En cualquier caso, aunque ambos tipos de coordinación hacen referencia a formas de sincronizar los robots, y en principio son independientes de la tarea que realicen, es frecuente (y así se hará en esta tesis) denominar con estos términos a los tipos de aplicaciones que en general requieren cada método de coordinación. Así, un problema será de coordinación fuerte si necesariamente hay que ligar las configuraciones entre los robots de forma continua. Por ejemplo, la evitación de colisiones de dos robots que se mueven independientemente en el mismo entorno de trabajo, es un problema de coordinación débil, mientras que dos robots transportando una carga se considera un problema de coordinación fuerte.

Por otra parte, en relación a la forma de llevar a la práctica las sincronizaciones se pueden considerar dos tipos de estrategias distintas:

- Sincronización a nivel de planificación: El estudio de las interacciones requeridas entre los distintos robots se realiza durante la planificación dando lugar a una trayectoria para cada uno de los robots. Posteriormente, cada uno de los robots seguirá esta trayectoria independientemente de los demás. Este tipo de estrategia presenta una serie de problemas. En primer lugar, es posible que se originen colisiones si algún robot no es capaz de moverse a la velocidad programada en

algún tramo del camino. Por otra parte, este tipo de trayectorias no pueden ser implementadas en un gran número lenguajes de programación de robots industriales

- Sincronización a nivel de ejecución: Durante la ejecución del movimiento se establecen unos mecanismos que permitan asegurar que el movimiento se realizará de acuerdo a lo planificado. Esta estrategia requiere en general una arquitectura más compleja.

En el resto del capítulo en primer lugar se definirá el concepto de *espacio de las configuraciones,* la forma más utilizada para representar al robot y su entorno, cuando éste es estructurado. Seguidamente se describirán brevemente algunos de los métodos existentes para planificación de un sólo robot, ya que algunos de los métodos para planificación de caminos pueden ser aplicados directamente a más de un robot, o bien pueden ser una herramienta para dicho objetivo (cuando la planificación de varios robots se descompone en la planificación por separado de cada uno de los robots).

Luego se describirán técnicas de planificación para sistemas con varios robots, mencionando en primer lugar algunos métodos para la *coordinación fuerte*, para finalmente describir de una forma más detallada las técnicas utilizadas para planificar el movimiento coordinado *débil*, que constituye el objetivo central de esta tesis.

2.2 Espacio de las configuraciones

En la planificación del movimiento de robots, el primer problema que se presenta es el de la definición de una representación adecuada tanto del robot (o los robots) como de los obstáculos del entorno. En este sentido, el *espacio de las configuraciones* [72] es la forma de representación más popular, especialmente en el problema de la planificación de caminos. En este apartado se describe el concepto de espacio de las configuraciones aplicados a sistemas con un único robot. En un apartado posterior se amplían estos conceptos al caso de múltiples robots.

Para la definición del espacio de las configuraciones se usará la notación definida en [62] y [5]. Considérese un sistema con un robot $\mathcal{A}$ y un espacio de trabajo $\mathcal{W}$ en general tridimensional, donde se sitúan obstáculos fijos. Sea un sistema de coordenadas ligado a $\mathcal{W}$ que se denominará $\mathcal{F}_{\mathcal{W}}$. Una configuración de $\mathcal{A}$ es la especificación de la posición de cada punto de $\mathcal{A}$ respecto a $\mathcal{F}_{\mathcal{W}}$. El espacio de las configuraciones de $\mathcal{A}$, que se denominará $\mathcal{C}$ se define como el conjunto de todas las posibles configuraciones de $\mathcal{A}$. Al subespacio de $\mathcal{W}$ ocupado por $\mathcal{A}$ cuando se encuentra en la configuración q se denomina $\mathcal{A}(q)$.

Sean $\mathcal{B}^i$ con $i=1,..,r$ los obstáculos estacionarios contenidos en el entorno de trabajo. Cada uno de los obstáculos tendrá una representación en el espacio de las configuraciones de la siguiente forma:

$$\mathcal{CB}^i=\{q\in \mathcal{C} / \mathcal{A}(q) \cap \mathcal{B}^i \neq \varnothing\}$$

El espacio de las configuraciones libre $\mathcal{C}_{libre}$ se define como el subespacio de $\mathcal{C}$ que no pertenece a ningún $\mathcal{CB}^i$, cualquiera que sea el valor de i.

En un robot manipulador con n grados de libertad, la forma más general de especificar una configuración q es mediante una lista de n coordenadas $q=(q_1,...,q_n)$, de manera que cada una de ellas describe el estado de cada una de las articulaciones. Con esta forma de representación, al quedar el robot definido por un punto en el espacio de las configuraciones, se transforma el problema de la planificación del movimiento de un robot, en el problema de planificar el movimiento de un punto. Sin embargo, el principal problema de este tipo de representación es la complejidad en la obtención de los obstáculos en el espacio de las configuraciones ($\mathcal{CB}^i$), mucho más en el caso de manipuladores que en el caso de robots móviles. En este último caso, es frecuente que exista una semejanza entre la representación geométrica de los obstáculos y su representación en el espacio de las configuraciones. Esta semejanza no existe en el caso de manipuladores, debido a las restricciones cinemáticas que ligan a los distintos elementos que forman un manipulador.

2.3 Planificación del movimiento para un único robot

En el problema de la planificación del movimiento es habitual descomponer el problema en la planificación de caminos y de trayectorias. En este apartado se hace una revisión de los algoritmos propuestos en ambas líneas, finalizando con algunos métodos que resuelven el problema de forma conjunta.

2.3.1 Planificación de caminos

La planificación de caminos consiste en la obtención de un camino geométrico desde una configuración inicial del robot a otra final, de forma que a lo largo de todo este camino ningún elemento del robot colisione con los obstáculos del entorno, que se supondrán estacionarios.

Dentro de los métodos que utilizan una forma de representación basada en el espacio de las configuraciones, se pueden distinguir dos grandes grupos [4],[5],[7]:

- Métodos globales: Están basados en la construcción completa del espacio de las configuraciones libres, previamente al comienzo del proceso de búsqueda. La principal ventaja de estos métodos, es que aseguran encontrar un camino si este existe, siendo su principal inconveniente el tiempo de cálculo necesario para construir dicho espacio, especialmente cuando el número de grados de libertad es elevado.

- Métodos locales: El proceso de búsqueda en el espacio de las configuraciones se realiza a partir de información heurística basada en el conocimiento parcial del espacio de las configuraciones. Al ser innecesario el cálculo previo del espacio de las configuraciones, estos métodos pueden ser mucho más rápidos en casos favorables. Sin embargo, la utilización de información parcial hace que la búsqueda sea más costosa en casos menos favorables, e incluso puede ocurrir que no encuentre una solución existente.

Dentro de los métodos globales, el cálculo de las ecuaciones exactas de los obstáculos en el espacio de las configuraciones sólo ha sido resuelto para casos particulares simples. En [76], Luh y Campbell estudian el caso de un robot Stanford, y Lumelwski en [78],[79] varios tipos de robos con dos grados de libertad, entre ellos un manipulador plano con dos articulaciones de revolución. Este último autor, para encontrar el camino una vez obtenido el espacio de las configuraciones usa al siguiente método: Mediante una línea recta une las configuraciones iniciales y finales y calcula las intersecciones de ésta con la región de colisión. El camino construido está compuesto por dos tipos de tramos, o bien son tramos pertenecientes a la línea recta o bien sigue el contorno de los obstáculos.

Sin embargo, en la mayor parte de los algoritmos existentes, el esfuerzo principal se realiza en la reducción de la complejidad en la obtención del espacio de las configuraciones. Así, una de las técnicas más utilizadas es la descomposición geométrica de la zona libre de colisiones del espacio de las configuraciones en subespacios denominados *celdas*. La relación de vecindad entre las distintas celdas es representada mediante un *grafo de conectividad,* de forma que el proceso de búsqueda del camino se reduce a encontrar una secuencia de celdas consecutivas en dicho grafo desde la celda conteniendo la configuración inicial del robot hasta la que contenga la configuración final. En dicha línea, cabe destacar los trabajos de Lozano-Pérez aplicados a robot cartesianos [71], [72] y a manipuladores en general [73], donde se calculan aproximaciones al espacio libre de configuraciones mediante discretizaciones sucesivas del rango de valores de cada grado de libertad, comenzando por la articulación más próxima a la base. Esta representación discretizada del espacio libre es compactada posteriormente agrupándola en distintas regiones dando lugar al *grafo de regiones*, donde se reflejan las conexiones entre

las distintas regiones libres. El proceso de búsqueda se realiza en este grafo mediante un algoritmo A^* optimizando la distancia al nodo objetivo. Un procedimiento muy similar es el aplicado en los trabajos de Schwartz y Sharir [105], Ward [123] y Maciejewski [80] para robots planos y Shiller [107]. En este último trabajo, se representa el espacio mediante una malla regular (*grid*), utilizándose como criterio de optimización el tiempo invertido por el robot para recorrer dicho camino, para lo cual utiliza varios modelos del movimiento del robot, desde uno simple basado en un perfil de velocidades trapezoidal hasta uno más complejo con perfil de velocidades óptimo.

Otra técnica muy utilizada para simplificar el cálculo del espacio de las configuraciones consiste en la descomposición del problema en varios subproblemas más simples. Así en [49], Hasegawa y Terasaki dividen el espacio de las configuraciones en dos subespacios: el correspondiente al brazo (tres primeros grados de libertad) y el de la mano (otros tres grados de libertad). El movimiento se divide igualmente en movimiento de partida, movimiento intermedio y movimiento de aproximación. Mientras en el movimiento intermedio solo se utiliza el subespacio asociado al brazo, en los de aproximación y partida se realiza una búsqueda combinada en los dos. La misma descomposición del movimiento se realiza en [64] para la planificación de un robot cartesiano, con la restricción adicional de considerar los movimientos de partida y de aproximación realizados sólo según el eje vertical del sistema de coordenadas ligado al espacio de trabajo, mientras que el movimiento intermedio se realiza en el plano horizontal. El método utilizado para este último movimiento es una búsqueda en un *grafo de visibilidad* [83]. Los nodos de dicho grafo son las configuraciones iniciales y finales y los vértices de los obstáculos (definidos como polígonos), mientras que los arcos son los segmentos rectilíneos que unen dos nodos y no se intersecan con ningún obstáculo. Este método ha sido ampliamente utilizado en planificación de robots móviles, pero es muy difícil de adaptar de forma general a otro tipo de manipuladores.

Más recientemente, Gupca [46] realiza una búsqueda secuencial en un número de etapas igual al número de grados de libertad del robot. En cada etapa i se dispone de un camino libre de colisiones para las i-1 articulaciones anteriores calculado previamente, realizándose la búsqueda en un espacio de dos dimensiones formado por la articulación i y el camino anteriormente indicado. El procedimiento de búsqueda está basado en campos potenciales numéricos [62]. Un procedimiento de retropropagación es necesario cuando no se encuentra una solución en una etapa. Un planteamiento similar se realiza por el mismo autor en [45], resolviéndose cada etapa mediante un procedimiento de búsqueda basado en un *grafo de visibilidad.* Del mismo modo, Gini *et al.* [40] realizan también una búsqueda secuencial, empezando con el primer grado de libertad y finalizando con el último, haciendo uso de las particularidades geométricas de un robot tipo PUMA.

Otras técnicas no utilizan como forma de representación del sistema el espacio de las configuraciones, sino el espacio tridimensional de trabajo. En esta línea cabe destacar

el trabajo de Brooks [11] que representa el espacio libre mediante *conos generalizados* y de Wu [125], que realiza una descomposición del problema similar a la comentada anteriormente en [49]. La principal aportación de este último trabajo consiste en que la obtención del camino está orientado a la tarea que realiza el robot. La idea principal consiste en obtener un camino previo que satisfaga ciertas restricciones asociadas a la tarea realizada (posición u orientación a lo largo del camino) y modificarlo en relación a las restricciones cinemáticas u obstáculos del entorno.

Frente a estos métodos de búsquedas globales, se sitúan los métodos locales. Estos últimos, al no disponer de un conocimiento global del espacio de las configuraciones, deben compensar esta falta con técnicas heurísticas potentes para dirigir la búsqueda. Kondo [58], [59], [60] descompone el espacio de las configuraciones en celdas regulares de pequeño tamaño, realizando un proceso de expansión, que busca un camino formado por celdas consecutivas desde la celda inicial a la final. La búsqueda se realiza mediante un algoritmo del tipo A^*. Esta técnica se desarrolla en el apéndice B, ya que debido a su simplicidad, a su carácter general y a los buenos resultados obtenidos en la mayoría de las situaciones, ha sido el seleccionado en esta tesis para abordar el problema de la planificación de caminos de un robot.

Recientemente, Bessière *et al.* [7] han desarrollado un método local basado en algoritmos genéticos. Básicamente, este método intenta encontrar un camino entre la configuración inicial y la final utilizando *movimientos de Manhattan* (movimientos elementales de un sólo grado de libertad, mientras los demás permanecen invariables). Caso de no aparecer ninguna solución, intenta reducir el problema en varios problemas más simples, mediante la colocación de puntos de control (*landmarks*) extendidos por el espacio de configuración, con la condición de que exista un camino conocido entre el punto inicial y el punto de control. A partir de entonces el problema pasa a ser el de encontrar un camino entre un punto de control y la configuración final utilizando el algoritmo genético anterior. Esta colocación de puntos de control se puede ver como un método para escapar de mínimos locales. Una de las ventajas que plantean los autores es su facilidad para ser implementado en paralelo.

Tal vez la técnica más clásica utilizando métodos locales es la de utilizar *campos potenciales* en el proceso de búsqueda. Consiste en suponer que el robot se encuentra bajo la influencia de un *campo potencial artificial* U, que localmente describe la estructura del espacio libre. Típicamente, la función potencial se define como la suma de un potencial de atracción, que empuja al robot hacia su configuración final, y un potencial de repulsión que aleja al robot de los obstáculos. En cada instante, se selecciona la dirección más prometedora del movimiento mediante el cálculo del gradiente de la función potencial $-\bar{\nabla}U$, evaluado en la posición del robot. El principal inconveniente de los métodos potenciales es la aparición de mínimos locales [42]. En esta línea se encuentran los trabajos de Khatib [57], que a partir de un modelo geométrico del entorno, construye la función

potencial, que prevee tanto la evitación de obstáculos como las restricciones asociadas a los límites de las articulaciones. Volpe y Khosla [122] planifican el camino en el espacio cartesiano, construyendo una función potencial que atrae el elemento terminal hacia el objetivo, y estableciendo fuerzas de repulsión entre enlaces y obstáculos. La función potencial, basada en superficies *supercuádricas*, previene la aparición de mínimos locales.

Barraquant *et al.* presentan en [4] y [5] un método potencial que puede ser catalogado como *mixto*, en el que se combinan los métodos locales y globales. A grandes rasgos, la idea consiste en precalcular una función potencial definida en el espacio del entorno de trabajo (en general tridimensional) seguido de una búsqueda guiada por dicho potencial en el espacio de las articulaciones. Para definir la función potencial, utiliza varios *puntos de control* repartidos sobre el robot, definiendo una función potencial para cada uno de ellos. Estas funciones se utilizan para formar una única función potencial mediante lo que denomina *función de arbitraje*. Durante el proceso de búsqueda se han utilizado varias técnicas para evitar mínimos locales, destacando la realización de movimientos aleatorios.

2.3.2 Planificación de trayectorias

Para que el movimiento del manipulador quede determinado, será necesario añadir a las distintas configuraciones de éste que forman parte de un camino, una componente adicional que indique el instante de tiempo en que el robot se encontrará en cada uno de los puntos del camino. Esta operación es realizada por el *planificador de trayectorias*, que deberá generar una secuencia de puntos de consigna a lo largo del tiempo para el sistema de control del manipulador.

En la mayor parte de los planificadores de trayectoria incluidos en los controladores de los robots industriales, la especificación de un camino consiste en una secuencia de puntos (en el espacio cartesiano o en el de las articulaciones) por los cuales el robot debe pasar en su movimiento, unido a algunas especificaciones geométricas del camino, como por ejemplo caminos lineales en el espacio de las articulaciones, lineales en el espacio cartesiano, circulares, etc. El trabajo más clásico en la especificación de trayectorias en el espacio de las articulaciones se debe a Paul [87], que aproxima la trayectoria entre los distintos puntos de cada articulación mediante funciones polinomiales, con la restricción de satisfacer en éstos ciertas condiciones, como continuidad en posición, velocidad y aceleración.

Más compleja es la planificación de trayectorias en el espacio cartesiano, ya que al ser las variables controladas las articulares, es necesario realizar conversiones entre ambos tipos de variables. En este sentido los trabajos más clásicos se deben a Paul [87], Taylor

[116] y Luh [74]. El problema se complica aún más si se consideran restricciones de par en cada articulación, ya que se tiene el camino especificado en coordenadas cartesianas y las restricciones de par en coordenadas articulares [34]. Para resolver este problema, la mayor parte de las aplicaciones convierten el camino a coordinadas articulares, mediante la selección de una serie de puntos en el camino cartesiano y una interpolación entre éstos mediante funciones polinomiales. Entre estas aplicaciones se encuentran los trabajos de Lin *et al.* [68],[69], Luh y Lin [77], Chang *et al.* [16] y Simon e Isik [111]. Un trabajo similar es el realizado por Davidor [20], que utilizando algoritmos genéticos intenta obtener programas de robots que se aproximen a una trayectoria dada en un tiempo óptimo, mediante la definición de un conjunto de configuraciones intermedias.

En un segundo grupo se pueden incluir los planificadores que obtienen una trayectoria óptima en cuanto a tiempo de ejecución. Estos algoritmos parten de una curva parametrizada en el espacio de las articulaciones o en el cartesiano, que representa el camino a seguir por el manipulador, y consideran la dinámica del manipulador y las restricciones de par. Estos métodos están basados en la representación de la ecuación dinámica y las restricciones del movimiento en función de un único parámetro del camino, minimizando el tiempo necesario para la ejecución del movimiento usando las ecuaciones anteriores. Entre otros, se pueden destacar los trabajos de Shin y McKay [108], [109], Pfeiffer y Johanni [91] y Slotine y Yang [112]. La complejidad en el manejo de las ecuaciones dinámicas hacen que estos métodos solo puedan ser aplicados a planificadores que operan *fuera de línea*.

Existen métodos que realizan la planificación de caminos y trayectorias de forma simultánea. En esta línea están los trabajos de Bobrow [9], que modela el camino cartesiano mediante B-splines, para lo cual parte de una solución inicial, y progresivamente va ajustando los coeficientes de dichos B-splines de acuerdo con un proceso de optimización. Galicki [35] combina técnicas basadas en campos potenciales con un problema de optimización de una determinada funcional.

En esta tesis, como se verá posteriormente, para la coordinación de manipuladores se generarán programas escritos en lenguajes de programación para robots, por lo que para la ejecución de dichos programas se utilizarán los planificadores de trayectorias incluidos por el fabricante en los controladores del robot. Teniendo en cuenta que se pretende que dichos programas sean óptimos en cuanto el tiempo de ejecución, ha sido necesario utilizar un modelo de los planificadores de trayectorias. Este modelo debe ser simple para poder ser utilizado con profusión durante el proceso de optimización, lo que ha descartado los métodos que tienen en cuenta la dinámica del sistema, y a la vez lo suficientemente exacto. Por ello, se ha elegido el método propuesto por Tondu [119] que es extensamente descrito en el apéndice C.

2.4 Planificación del movimiento para múltiples robots

La planificación del movimiento de varios robots que operan en un mismo entorno de trabajo, añade una problemática adicional a la estudiada para un sólo robot. No sólo es necesario evitar las colisiones de los robots con los obstáculos fijos, sino que además es necesario realizar la planificación de forma que no se produzcan colisiones entre ellos. Esto obliga a realizar una tarea de coordinación entre los distintos robots, lo que implicará planificar no sólo el movimiento, sino también las comunicaciones necesarias para la sincronización entre ellos. A continuación se describen distintas técnicas de planificación referidas a los dos tipos de problemas más clásicos: planificación del movimiento en que se requiere que en todo momento el movimiento de los robots esté cinemáticamente ligado, por lo que requieren coordinación fuerte, y la planificación del movimiento en que la ligadura solo afecta a las configuraciones iniciales y finales con lo que puede resolverse utilizando coordinación débil.

2.4.1 Coordinación fuerte

Las aplicaciones que requieren coordinación fuerte son aquellas en que la configuración de cada uno de los robots deberá estar continuamente ligada durante el movimiento a la de los demás. El caso que más frecuentemente se aborda en la literatura consiste en estudiar el movimiento para varios robot que transportan una carga.

En relación a la planificación de caminos, existen pocos trabajos y la mayor parte referidos a sistemas específicos. Fortune *et al.* [31] proponen un método para dos manipuladores planos tipo Standford con dos grados de libertad que se mueven en un entorno con obstáculos poligonales y que mantienen unidos los elementos terminales durante el movimiento. Para cada uno de los robots estudian las curvas límites del espacio libre, generando a partir de éstas un grafo de conectividad que utilizan para conocer si entre dos configuraciones del robot es posible realizar un movimiento libre de colisiones. El espacio libre conjunto de los dos robots lo construyen mediante la intersección de los dos espacios libres obtenidos anteriormente.

Chien *et al.* [17] abordan el problema de la planificación de caminos para dos robots planos (dos enlaces y tres articulaciones de revolución cada uno, la última situada en el elemento terminal) que transportan una carga puntual. El sistema lo modelan como una cadena cinemática cerrada con cinco enlaces lineales (los cuatro enlaces de los dos robots y la línea que une las dos bases). Debido a las restricciones holónomas del sistema sólo dos variables son independientes, existiendo para cada par de valores de la variable independiente dos configuraciones de la cadena, que denominan *configuración grande* y *configuración pequeña,* dependiendo de que el valor de un tercer ángulo esté por encima o por debajo respectivamente de un determinado valor crítico. El problema es resuelto en

el espacio de las configuraciones, descomponiendo dicho espacio en otros dos, el subespacio de las configuraciones grandes y el subespacio de las configuraciones pequeñas, de forma que existen una serie de configuraciones de transición que permiten pasar de uno a otro. La planificación la resuelven utilizando el método propuesto por Lumelwsy [78] en el que se unen las posiciones iniciales y finales con una línea recta y se busca un camino formado por tramos de dicha línea o por los bordes de los obstáculos, considerando transiciones entre los dos subespacios a través de configuraciones críticas.

Xue y Chien [127] mejoran los resultados para el mismo problema. Para evitar la obtención explícita del espacio de las configuraciones, la búsqueda se realiza solamente en determinadas líneas rectas paralelas a los ejes en los espacios de las configuraciones, seleccionadas sistemáticamente. Inicialmente el número de líneas es pequeño, y caso de no encontrarse una solución se va ampliando su número. La búsqueda se realiza utilizando un algoritmo del tipo A^*. Lógicamente, el camino obtenido no es óptimo al estar basado exclusivamente en líneas paralelas a los ejes.

Un problema similar es resuelto por Xue *et al.* [126], en que dos robots del mismo tipo que el anterior transportan una carga rectangular, formando pues una cadena cinemática de seis enlaces y por consiguiente con tres grados de libertad. Los autores presentan un método en el que en primer lugar estudian la forma de seleccionar las variables independientes, y posteriormente discretizan los valores de dos de estas variables independientes y obtienen, para cada par de estos valores, el rango de valores que puede alcanzar la tercera variable independiente. De nuevo se presenta el problema de las dos configuraciones para un mismo conjunto de valores de las variables independientes. Con los datos obtenidos construyen un *grafo de conexión* que utilizan para encontrar un camino.

Finalmente en [18] se generaliza la descomposición en dos subespacios descrita anteriormente a cadenas cinemáticas de n enlaces en el espacio tridimensional, donde pueden existir tanto articulaciones de revolución como prismáticas.

Otros trabajos están enfocados a la realización de otro tipo de tareas distintas. Tabarah *et al.* [115] estudian la planificación de trayectorias en tareas desarrolladas por dos robots sobre un objeto, en las que uno de ellos sostiene el objeto, mientras que en el otro robot es donde está montada la herramienta. Dado el camino a seguir por la herramienta relativo al objeto, se propone un algoritmo para obtener las *trayectorias conjugadas* de cada robot, de forma que se optimice el tiempo de ejecución de la operación.

2.4.2 Coordinación débil

Dada una configuración inicial y una final para cada uno de los robots, se pretende encontrar un camino o trayectoria para cada uno de los robots de forma que no colisionen ni con los obstáculos del entorno ni con el resto de los robots. Básicamente se pueden indicar tres metodologías. La *planificación centralizada* intenta resolver el problema de una forma completa, esto es, utiliza un espacio de trabajo cuya dimensión es la suma del número de grados de libertad de cada uno de los robots a planificar. La gran dimensión de este espacio de trabajo dificulta su aplicación práctica. En la *planificación con prioridad* se planifica el movimiento de cada robot de forma individual y secuencial, es decir, de forma que un robot deberá evitar colisiones con los obstáculos fijos y con los robots que le precedan en la secuencia. Finalmente, en la *planificación desacoplada* los caminos o trayectorias de los distintos robots se obtienen por separado y de forma independiente, por lo que es necesario posteriormente estudiar las interacciones entre los robots, y si es necesario, evitar las colisiones modificando dichos caminos o trayectorias. Estos dos últimos métodos presentan la ventaja de reducir la complejidad de problema, pero también el inconveniente de no ser completos, es decir, pueden no encontrar solución aun cuando ésta exista.

2.4.2.1 Planificación centralizada

Como extensión al concepto de espacio de las configuraciones, para el caso de sistemas con más de un robot, se puede definir el concepto de *espacio compuesto de las configuraciones* [62]. Sea $\mathcal{A}=(\mathcal{A}^1,...,\mathcal{A}^R)$ un sistema formado por R manipuladores. Como cada robot se mueve independientemente de los otros, una configuración en este sistema se define como un vector formado por una configuración de cada uno de los robots, es decir $q=(q^1,...,q^R)$, donde $q^i=(q_1{}^i,...,q_n{}^i)$ es una configuración para el i-ésimo manipulador. Por tanto, si $\mathcal{C}_i$ es el espacio de las configuraciones del robot i, se define el espacio compuesto de las configuraciones como $\mathcal{C}=(\mathcal{C}_1 \times ... \times \mathcal{C}_R)$.

En el espacio de las configuraciones se definen dos tipos de obstáculos. En primer lugar los producidos por la interacción entre un robot $\mathcal{A}^i$ y un obstáculo fijo $\mathcal{B}^j$. La representación de este tipo de obstáculos en el espacio compuesto de las configuraciones será:

$$\mathcal{CB}^{ij}=\{q=(q^1,...,q^i,...,q^R)\in \mathcal{C} \,/\, \mathcal{A}^i(q^i) \cap \mathcal{B}^j \neq \varnothing\}$$

El segundo tipo de obstáculos se debe a las colisiones que se pueden dar entre dos robots, y su representación será:

$$\mathcal{CA}^{ij}=\{q=(q^1,...,q^i,...,q^j...,q^R)\in \mathcal{C} / \mathcal{A}^i(q^i) \cap \mathcal{A}^j(q^j) \neq\varnothing\}$$

Del mismo modo, el espacio libre se define como el conjunto de puntos del espacio compuesto de las configuraciones que no pertenece ni a $\mathcal{CB}^{ij}$ ni a $\mathcal{CA}^{ij}$ para cualquier valor de i y de j. Por tanto el problema de la planificación del movimiento de múltiples robots se reduce a encontrar un camino en el espacio compuesto de las configuraciones que quede incluido dentro del espacio libre.

El planteamiento del problema de esta forma hace que teóricamente sean aplicables la mayor parte de los métodos estudiados para la planificación de caminos para un sólo robot; sin embargo, en la práctica, la elevada dimensión del espacio de búsqueda hace que en la mayoría de los casos el problema sea inabordable desde un punto de vista general, aunque se han propuesto algunas soluciones para casos particulares. Así por ejemplo Schwartz y Sharir [105] proponen un método basado en una descomposición en celdas para la planificación de dos discos que se mueven en el plano entre obstáculos poligonales. Barraquand *et al.* [5] resuelven el problema del movimiento de varios discos en un entorno plano mediante un método basado en campos potenciales. Para escapar de los mínimos locales los autores proponen la realización de *movimientos restringidos*. Dichos movimientos consisten en hacer que una coordenada del espacio de trabajo aumente o disminuya hasta que se alcance un punto de silla del potencial.

Para robots manipuladores también se han propuesto algunos métodos, la mayor parte aplicables a robots planos con estructuras concretas. Fortune *et al.* [31] proponen un método para el movimiento independiente de dos manipuladores planos tipo Standford con dos grados de libertad y obstáculos poligonales, como extensión al comentado anteriormente en el que los robots transportan una carga puntual. Sean $\mathcal{A}_1$ y $\mathcal{A}_2$ los dos robots. Fijando la posición de uno de los robots (Por ejemplo $\mathcal{A}_1$) se puede obtener el espacio libre para el segundo robot, tomando $\mathcal{A}_1$ como obstáculo (como se comentó anteriormente, la representación del espacio libre la realizan los autores mediante un grafo de conectividad que representa los límites de la zona libre). Para representar el espacio conjunto, particionan las configuraciones de $\mathcal{A}_1$ de forma que para todas las configuraciones de dicho robot que pertenezcan a una misma partición, el grafo de conectividad de $\mathcal{A}_2$ sea cualitativamente igual. Las transiciones entre particiones se realizan cuando la configuración de $\mathcal{A}_2$ pertenezca a una *curva crítica*. Los autores demuestran que la complejidad del algoritmo es $O(n^3)$ donde n es el número de vértices de los obstáculos poligonales.

Sharir y Sifrony [106] proponen un método similar, aunque más general, que puede ser aplicado a otros tipo de robots planos (los autores aplican al método a dos robots Standford y a dos discos), siendo su complejidad $O(n^2)$. La idea consiste en obtener por

separado el espacio libre de cada uno de los robots y realizar una descomposición exacta en celdas de forma que, geométricamente, cada una de ellas esté limitada por un número constante de curvas de grado algebraico limitado. Relacionando las celdas de cada uno de los espacios libres se obtiene el espacio en cuatro dimensiones correspondiente al movimiento simultáneo de ambos robots.

Recientemente, Conte *et al.* [19] han propuesto un método general para la obtención del espacio compuesto de las configuraciones, mediante la utilización de un algoritmo jerárquico para la descomposición aproximada del espacio en celdas. Básicamente la idea consiste en encontrar un camino en el espacio discretizado con las celdas de mayor tamaño, para posteriormente refinar el camino discretizando en un segundo nivel las celdas pertenecientes al camino inicial. En ambos casos, para encontrar los caminos utilizan una función de navegación simple (*wavefrot expansion*). El algoritmo utiliza retropropagación en caso de no encontrar un camino en alguna de las etapas.

Freund y Hoyer [32],[33] hacen un planteamiento *local* del problema. Presentan un sistema de control que coordina el movimiento de robots planos con una articulación prismática y otra de revolución. Para evitar colisiones calculan en tiempo real los valores de las distintas articulaciones de cada uno de los robots de forma que se satisfagan las restricciones geométricas. Para ello proponen un coordinador jerárquico que considerando el estado de cada articulación y las restricciones geométricas, proporciona las referencias para los sistemas de control de cada robot. Esta solución, que presenta la ventaja de poder operar en tiempo real, tiene el inconveniente de no abordar el problema de los *puntos muertos*, debido a que solamente se considera el estado instantáneo de cada robot.

Otro método local es el propuesto por Tournassoud [120], donde se describe una técnica para coordinar varios objetos móviles, pudiéndo ser aplicado a manipuladores. La idea es definir planos de separación (*tangent separating hyperplanes*) que en cada instante aseguren que dos de los objetos queden en lados opuestos. De nuevo la no consideración de los *puntos muertos* es el principal inconveniente.

2.4.2.2 Planificación con prioridad

Una técnica menos usada en la bibliografía es la planificación con prioridad. El movimiento de los robots se planifica en un orden determinado, uno cada vez, de forma que al calcular el movimiento del robot i se consideran los obstáculos fijos del entorno, mientras que los i-1 robots que han sido planificados con anterioridad se consideran como obstáculos móviles.

De esta forma, la planificación del primer robot $\mathcal{A}_1$ se realiza en el espacio de las configuraciones $\mathcal{C}_1$ de dicho robot. El resto de los robots es más complicado al tener que

considerar obstáculos móviles. En el método propuesto por Erdmann y Lozano-Pérez [25] dicha planificación se realiza en el espacio $\mathcal{CT}$, espacio obtenido a partir del espacio de las configuraciones, al que se añade una dimensión adicional, el tiempo. Otra posibilidad sería planificar el movimiento a partir del segundo robot en un espacio de dimensión dim($\mathcal{C}$)+1, pero utilizando como dimensión adicional, en lugar del tiempo, un camino en el espacio compuesto de las configuraciones de los robots anteriores. Es decir, al planificar el camino del robot i se toma como dimensión adicional el camino obtenido en la etapa anterior al planificar el robot i-1.

Una de las dificultades que presenta el método es la determinación del orden en que se realizará la planificación. La más simple de ellas consiste en establecer una ordenación de forma aleatoria, aunque es preferible obtenerlas a partir de datos que tengan en cuenta las características del problema, como por ejemplo realizar primero la planificación de los robots de mayor tamaño [62]. Buckley [12], para resolver el problema de objetos que se trasladan en el plano, propone ordenarlos de forma que se maximice el número de robots que puedan moverse desde la configuración inicial a la final siguiendo una línea recta. Para ello construye el *grafo de prioridad*, cuyos nodos son los robots, de forma que existirá un arco desde el robot i al j si el robot i tiene prioridad sobre el j. Las situaciones conflictivas se dan cuando existen ciclos en el grafo, lo que implica que alguno de los robots no puede moverse en línea recta. Para evitar ciclos se eliminan los nodos necesarios del grafo (aquellos que al desaparecer eliminen un mayor número de ciclos). Los robots eliminados requieren un camino más complejo para cuya obtención propone un algoritmo basado en un grafo de visibilidad.

2.4.2.3 Planificación desacoplada

Por su eficiencia, la planificación desacoplada ha sido la más utilizada en los algoritmos descritos en la bibliografía. Como se comentó anteriormente, se basa en la planificación por separado del movimiento para cada uno de los robots obteniendo caminos o trayectorias que no colisionan con los obstáculos fijos del entorno. Evidentemente, estos planes no aseguran que no existan colisiones entre los robots. Un posterior análisis de estas colisiones se traduce en modificaciones de las trayectorias, o en algunos casos de los caminos, previamente calculados. El principal inconveniente de este método consiste en no ser *completo*, incluso en problemas donde existe solución, el procedimiento puede no encontrarla. Sin embargo presenta también considerables ventajas, siendo la principal de ella su simplicidad, que hace que el tiempo de cálculo sea considerablemente menor que en las metodologías anteriores. Otra ventaja es la posibilidad de poder utilizar para planificar el movimiento los métodos para un sólo robot, problema mucho más estudiado en la bibliografía que la planificación simultánea de múltiples robots.

Esta técnica de planificación fue propuesta inicialmente por Kant y Zucker [55],[56] en su aplicación a robots móviles. Los autores obtienen caminos para cada robot, realizando la coordinación estableciendo tiempos de paso por algunos puntos críticos. Posteriores variaciones del método han llevado a algoritmos capaces de realizar una coordinación más compleja, incluyendo modificaciones del camino. Una de ellas es la propuesta por Liu *et al.* [70] donde las restricciones impuestas por el movimiento simultáneo de los dos robots quedan especificadas mediante una red de Petri. Cuando es necesario, propone una modificación de los caminos, consistente en apartar uno de los robots de su camino original para dejar paso al otro (basado en esta idea, en esta tesis también se desarrolla un método de modificación de caminos para manipuladores).

Tal vez el método más referenciado en la bibliografía aplicado a manipuladores es el descrito por Lee y Lee en [63], donde se estudia el movimiento coordinado de dos robots, modelados mediante esferas (en realidad sólo se modela el elemento terminal) que se mueven en línea recta en el espacio cartesiano. La trayectoria de uno de los robots queda fijada, alterándose en caso de colisión la trayectoria del segundo de los robots (el camino de éste no se altera). La forma de detectar colisiones es mediante la construcción del *mapa de colisiones*, que es un gráfico bidimensional en el que se representa el camino del segundo robot frente al tiempo. En este gráfico se incluye la *región de colisión*, puntos de dicho mapa que provocan colisión con la trayectoria preestablecida del primer robot (ver Figura 2.1). La idea fundamental es modificar la trayectoria del segundo robot para que su representación en el mapa de colisiones no corte la región de colisión. Por simplicidad, la región de colisión es aproximada mediante rectángulos (*rectángulo de colisión*). Se proponen varias estrategias para evitar las colisiones, que van desde retrasar el instante en que el segundo robot empieza su movimiento hasta reducciones de velocidad en algunos puntos determinados del camino.

Un trabajo reciente realizado por Chang *et al.* [15] ha eliminado algunas de las limitaciones del método anterior. Así, el método lo extiende a robots modelados por poliedros, proponiendo además un método eficiente para la obtención de la región de colisión, calculando sólo parte del contorno de dicha región, lo que hace innecesario su aproximación mediante rectángulos.

Otro de los trabajos clásicos en este campo es el realizado por O'Donnell y Lozano-Pérez [85], donde de nuevo se estudia el movimiento de dos robots. Parten de un camino previamente calculado compuesto por tramos rectos. El estudio de las colisiones lo realizan en un gráfico de dos dimensiones, donde se representan en cada eje los caminos de los dos robots y en el que se reflejan las situaciones que provocan colisiones entre los robots. Este gráfico lo denominan los autores *diagrama de conclusión de tareas*[2]. El movimiento

[2] Este tipo de diagrama ha recibido en la literatura una gran cantidad de nombres distintos. En esta tesis se ha optado por *espacio de coordinación*

coordinado de los robots viene representado mediante una curva monótona creciente que una la esquina inferior izquierda del rectángulo con la superior derecha. Utilizando el concepto de *clausura-SW* [95] (que posteriormente se analizará en detalle) proponen un método que elimina las situaciones en que se alcanza un *punto muerto* (*deadlock*), donde cada uno de los robots está bloqueado por el otro. La eliminación de estos puntos les permite utilizar métodos de búsqueda *irrevocables* para la obtención de una solución coordinada.

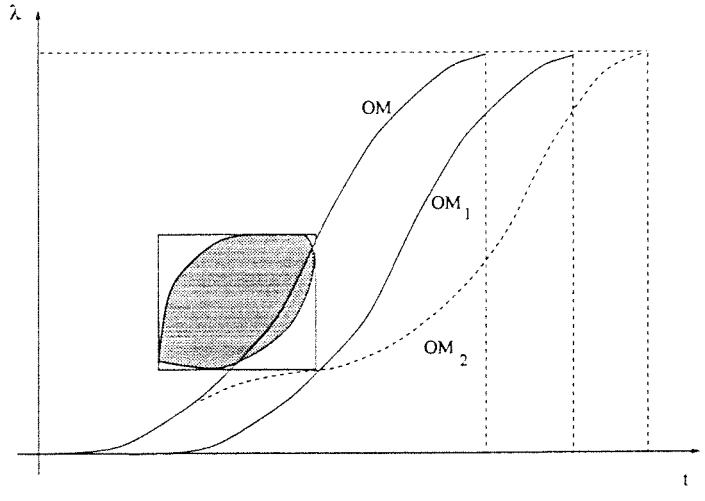

Figura 2.1 Mapa de colisión del método de Lee y Lee. OM es la trayectoria inicial con colisión. OM_1 es la obtenida retrasando el instante de comienzo y OM_2 se obtiene decelerando en un punto intermedio del camino.

En esta misma línea se localizan los trabajos de Basta *et al.* [6], que también modelan el robot mediante una esfera que se corresponde con el elemento terminal, y consideran movimientos en línea recta. El algoritmo, orientado a detección de colisiones, parte de una trayectoria para cada uno de los robots. La detección de colisiones se realiza en el espacio de coordinación en dos niveles de detalle. En el nivel más alto, la región de colisión se sustituye por rectángulos que la engloben. Si la representación de las trayectorias cruza este rectángulo puede que exista colisión, por lo que se analizan los caminos dentro del rectángulo mediante un algoritmo iterativo, en el que se utiliza el perfil exacto del la región de colisión. En caso de existir colisiones, los autores indican algunas posibilidades para la modificación de los caminos o trayectorias.

El concepto de espacio de coordinación también aparece en el trabajo de Bien y Lee [8], donde presentan un algoritmo para obtener las trayectorias de dos robots, de modo que se minimice el tiempo de ejecución del movimiento. Este método está restringido a

espacios de coordinación con una única región de colisión. El método se basa en la obtención previa de las trayectoria óptimas para cada robot (en la página 15 se referencian algunos de estos algoritmos). El método para evitar colisiones entre los robots consiste en fijar la trayectoria de uno de los robots y modificar la del segundo. La forma de modificar la trayectoria que proponen consiste en retrasar el comienzo del movimiento de este último robot, hasta conseguir que la representación del movimiento de los dos robots en el espacio de coordinación sea tangente a la región de colisión (ver Figura 2.2). Para ello utilizan un método muy similar al de Lee y Lee [63]. Fijan la trayectoria del robot 1 y transforman el espacio de coordinación en el espacio representado por el camino del robot 2 frente al tiempo. En dicho espacio calculan el retraso mínimo para que la trayectoria del robot 2 no produzca colisiones con la del robot 1.

Casi simultáneamente, Shing y Zheng [110] proponen un método muy similar, donde se realiza una demostración teórica de que la solución es óptima siempre que la región de colisión cumpla unas determinadas condiciones geométricas.

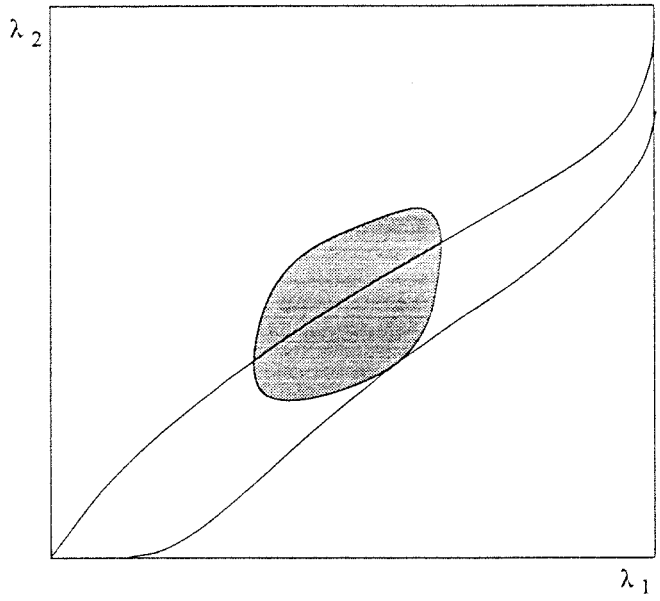

Figura 2.2 Método de Bien. Se consigue que la representación de las trayectorias de los robots sea tangente a la región de colisión.

J. Lee en [65] amplia los resultados de [8] a situaciones en las que existe más de una región de colisión. El autor plantea la coordinación como un problema de búsqueda de caminos en el espacio tridimensional formado por los caminos de cada uno de los robots y la velocidad de uno de ellos (nótese que la velocidad del otro robot viene determinada

por la velocidad del primer robot y la pendiente de la curva en el espacio de coordinación). El camino es encontrado discretizando el espacio de coordinación y aplicando técnicas de programación dinámica, minimizando el tiempo total de realización del movimiento coordinado.

Una técnica más conservadora es la aplicada por Ward [123] para planificar el movimiento coordinado de dos robots tipo SCARA. El problema queda reducido a la planificación de los dos primeros grados de libertad, con lo que cada robot se puede modelar por dos polígonos. El autor obtiene el área barrida por cada uno de los robots en su movimiento, y considera que es libre de colisión cuando dichas áreas no se intersecan. Ward propone esta técnica dentro de un método más completo de planificación de tareas, por tanto, en caso de producirse colisión modifica la secuencia de tareas. La dificultad de este método es su difícil adaptación a otro tipo de robot más complejo, que supondría en general calcular el volumen barrido por los robots. Otro inconveniente es su excesivo conservadurismo (nótese que cualquier problema que tuviera al menos una región de colisión en el espacio de coordinación supondría colisión usando esta técnica).

En la mayor parte de estos métodos (salvo [85] y [123]) la planificación obtenida es difícil llevarla a la práctica, ya que proporcionan una trayectoria para cada uno de los robots. Caben entonces dos posibilidades: o bien se siguen independientemente por cada uno de los robots, lo que significaría que cualquier error en el seguimiento de la trayectoria puede llevar a una colisión, o bien se trata el problema como de coordinación fuerte, lo que supone la necesidad de una arquitectura más compleja a la estrictamente necesaria.

El trabajo presentado en esta tesis puede enmarcarse dentro de las técnicas de planificación desacoplada basadas en el uso del espacio de coordinación. La gran ventaja que presentan los métodos propuestos es la facilidad de ser llevada a la práctica, ya que por un lado las trayectorias que se proporcionan están descritas mediante lenguajes de programación para robots, y por otro el problema de la sincronización se trata como de coordinación débil, lo que supone arquitecturas mucho más simples.

Capítulo 3

Planificación Desacoplada

3.1 Introducción

En este capítulo se describe una de las técnicas más frecuentemente utilizadas para realizar la planificación del movimiento de varios manipuladores que operan en el mismo espacio de trabajo: la planificación desacoplada. Como se describió en el capítulo anterior, la filosofía del método consiste en descomponer el problema en dos subproblemas más simples: en primer lugar la obtención para cada uno de los manipuladores, y de forma independiente, de un camino libre de colisiones con los obstáculos fijos del entorno. El segundo subproblema consiste en obtener una trayectoria para cada uno de los robots, de forma que no se produzcan colisiones entre ellos. La principal ventaja del método consiste en la reducción de la complejidad del problema respecto a métodos que intentan resolver los dos subproblemas simultáneamente. Su principal inconveniente radica en la posibilidad de que el método no encuentre solución, aun cuando ésta exista.

Esta tesis se centra en el segundo de los subproblemas. Así, en este capítulo se introducen los conceptos generales que van a permitir abordarlo, partiendo de la base de que se dispone de un camino para cada uno de los robots libre de obstáculos en relación con los obstáculos del entorno. También se exponen distintos algoritmos necesarios para su resolución, y finalmente, para los casos en que el método no encuentra solución, se proponen distintas modificaciones a los caminos iniciales.

3.2 Conceptos generales

Sea un sistema formado por R robots que comparten un mismo espacio de trabajo. El número de grados de libertad del robot j vendrá dada por N_j. Supóngase que se dispone de un camino libre de colisiones con los obstáculos del entorno para cada uno de los robots. Dichos caminos vendrán representados mediante curvas paramétricas en el espacio de las

configuraciones (por simplicidad, en esta tesis se considerará que las articulaciones son de revolución, aunque los métodos aquí propuestos serían igualmente válidos para otros tipos de articulaciones). Sea la siguiente familia de funciones paramétricas:

$$P^j=\phi^j(\lambda) \quad 0\leq\lambda\leq\lambda^j_{\max} \quad \text{con} \quad \phi^j:\mathbb{R}\rightarrow\mathbb{R}^{N_j} \quad \text{y} \quad 1\leq j\leq R \tag{3.1}$$

donde λ se define como la distancia a lo largo de cada camino, de forma que $\lambda=0$ se corresponde con la posición inicial de cada robot y $\lambda=\lambda^j_{\max}$ representa la posición final del robot *j*.

Se define el *espacio de coordinación* (*EC*) [8], [55] como una región en $\mathbb{R}^R$ cuyos puntos satisfacen la siguiente condición:

$$EC=\left\{(\lambda^1,..,\lambda^R) \;/\; 0\leq\lambda^j\leq\lambda^j_{\max} \text{ con } 1\leq j\leq R\right\} \tag{3.2}$$

Un punto en el espacio de coordinación $(\lambda^1,..,\lambda^R)$ representa una configuración determinada del sistema de robots, de forma que el estado de las articulaciones de cada uno de los robots vendrá dado por la función $\phi^j(\lambda^j)$. Lógicamente el punto (0,..0) describe el estado en que todos los robots se encuentran en la posición inicial de los caminos, y por tanto representa el estado inicial del sistema de robots, mientras que el punto $(\lambda^1_{\max},...,\lambda^R_{\max})$ se corresponde con la posición final del sistema.

Se define la *región de colisión* (*RC*) [8], [55] como el conjunto de puntos del espacio de coordinación en los que se produce colisión entre al menos dos de los robots. En general, esta región de colisión estará formada por varias zonas conexas con perfiles complejos. En la Figura 3.1 se muestra el espacio de coordinación con la región de colisión para el caso de dos robots simples.

Se denomina *camino coordinado* o *de coordinación* (*CC*) [8] a cualquier camino continuo en el espacio *EC* que cumpla las dos siguientes condiciones:

- Debe comenzar en (0,..0) y terminar en $(\lambda^1_{\max},...,\lambda^R_{\max})$
- $d\lambda^i/d\lambda^j\geq 0 \quad \forall i, j$

Esta segunda condición asegura que los caminos de cada uno de los robots no puedan recorrerse en sentido inverso.

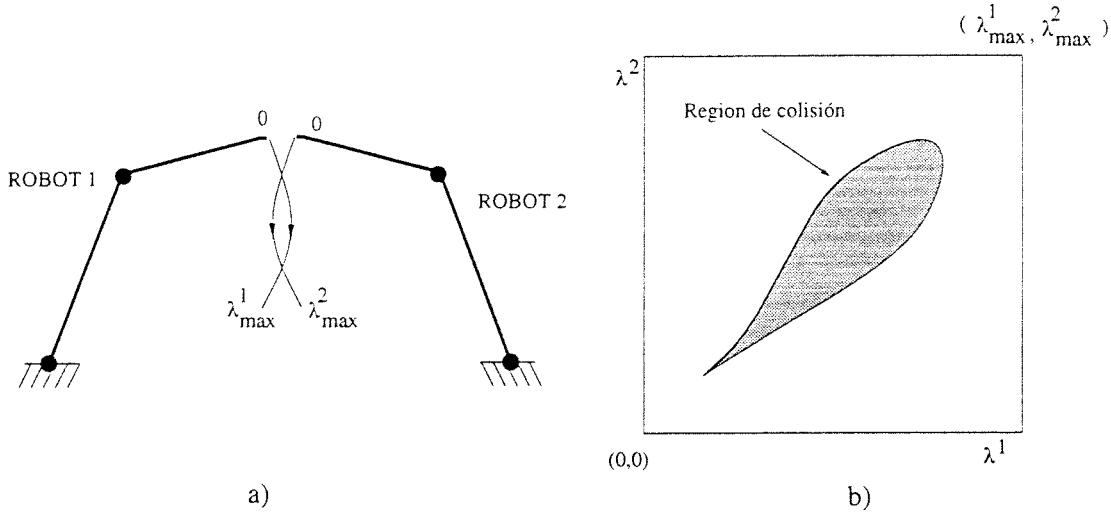

Figura 3.1 a) Sistema de dos robots con sus caminos. b) Espacio de coordinación y región de colisión correspondiente al sistema anterior.

Dada una trayectoria para cada uno de los robots, la ejecución coordinada de éstas dará lugar a un camino de coordinación. Evidentemente, un camino de coordinación no da lugar de forma unívoca a trayectorias para cada robot, simplemente refleja una información sobre la velocidad relativa entre los distintos robots.

Si un camino coordinado no atraviesa la zona de colisiones se dice que es un *camino coordinado libre de colisiones*. Por tanto *CC* estará libre de colisiones si:

$$\forall x \in CC \Rightarrow x \notin RC \tag{3.3}$$

Con estas definiciones, el problema que se aborda en este tesis puede describirse de la siguiente manera: Dado un camino para cada robot del sistema que no colisione con los objetos del entorno, encontrar trayectorias asociadas a cada uno de ellos, de forma que den lugar a un camino de coordinación libre de colisiones.

Como se verá posteriormente, los métodos propuestos están basados en procesos de búsqueda en el espacio de coordinación. Por tanto, y para reducir este espacio de trabajo, en primer lugar se realizará una discretización de los caminos, de modo que cada uno de ellos quede dividido en intervalos iguales. Así, los intervalos del robot j quedarán numerados desde 1 hasta max_j, de modo que δ^j_k representará el k-ésimo intervalo

correspondiente al camino del robot j. De esta forma, el camino discretizado de dicho robot puede describirse como el siguiente conjunto de intervalos:

$$\Omega_j=\left\{\delta_k^j \;/\; 1\leq k\leq max_j\right\} \tag{3.4}$$

Se define una *celda* C como el subespacio en $\mathbb{R}^R$ formado por un intervalo de cada uno de los R manipuladores:

$$C=\delta_{n_1}^1\times\ldots\times\delta_{n_R}^R \;\;/\;\; \delta_{n_j}^j\in\Omega_j \tag{3.5}$$

Una celda está *libre de colisiones* si ningún punto de ella pertenece a la región de colisión, es decir:

$$\forall\lambda^j\in\delta_{n_j}^j \text{ con } 1\leq j\leq R \Rightarrow (\lambda^1,\ldots,\lambda^R)\notin RC \tag{3.6}$$

Por simplicidad, se hará referencia a un intervalo utilizando su número de orden en la sucesión, de modo que $\Omega_j=\{k \;/\; 0\leq k\leq max_j\}$. Es importante establecer la correspondencia entre un intervalo de un camino y las posiciones de cada una de las articulaciones del robot. Para esto, se asociará a cada intervalo k un punto λ_k contenido en él que representará a dicho intervalo (normalmente se considerará el punto intermedio). El convenio utilizado en esta tesis será denominar $\hat{k}$ al vector que define las posiciones de cada una de las articulaciones del robot en el punto representativo del intervalo k, es decir, para el robot i:

$$\hat{k}=\phi^i(\lambda_k) \tag{3.7}$$

Con estas consideraciones, una celda se define como $C=(n_1,\ldots,n_R)$. Debido a la utilización de estos caminos discretizados, el espacio de coordinación se transforma en una matriz de dimensión R compuesta por celdas, a la que se denominará *diagrama de coordinación* [62]. En este diagrama se denominará $\mathbb{C}_0$ a la celda formada con el primer intervalo de cada uno de los caminos de los manipuladores, mientras que $\mathbb{C}_{max}$ es la celda compuesta por los

últimos intervalos de cada camino, es decir las celdas (1,...,1) y $(max_1,...,max_R)$ respectivamente.

Por extensión, se define un *camino coordinado* en un diagrama de coordinación como una secuencia monótona creciente de celdas consecutivas desde $\mathbb{C}_0$ hasta $\mathbb{C}_{max}$, es decir:

$$CC=\left\{C^i=(n_1^i,...,n_R^i)\ 0\le i\le L\right\} \quad \text{con } C^0=\mathbb{C}_0\ ,\ C^L=\mathbb{C}_{max} \text{ y}$$

$$n_j^{i+1}=\begin{cases} n_j^i+1 & \text{para al menos un valor de } j \\ n_j^i & \text{para los demás valores de } j \end{cases} \tag{3.8}$$

Finalmente, un camino de coordinación libre de colisiones será aquel formado exclusivamente por celdas libres.

3.3 Obtención del diagrama de coordinación

El primer problema que es necesario resolver para la utilización eficiente de técnicas de planificación desacopladas consiste en obtener el diagrama de coordinación. En esta línea es necesario resolver dos problemas: la evaluación del estado de las celdas y el diseño de algoritmos eficientes para la construcción de dicho diagrama.

3.3.1 Cálculo del estado de las celdas

Tal vez el aspecto más crítico en la obtención del diagrama de coordinación es disponer de un algoritmo capaz de calcular los estados de colisión entre dos o más robots, para lo cual en primer lugar es necesario obtener una representación de los volúmenes geométricos que forman cada uno de los manipuladores. El grado de precisión en la representación es un aspecto crítico, ya que una representación muy exacta lleva a algoritmos de detección de colisiones que requieren un gran esfuerzo de cálculo. Por este motivo, en la mayoría de los casos se suele recurrir a representaciones aproximadas.

En el problema de la detección de colisiones orientado a la obtención del diagrama de coordinación se establecen dos cuestiones. En primer lugar, la detección de colisiones entre robots, es decir, dada una configuración del sistema de robots, decidir si existe colisión entre alguno de ellos. El segundo está ligado a la discretización realizada en los

caminos de los robots. Consiste en decidir si una celda está libre de colisiones, para lo cual es necesario realizar la anterior comprobación para un rango de configuraciones. La detección de colisiones no ha sido un tema principal de investigación en esta tesis, sino una herramienta. Por tanto se ha optado por utilizar técnicas que de una forma simple den lugar a resultados aceptables en cuanto a tiempo de cálculo y precisión. A continuación se estudian de forma breve las soluciones adoptadas para cada uno de los problemas.

3.3.1.1 Detección de colisiones

Una gran parte de los módulos de detección de colisiones implementados en aplicaciones reales están basados en el cálculo de distancia entre los distintos sólidos que componen el sistema. La forma de representación de los sólidos juega un papel primordial en este tipo de algoritmos. Tal vez la forma más popular de representación ha sido mediante poliedros convexos, de forma que el cálculo de distancias se realiza mediante la obtención de las diferentes distancias entre caras, vértices y aristas. Entre estos métodos destaca el propuesto por Gilbert [38]. Este método fue ampliado para ser aplicado a cualquier sólido convexo en [39].

Con objeto de obtener rutinas de detección de colisiones rápidas ha sido frecuente el modelado de sólidos con primitivas que guardan cierta simetría, en especial esferas [86]. Tornero *et al.* [117],[118] proponen la utilización del método de Gilbert [38] usando el concepto de politopos esféricos.

Otra constante en este tipo de algoritmos es la utilización de modelos jerárquicos que caracterizan los sólidos a distintos niveles de detalle, de forma que los niveles más bajos consiguen detecciones rápidas con modelo poco aproximados, mientras que los niveles más precisos sólo se utilizan en algunas situaciones concretas [59],[10]. Estas técnicas jerárquicas han sido utilizadas en esta tesis para obtener el diagrama de coordinación, utilizándose tres niveles de descripción: el primero, donde los elementos o conjuntos de elementos de cada robot se aproximan mediante esferas; el siguiente mediante politopos esféricos simples (biesferas) y el último, el modelo más exacto, para el que se ha utilizado CATIA, un paquete informático de CAD [54].

En el primero de los niveles cada objeto que compone el manipulador es modelado mediante dos esferas concéntricas, cuyo centro se corresponde con el centro geométrico del sólido [10]. La primera de ellas engloba completamente al objeto, teniendo como radio la distancia máxima desde el centro hasta cualquier punto del objeto. La segunda esfera está totalmente incluida dentro del objeto, es decir su radio se corresponde con la mínima distancia del centro a un punto del objeto. Dados dos objetos modelados de esta forma, el estudio de colisiones da lugar a los siguientes casos:

- Si las dos esferas exteriores no colisionan, se puede asegurar que dichos sólidos no colisionan (Figura 3.2-a).

- Si las dos esferas interiores colisionan, se afirma que dichos objetos colisionan (Figura 3.2-b).

- En cualquier otra circunstancia, no existe certeza sobre las colisiones de los objetos por lo que habrá que recurrir a modelos más aproximados (Figura 3.2-c).

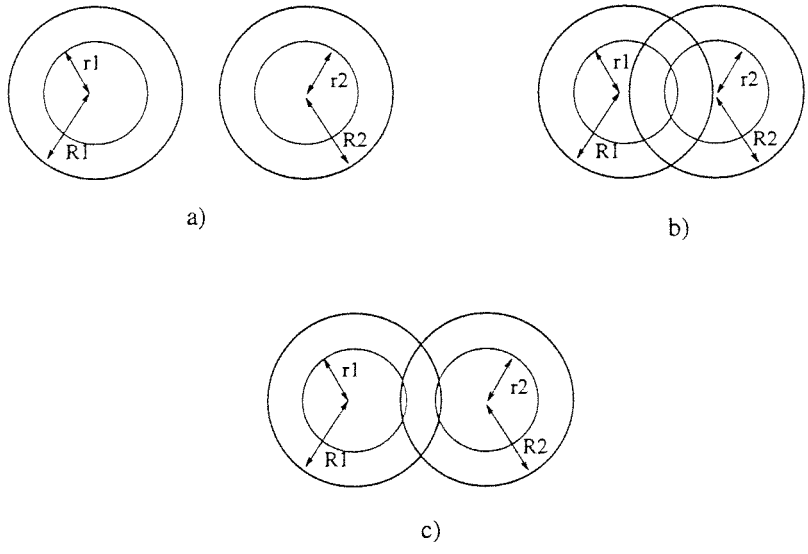

Figura 3.2 Detección de colisiones en el nivel de representación mediante esferas. a) Sin colisión. b) Colisión. c) Estado indeterminado.

La principal ventaja de modelar con esferas se encuentra en su simplicidad, ya que la detección de colisiones entre dos sólidos se reduce al cálculo de la distancia entre dos puntos, los centros de las esferas.

El segundo nivel se basa en la utilización de politopos esféricos [118] en su modo más simple. Una esfera se define como un vector de cuatro dimensiones, de forma que las tres primeras coordenadas se corresponden con el centro y la cuarta con el radio de la esfera, siendo *SS* el espacio formado por todas las esferas. Un politopo esférico es el casco convexo de un conjunto finito de esferas primitivas, es decir:

$$S_{0\text{-}n}=\left\{ s\in SS \;/\; s=\sum_{i=0}^{n}\lambda_i s_i,\; s_i\in\{s_0,s_1,\ldots,s_n\},\; \lambda_i\geq 0\;\forall i,\; \sum_{i=0}^{n}\lambda_i=1 \right\} \tag{3.9}$$

Un politopo formado a partir de dos esferas primitivas se denomina biesfera. Particularizando (3.9) se obtiene la expresión para una biesfera:

$$S_{0\text{-}1}=\{s\in SS \;/\; s=s_0+\lambda_1(s_1-\lambda_0),\; 0\leq\lambda_1\leq 1\} \tag{3.10}$$

Físicamente una biesfera se correspondería con cilindros terminados en semiesferas (cilindros esféricos) y con conos o troncos de cono igualmente terminados en esferas (conos o troncos de cono esféricos). Nótese que este tipo de sólidos se adapta de una forma relativamente precisa al tipo de sólido que generalmente forman los enlaces de los manipuladores. En [118] se explica de forma detallada los pasos a seguir para obtener la distancia entre dos biesferas.

Al igual que en el primer nivel de detalle, un sólido puede modelarse mediante dos biesferas, la primera que envolverá totalmente al sólido y la segunda incluida en éste. La detección de colisiones se realiza igual que en el caso anterior.

Para modelar niveles más precisos se podría haber optado por utilizar las técnicas propuestas en [118] mediante la utilización de politopos esféricos más complejos y splines esféricos. Sin embargo, por simplicidad se ha optado por utilizar un modelo tan preciso como se desee mediante la utilización del paquete informático de CAD denominado CATIA. En este caso, los modelos se han definido mediante primitivas simples: esferas, cilindros, prismas, etc., mientras que para la detección de colisiones se han utilizado las rutinas definidas en el propio CATIA. La poca eficiencia de estas rutinas ha hecho necesario la utilización de la estructura jerárquica definida anteriormente.

3.3.1.2 Detección de colisiones para celdas

El apartado anterior describía el cálculo de colisiones cuando el sistema de robots se encontraba en una configuración determinada. Sin embargo el cálculo del estado de una celda del diagrama de coordinación es algo más complicado debido a que no se trabaja con configuraciones concretas sino con los rangos de configuraciones, a los que dan lugar los

intervalos en los que se han dividido los caminos de los robots. La complejidad de este problema hace que se opte por soluciones aproximadas, encontrándose en la bibliografía básicamente dos tipos de soluciones. La primera consiste en modelar el robot como el volumen barrido por las distintas configuraciones que componen el intervalo, mientras que la segunda calcula el desplazamiento máximo que se puede producir en algún punto de cada elemento del manipulador en dicho intervalo, y aumenta el tamaño del sólido que forma el elemento de acuerdo a dicha cantidad. Estos dos tipos de soluciones se describen en los siguientes apartados.

Cálculo del volumen barrido

Este procedimiento para detectar colisiones entre segmentos de caminos ha sido extensamente utilizado en la literatura [71],[72],[36],[85],[123]. El *volumen barrido* de un objeto *A* que se mueve siguiendo un cierto camino es el resultado de obtener el conjunto de todos los puntos que pertenecen al objeto en algún momento de su movimiento. Este método propone utilizar como sólido para la detección de colisiones el volumen barrido por el robot en cada uno de los intervalos en que se ha descompuesto el camino (Figura 3.3).

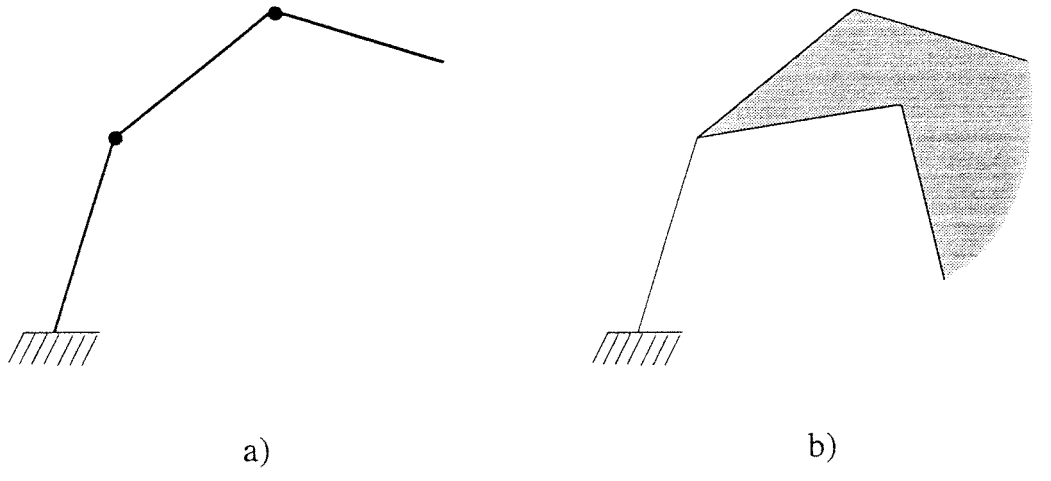

Figura 3.3 a) Robot plano con tres articulaciones. b) Volumen barrido en el robot anterior en el movimiento de las dos últimas articulaciones.

El principal inconveniente de este método radica en la dificultad de obtener de forma explícita dichos volúmenes para cualquier objeto tridimensional, realizándose normalmente aproximaciones mediante poliedros convexos [71],[36],[123]. El problema es más complicado para manipuladores en intervalos que supongan el movimiento de varias

articulaciones simultáneamente, ya que el movimiento de una de ellas arrastra a todas las posteriores (Ver Figura 3.3). En [85] O'Donnell *et al.* describen una aproximación a este problema. Estas dificultades hacen que este método se aplique cuando los intervalos sean grandes, o cuando se requiera una mayor precisión.

Cálculo del desplazamiento máximo

Esta técnica fue propuesta por Lozano-Pérez [73] y aplicada posteriormente por Chang *et al.* [15] y Conte *et al.* [19]. Consiste en obtener el desplazamiento máximo que se puede producir en cada elemento del manipulador y aumentar el tamaño del sólido que representa el elemento un una cantidad igual a dicho desplazamiento. Este método es mucho más conservador que el anterior, pero tiene la ventaja de ser también más simple.

Supóngase que un manipulador está posicionado en el centro del intervalo de trabajo. Sea Δ_k el máximo desplazamiento cartesiano que sufre cualquier punto del elemento k del robot en el rango de valores articulares del intervalo. Si cada uno de estos enlaces es agrandado con un radio igual a Δ_k el sólido resultante estará incluido dentro del volumen barrido.

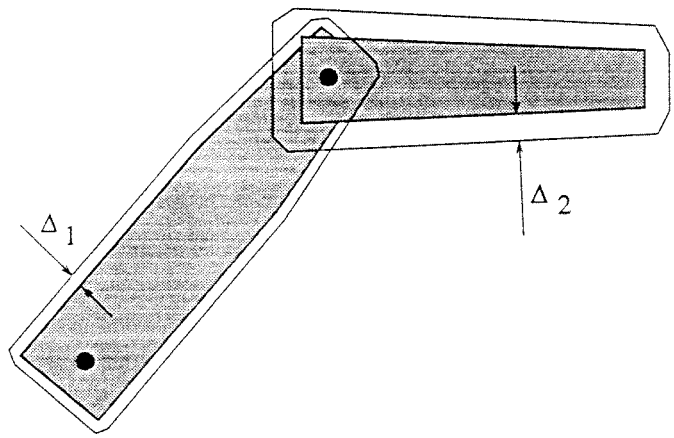

Figura 3.4 Robot plano con dos articulaciones, con sus enlaces aumentados una cantidad Δ_k.

Sea un sólido que gira alrededor de un eje. Si d es la distancia máxima del eje a un punto del sólido y θ es el ángulo de giro, el desplazamiento máximo vendrá dado por la expresión de la cuerda de una circunferencia, es decir, $d\sqrt{2(1-\cos\theta)}$. En un robot manipulador, esta expresión se complica debido a que el desplazamiento máximo depende tanto del giro de la articulación correspondiente como de la rotación de todos los elementos anteriores.

A título ilustrativo se muestra una expresión para los desplazamientos máximos de un robot plano con n articulaciones obtenidos a partir de los resultados de Lozano-Pérez [72]. Sea ε_i el máximo desplazamiento angular del enlace i en el intervalo, r_i la máxima distancia desde la articulación i hasta cualquier punto del enlace i y l_i la distancia desde la articulación i a la articulación i+1. La expresión indicada es:

$$\Delta_k = \sum_{i=1}^{k} \left[\left(\sum_{j=i}^{k-1} l_j + r_k \right) \cdot \sqrt{2(1-\cos\varepsilon_i)} \right] \tag{3.11}$$

que se corresponde con el desplazamiento mayor que se puede dar, cuando todas las articulaciones se encuentran totalmente extendidas. En robots tridimensionales, estos cálculos son diferentes debido a que los ejes de rotación no son necesariamente paralelos. En [15] se realizan los cálculos de los desplazamientos de las cinco primeras articulaciones para un robot tipo PUMA, a partir de los cuales se obtienen los resultados que se presentan a continuación. Sea d_{ik} la distancia máxima entre el eje de giro del elemento i y un punto del elemento k cuando el robot se encuentra totalmente estirado (con los elementos del 2 al 5 paralelos al suelo). El desplazamiento para el primer elemento es:

$$\Delta_1 = d_{11}\sqrt{2(1-\cos\varepsilon_1)} \tag{3.12}$$

El segundo eje de rotación es perpendicular al primero, que dará lugar a un desplazamiento perpendicular respecto al primero, por que la expresión del desplazamiento es:

$$\Delta_2 = \sqrt{2\left(d_{12}^2(1-\cos\varepsilon_1)+d_{22}^2(1-\cos\varepsilon_2)\right)} \tag{3.13}$$

Igualmente, teniendo en cuenta que los ejes de la tercera y de la quinta articulación son paralelos a la segunda, y que el giro de la cuarta articulación por coincidir su eje con el eje de simetría del cuarto elemento no afecta al desplazamiento, quedan las siguientes expresiones para los demás desplazamientos:

$$\Delta_3 = \sqrt{2\left(d_{13}^2(1-\cos\varepsilon_1)+\left(\sum_{i=2}^{3} d_{i3}\sqrt{2(1-\cos\varepsilon_i)}\right)^2\right.}$$

$$\Delta_4 = \sqrt{2\left(d_{14}^2(1-\cos\varepsilon_1)+\left(\sum_{i=2}^{3} d_{i4}\sqrt{2(1-\cos\varepsilon_i)}\right)^2\right.} \qquad (3.14)$$

$$\Delta_5 = \sqrt{2\left(d_{15}^2(1-\cos\varepsilon_1)+\left(\sum_{i=2,3,5} d_{i5}\sqrt{2(1-\cos\varepsilon_i)}\right)^2\right.}$$

Estas expresiones tienen la ventaja de la simplicidad, ya que al ser independientes de la posición en que se encuentre el robot, sólo es necesario obtener las distancias d_{ij} una vez. Al mismo tiempo, tienen el inconveniente de resultar bastante conservadores, ya que para configuraciones distintas a la considerada (la más desfavorable), los desplazamientos obtenidos serán en general menores. Una alternativa sería utilizar las expresiones anteriores, pero calculando para cada intervalo las distancias d_{ij}, de acuerdo con la configuración del robot en dicho intervalo.

El método del desplazamiento máximo irá proporcionando valores mayores para el desplazamiento a medida que el elemento considerado se aleja de la base. Evidentemente, el método puede ser aplicado de forma práctica sólo si los intervalos son suficientemente pequeños.

A título de ejemplo, indicar que para un intervalo con un giro máximo por articulación de 2 grados (los considerados en la mayor parte de los ejemplos de esta tesis), se obtiene para un robot SCARA SONY SRX-4CH un desplazamiento en el caso más desfavorable para el elemento terminal del orden de 3,4 cm., para un robot PUMA el desplazamiento ha sido de 5,7 cm., mientras que para un robot SCORBOT V se ha obtenido un desplazamiento máximo de 3,7 cm. Estos desplazamientos se han considerado aceptables, por lo que ha sido método usado para el cálculo de colisiones en celdas.

3.3.2 Clausura SW

En un diagrama de coordinación, existen algunas celdas que aún siendo libres, no pueden pertenecer a ningún camino coordinado libre de colisiones. En efecto, como se puede observar en la Figura 3.5-a, debido a que los caminos de coordinación deben ser monótonos crecientes, se puede llegar a situaciones en que se alcanza un *punto muerto* (*deadlock*). En la Figura 3.5-b aparecen marcados todas aquellas celdas que presentan esta problemática, y que se denominarán celdas de pseudo-colisión. Debido a esto, y en relación a la obtención de caminos de coordinación libres de colisión, todas estas celdas pueden ser consideradas de tipo colisión. Preparata *et al.* [95] presentan un método para obtener estas celdas de pseudo-colisión, siendo necesario para su desarrollo introducir previamente el concepto de *clausura-SW* (*South-West Clausure*), definido inicialmente por Yannakakis *et al.* [128] para modelar el acceso a bases de datos concurrentes.

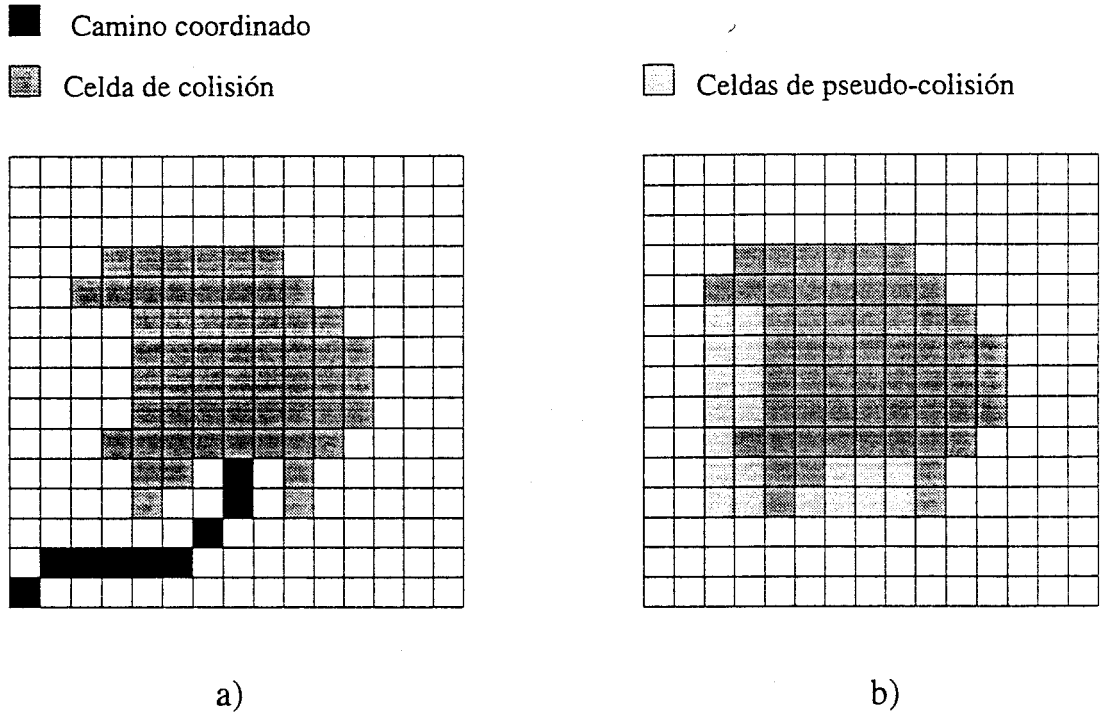

Figura 3.5 a) Camino que acaba en punto muerto. b) Celdas de tipo pseudo-colisión.

Sea un diagrama de coordinación de dos dimensiones y $s=(s_1,s_2)$ y $t=(t_1,t_2)$ dos celdas pertenecientes a dicho diagrama. Se dicen que s y t son dos celdas incomparables [62] si y sólo si $(s_1-t_1)(s_2-t_2)<0$. Supóngase que s y t son incomparables y sin pérdida de generalidad que $s_1<t_1$. Se denomina *conjugado SW* de s y t a la celda (s_1,t_2) (Ver Figura 3.6-a). Una región conexa R en el espacio de coordinación es *cerrada SW* si y sólo si:

Para todo par de celdas incomparables s y t de R (se supone $s_1<t_1$), de forma que exista un camino C incluido en R que los una, con la condición que para toda celda $x \in C$ cumpla que $x_1 \geq s_1$ y $x_2 \geq t_2$, entonces su conjugado-SW también pertenece a R. (Figura 3.6-b)

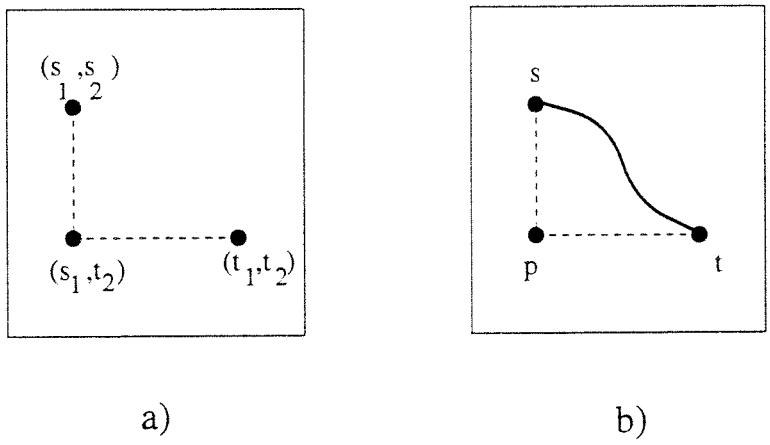

Figura 3.6 a) Dos celdas incomparables y su conjugado SW. b) Condición de una región para ser cerrada SW.

La *clausura SW* de una región conexa S es la más pequeña de las regiones cerradas SW conteniendo S. Del mismo modo, se puede demostrar que la clausura SW de regiones no conectadas es la clausura SW de la unión de las clausuras SW de todas sus componentes conexas.

Teniendo en cuenta estas definiciones, es fácil comprobar que la unión de las celdas de colisión y de pseudo-colisión se corresponde con la clausura-SW de la región de colisión. En efecto, si una celda del diagrama de coordinación $p=(p_1,p_2)$ pertenece a la clausura-SW de la región de colisión implica alguno de los dos siguientes casos:

a) Existen dos celdas incomparables s y t pertenecientes a la misma zona conexa de la región de colisión, de forma que p es el conjugado SW de dichos puntos. En este caso la existencia del camino C definido anteriormente entre s y t forma una

"barrera" que impide que cualquier camino coordinado a partir de p alcance el punto final del espacio de coordinación.

b) Si por el contrario s y t pertenecen a dos zonas conexas distintas de la región de colisión (o a la clausura SW de alguna de estas dos regiones), deberá existir un camino entre estos puntos con las condiciones anteriormente indicadas formado por celdas de colisión o de pseudo-colisión. De nuevo cualquier camino de coordinación a partir de p deberá incluir alguna de las celdas de este camino entre s y t. Pero esta celda no puede ser ni de colisión, ni de pseudo-colisión, ya que por definición, éstas no pueden pertenecer a ningún camino coordinado.

Por tanto para la obtención de caminos de coordinación, se procederá a añadir a la zona de colisión, todas las celdas de pseudo-colisión, es decir, se sustituirá la región de colisión por su clausura-SW.

3.3.3 Obtención de diagramas de coordinación bidimensionales

Basado en el algoritmo propuesto en [95], en este apartado se presenta un método eficiente para la obtención del diagrama de coordinación para dos robots, en que la región de colisión se sustituye por su clausura SW. La principal ventaja del método propuesto es que no es necesario calcular explícitamente el estado de colisión de todas y cada una de las celdas. Como se verá posteriormente, sólo se evalúan las celdas libres y las del perímetro de las zonas de la clausura SW de región de colisión.

Sea S la región de colisión de un diagrama de coordinación. Se añadirán inicialmente a S una serie de celdas mediante las dos siguientes reglas [62] (Figura 3.7):

- Si existe $j \in [1, max_2]$ tal que la celda (max_1, j) es de tipo colisión, entonces todas las celdas (max_1, k) con $1 \leq k \leq j$ son incluidas en S.

- Si existe $i \in [1, max_1]$ tal que la celda (i, max_2) es de tipo colisión, entonces todas las celdas (k, max_2) con $1 \leq k \leq i$ son incluidas en S.

Nótese que las celdas añadidas son de pseudo-colisión. Con esta nueva configuración de la región de colisión se puede aplicar el algoritmo propuesto por Preparata *et al.* [95]. El algoritmo se basa en la realización de un recorrido completo por el diagrama de coordinación mediante una línea de barrido (*sweep-line*), que se corresponde con el camino completo de uno de los robots, desde la posición final del otro robot hasta la inicial. A partir de ahora, sin pérdida de generalidad, se supondrá que la línea de barrido se corresponde en el diagrama de coordinación con el camino del robot 2, es decir con

líneas verticales desde la parte superior a la inferior del diagrama de coordinación, realizándose el barrido de derecha a izquierda (Ver Figura 3.8).

Se denomina *sección activa* en la línea de barrido a cada uno de los intervalos que intersecan con la región de colisión. La Figura 3.8 muestra las dos zonas activas cuando la línea de barrido está en la posición *j*. En el estado inicial del algoritmo, es decir con la línea de barrido situada en la posición final del robot 1, hay dos posibilidades: o bien puede no existir ninguna sección activa, o bien existe una sola sección activa, que abarca desde la celda correspondiente a la posición inicial del robot 2 hasta la última celda en dicha línea que produzca colisión (Figura 3.7). A partir de las secciones activas de la posición inicial, se va realizando un barrido del diagrama hacia la izquierda, de celda en celda, de forma que en cada posición de la línea se van actualizando las secciones activas como se indica en los siguientes párrafos.

Sea $SA_j=\{I_j^i=(a_j^i,b_j^i)\ \ i=1,..,NINT_j\}$, el conjunto de secciones activas cuando la línea de barrido se encuentra en el intervalo *j* del camino del robot 1 (etapa *j* del barrido), I_j^i será cada una de las secciones activas para dicha línea de barrido, *a* y *b* son respectivamente los límites inferior y superior de cada sección y $NINT_j$ el número de secciones activas.

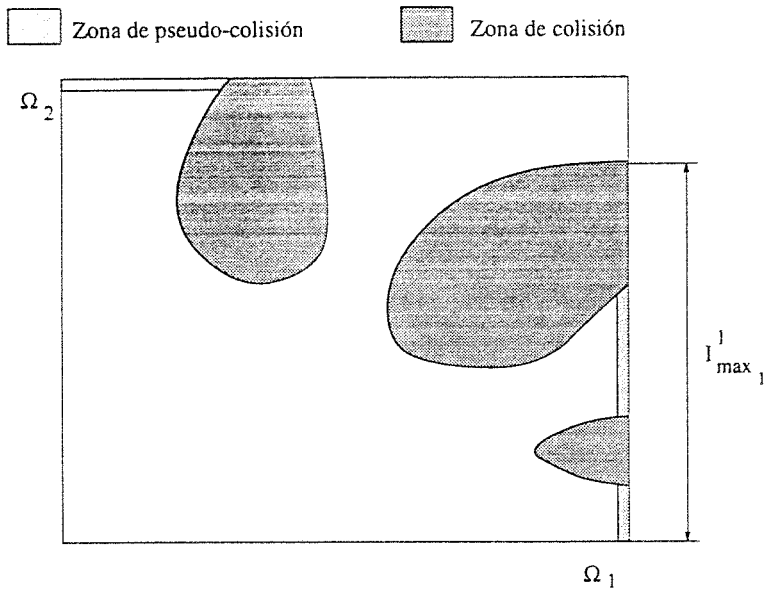

Figura 3.7 Zonas de pseudo-colisión añadidas en la etapa inicial del algoritmo.

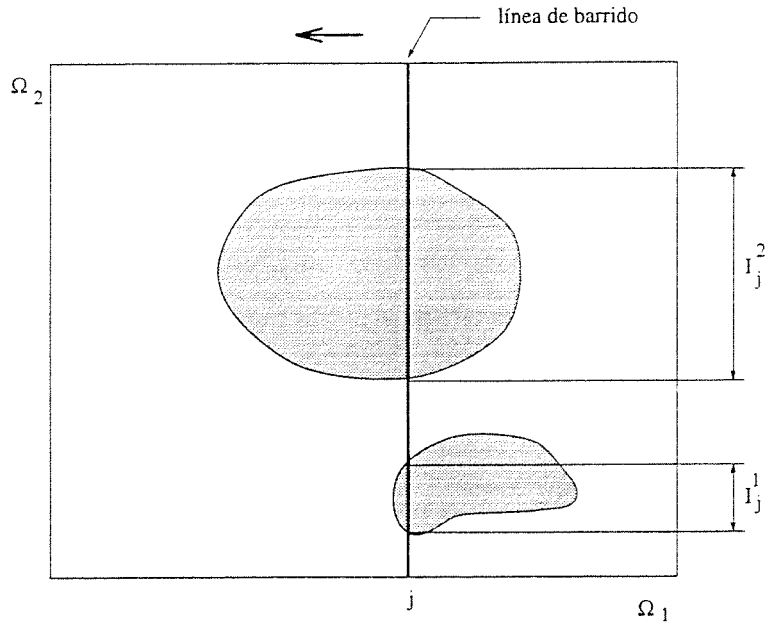

Figura 3.8 Línea de barrido e intervalos activos.

Para calcular las secciones activas en la etapa j-1, es decir SA_{j-1}, a partir de las secciones activas en j, se van obteniendo los estados de colisión de las celdas que forman parte de la línea desde la celda superior (posición final del robot 2) hasta la primera celda de tipo colisión que se encuentre o bien hasta el final de la línea. Se pueden presentar distintos casos:

Si todas la celdas de la línea son libres, significa que SA_{j-1} es un conjunto vacío, es decir, no existen secciones activas. Si por el contrario se ha llegado a una celda de tipo colisión, siendo c el intervalo del camino del robot 2 que se corresponde con dicha celda, puede ocurrir:

- Si $c \notin I_j^i$ para ningún valor de i, se continua obteniendo los estados de las celdas siguientes hasta que o bien se llegue a una celda libre de colisión (c'), o bien hasta que se alcance una celda que se corresponda con el límite superior de alguna sección I_j^k. En el primer caso (Figura 3.9-a) se creará una nueva sección activa en SA_{j-1}, con límite superior c y límite inferior c'+1. En el segundo caso (Figura 3.9-b), la zona de colisión que daba lugar a I_j^k se mantiene dando lugar a la sección I_{j-1}^k, de forma que en general será necesario modificar los límites. En este caso el nuevo límite superior de la sección pasará a ser c. Posteriormente se explica la forma de obtener el límite inferior.

- Si $c \in I_j^k$ es similar a la segunda circunstancia del apartado anterior (Figura 3.9-c). De nuevo será necesario modificar los límites de la sección activa, de forma que en I_{j-1}^k el nuevo límite superior será c.

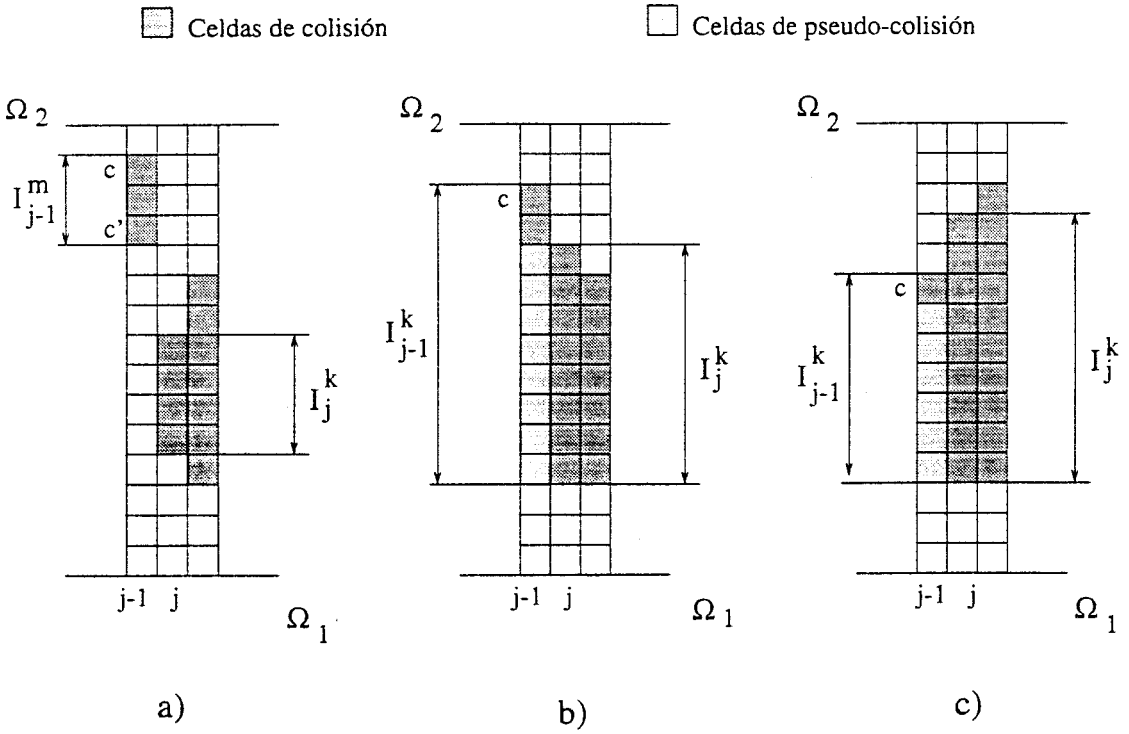

Figura 3.9 Obtención de las secciones activas en la etapa *j*-1 a partir de la etapa *j*. a) Creación de una nueva sección activa. b) y c) Modificación de los límites de secciones activas existentes en la etapa *j*. Las celdas de pseudo-colisión no es necesario evaluarlas.

Con respecto a la obtención de los límites inferiores de los dos casos estudiados anteriormente, se parte de la situación reflejada en la Figura 3.10-a. En la etapa *j* existe una zona de colisión que da lugar a la sección activa I_j^k, y dicha zona se mantiene en la etapa *j*-1, habiendo sido calculado el nuevo límite superior. Nótese que el nuevo límite inferior será como máximo igual al correspondiente a la sección activa I_j^k, es decir a_j^k, independientemente que las celdas en la etapa *j-1* situadas entre c y a_j^k sean de tipo colisión o no (estas celdas forman parte de la clausura SW), por lo que no es necesario comprobar su estado.

Por tanto, para obtener el límite inferior de la nueva sección activa, se hace éste igual al límite inferior de la sección de la etapa anterior de la cual proviene. A partir de este punto se continua evaluando el estado de las celdas hacia abajo, hasta que se encuentre

una celda libre c', el extremo inferior del diagrama o una celda c' que pertenezca a otra sección I_j^m perteneciente a SA_j.

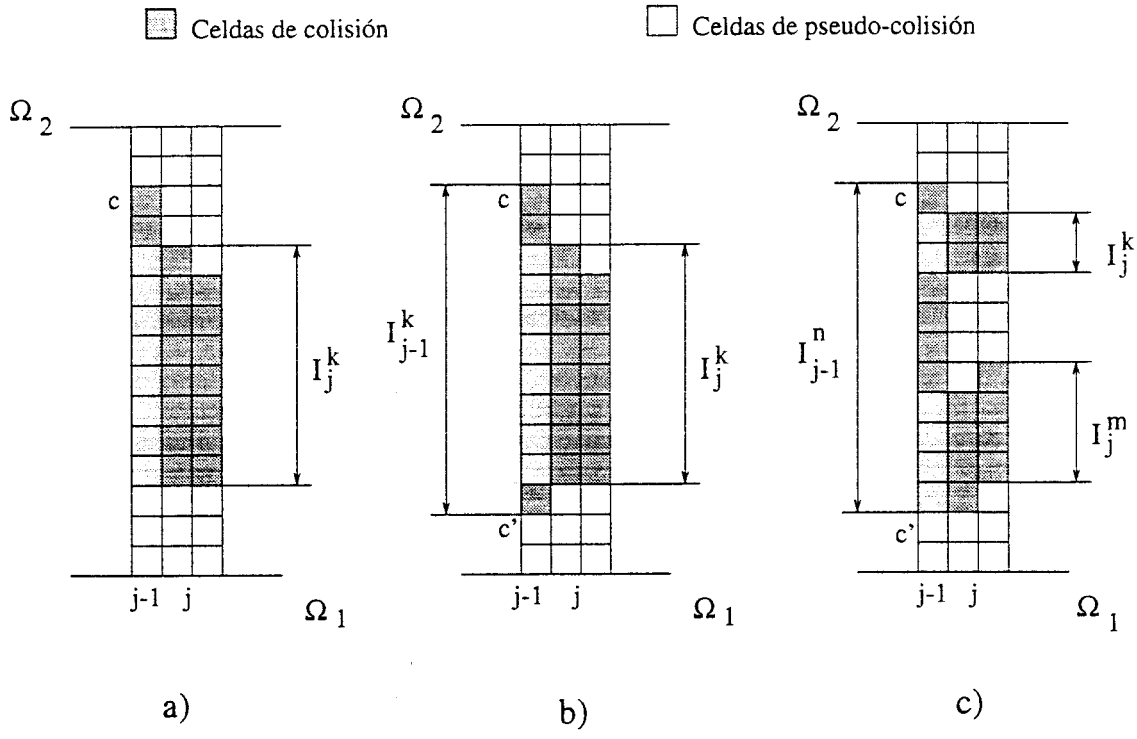

Figura 3.10 Obtención de los límites inferiores de las secciones activas. a) Estado de partida. b) Modificación de los límites inferiores respecto a los límites en la etapa anterior. c) Unión de dos secciones activas de la etapa anterior.

En el primer caso (Figura 3.10-b), el extremo inferior de la sección será $c'+1$. En el segundo caso el inicio de la sección se corresponde con el extremo del diagrama de coordinación. El tercer caso es el más complicado (Figura 3.10-c), pues supone que dos zonas de colisión distintas en la etapa j se unen en una sola en la etapa j-1. Por tanto, se hace el límite inferior igual al correspondiente al de la sección activa I_j^m y se repite el mismo proceso de búsqueda del límite inferior a partir de éste.

La clausura SW estará formada por la superficie cubierta por todas las secciones activas al desplazar la línea de barrido desde el lateral derecho del diagrama de coordinación hasta el lateral izquierdo. La utilización de este algoritmo tiene dos grandes ventajas: en primer lugar la obtención de la clausura SW reduce el espacio de búsqueda de soluciones coordinadas. Por otro lado, la forma en que ha sido diseñado el algoritmo hace que no sea necesario evaluar el estado de todas las celdas del diagrama, ya que de hecho

no se evalúa ninguna celda del interior de la clausura SW, sólo las celdas libres y las del contorno de dicha región. Esto supone una reducción considerable en tiempo de cálculo para diagramas de coordinación con grandes zonas de colisión. En Figura 3.11 se muestra el resultado final de la aplicación del algoritmo.

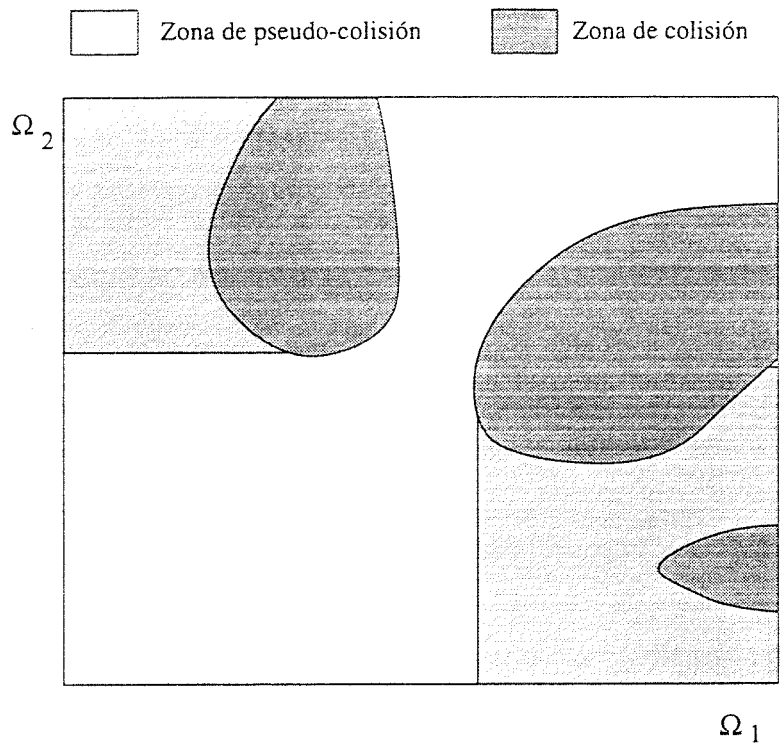

Figura 3.11 Resultado final de la aplicación del algoritmo a un diagrama de coordinación.

Evidentemente, del mismo modo que se ha utilizado el camino del robot 2 como línea de barrido, se podría haber utilizado el robot 1, dando lugar a un barrido de arriba a abajo, obteniéndose los mismos resultados.

3.3.4 Diagramas de coordinación de más de dos dimensiones

Hasta el momento todos los conceptos desarrollados están aplicados a la obtención de diagramas de coordinación para dos robots. En este apartado se analizan los diagramas de coordinación en el caso en que el número de robots es mayor.

Un diagrama de coordinación para R robots es un espacio formado por celdas de dimensión R. Teniendo en cuenta que el camino de cada robot j está dividido en max_j intervalos, el número de celdas en un diagrama de coordinación R-dimensional es $max_1 \times max_2 \times \ldots \times max_R$. Sin embargo se comprobará que el número de celdas a analizar es mucho menor.

Supóngase que se está estudiando la coordinación de tres robots, que dará lugar a un diagrama de coordinación de tres dimensiones. Lógicamente si entre los robots 1 y 2 existe colisión en un punto determinado de sus respectivos caminos, esta colisión seguirá existiendo independientemente de la posición en que se encuentre el robot 3. Esto hace que el diagrama de coordinación tenga regiones de colisión en forma de "columnas", como se aprecia en Figura 3.12. Por tanto, la región de colisión estará formada por un conjunto de volúmenes cilíndricos en las direcciones de los tres ejes.

Esta circunstancia hace que la obtención del diagrama de coordinación para R grados de libertad se reduzca a la obtención de los diagramas de coordinación bidimensionales de cada robot con todos los demás. Sea DC_{ij} el diagrama de coordinación bidimensional entre los robots i y j, y sea RC_{ij} su correspondiente región de colisión. Una celda del diagrama de coordinación R-dimensional $(x_1,\ldots,x_R)$ estará libre de colisión si cumple la siguiente condición:

$$\forall i,j \text{ con } i \neq j \Rightarrow (x_i,x_j) \notin RC_{ij} \tag{3.15}$$

Por tanto, el número de diagramas de coordinación bidimensionales que será necesario obtener vendrá dado por la conocida expresión que proporciona las combinaciones de R elementos tomadas de dos en dos:

$$\text{número DC} = \frac{R!}{2(R-2)!} \tag{3.16}$$

De esta forma disminuye de forma considerable el número de celdas a calcular, además de la ventaja que supone reducir un problema R-dimensional a varios problemas bidimensionales, con lo que las técnicas vistas anteriormente para la obtención de los diagramas continúan siendo válidas.

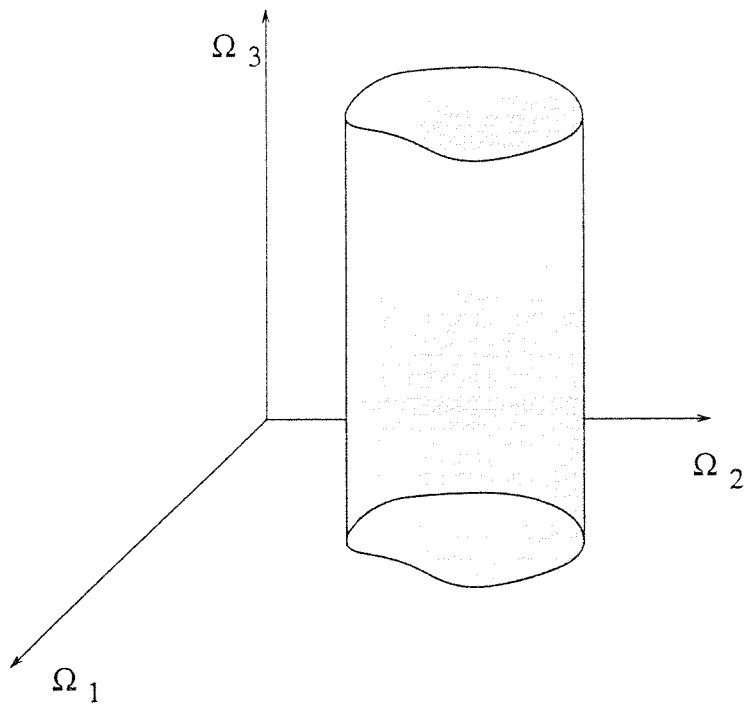

Figura 3.12 Diagrama de coordinación de tres robots con una región de colisión producida por los robots 1 y 2.

3.4 Diagramas de coordinación bidimensionales sin solución. Modificación de caminos

Dado un camino para cada uno de los robots, no siempre es posible encontrar una forma coordinada de ejecutar los movimientos sin que exista colisión entre ellos. Este hecho se refleja en diagramas de coordinación para los que no existe ningún camino de coordinación libre de obstáculos, diciéndose en tal caso se dice que el diagrama de coordinación no tiene solución. El diagrama de la Figura 3.11 tiene solución, mientras que el de la Figura 3.16 no la tiene.

Los algoritmos propuestos para la obtención de los caminos de coordinación, como se verá posteriormente, son métodos de búsqueda computacionalmente costosos. Debido a esto, es interesante antes de realizar esta búsqueda, conocer si el diagrama de

coordinación considerado tiene o no solución, con objeto de no realizar una búsqueda infructuosa. De nuevo la obtención de la clausura SW proporciona esta información. En efecto, es fácil comprobar que para que un diagrama de coordinación tenga solución es condición necesaria y suficiente que la celda inicial $\mathbb{C}_0$ no sea de tipo pseudo-colisión (ni lógicamente de tipo colisión).

Sin embargo, la no existencia de solución en un diagrama de coordinación no implica que no sea posible encontrar una solución coordinada al problema global, esto es, considerando como dato de partida las configuraciones iniciales y finales de los robots. Es posible que considerando otros caminos distintos para los robots, se encuentre una solución.

3.4.1 Modificación de los caminos

Para resolver problemas en los que las técnicas de planificación desacoplada no encuentran solución, se suele recurrir a métodos de planificación más generales, como la planificación con prioridad o la planificación centralizada, con el esfuerzo de cálculo que ello supone. Para problemas con un acoplamiento fuerte y entornos de trabajo con numerosos obstáculos, esto puede llegar a ser indispensable. Sin embargo, en las experiencias realizadas en esta tesis, se ha llegado a la conclusión de que en la mayoría de los casos no es necesario buscar caminos alternativos totalmente distintos a los iniciales, sino que con modificaciones más o menos pequeñas de éstos es posible encontrar una solución.

En esta tesis se proponen dos tipos de modificaciones a los caminos iniciales que se pueden aplicar a sistemas de dos robots. Ambas están basadas en la alteración del camino de un solo robot, teniendo en común la simplicidad, de forma que el esfuerzo de cálculo sea pequeño en relación al necesario para utilizar otra técnica de planificación. Además, el hecho de que la modificación consista en alterar parte del camino original hace que, en general, una parte importante del diagrama de coordinación original se conserve en el modificado.

Las técnicas de modificación de caminos propuestas en esta tesis no pretenden sustituir a otros métodos de planificación más generales, sino analizar si realizando una pequeña modificación del camino (realizada de forma heurística) es posible obtener un diagrama de coordinación con solución. Los resultados experimentales obtenidos hacen de estos procedimientos una técnica útil debido al gran número de problemas para el que encuentran una solución y el coste de cálculo requerido para obtenerla, aunque en ningún caso se pueda garantizar el encontrar una solución. En los siguiente apartados se estudian por separado cada una de las modificaciones propuestas.

3.4.1.1 Modificación tipo 1

La primera modificación propuesta consiste en realizar una modificación del camino de uno de los robots sustituyendo un tramo del camino por un nuevo tramo de forma que se evite la colisión con el otro robot, cuando éste se encuentre en la posición de su camino que se estima más favorable para este objetivo. La aplicación de esta modificación plantea varios aspectos fundamentales que es necesario resolver:

- Robot en cuyo camino se realizará la modificación (robot primario)
- Determinación del tramo del camino del robot primario que va a ser sustituido.
- Posición más favorable en el otro robot (robot secundario) para ser evitado por el primario.
- Finalmente será necesario obtener el nuevo tramo, de tal forma que se pueda garantizar que el diagrama de coordinación resultante va a tener solución.

Considérese el diagrama de coordinación sin solución de la Figura 3.13-a, donde se representa un camino coordinado que atraviesa la zona de colisión horizontalmente (la posición del robot 2 es fija). ω_1 y ω_2 son los intervalos del camino del robot 1 donde empieza y termina la colisión con el robot 2 cuando éste se encuentra en la posición fija $\hat{\rho}$. La modificación propuesta consiste en buscar un camino para el robot 1 desde $\hat{\omega}_1$ hasta $\hat{\omega}_2$ considerando como obstáculos los propios del entorno y el robot 2 en la posición fija $\hat{\rho}$. En la Figura 3.13-b se muestra gráficamente la modificación realizada. Se representa el espacio de las configuraciones del robot 1 (suponiendo que tiene dos grados de libertad θ_1 y θ_2) frente al camino parametrizado del robot 2. El diagrama de coordinación en este espacio es una superficie paralela a Ω_2 y la zona de colisión entre los dos robots forma parte de un volumen que representa las configuraciones del robot primario que dan lugar a colisión con el robot secundario cuando éste se encuentra en algún punto del camino Ω_2, tal como aparece en la figura. La búsqueda se realiza en el plano horizontal $\Omega_2=\rho$, intentando bordear este volumen que representan las colisiones. La línea discontinua representa el camino modificado que como se puede comprobar comparte tramos con el original.

En las condiciones anteriores, si existe un camino de coordinación libre de obstáculos desde $\mathbb{C}_0$ hasta la celda (ω_1,ρ) y del mismo modo otro desde (ω_2,ρ) hasta la celda final $\mathbb{C}_{max}$, la modificación anterior garantiza que se obtendrá un diagrama de coordinación con solución, ya que dicha modificación asegura que al menos las celdas comprendidas entre (ω_1,ρ) y (ω_2,ρ) serán libres en el nuevo diagrama. El nuevo diagrama de coordinación se diferenciará del original exclusivamente en la banda comprendida entre las dos líneas discontinuas que aparecen en la Figura 3.13-a. En la Figura 3.15-a se muestra el resultado de la modificación.

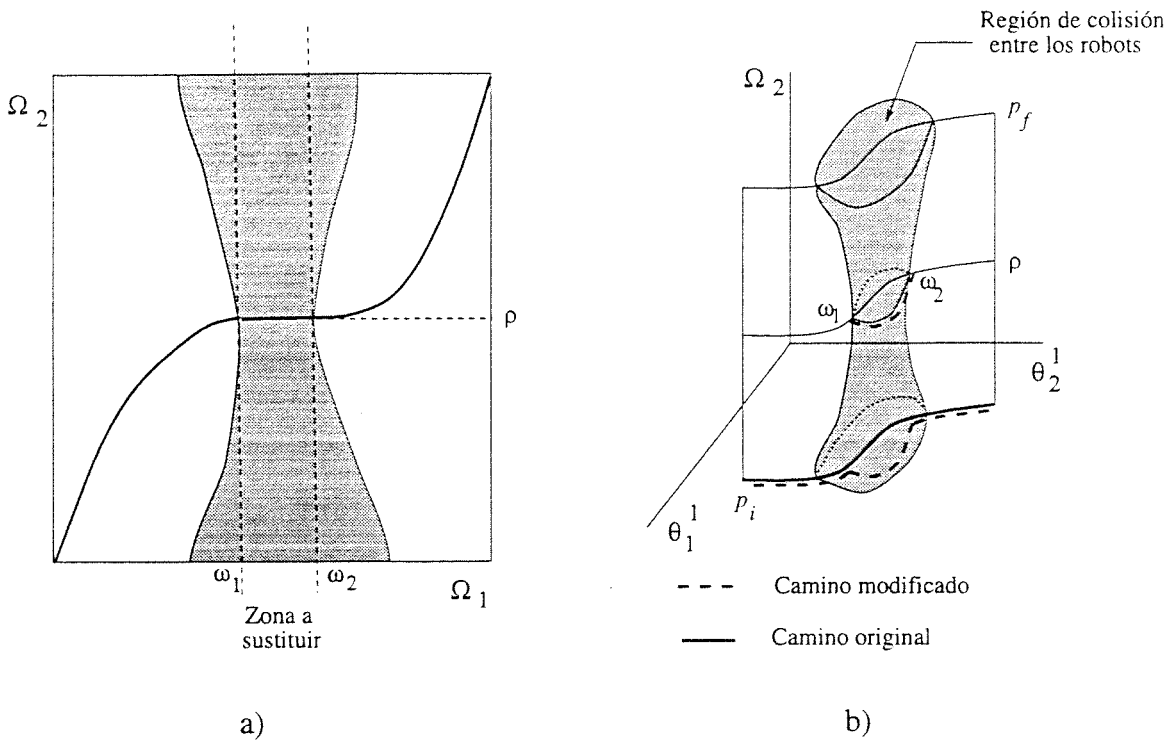

Figura 3.13 Modificación tipo 1. a) Diagrama de coordinación original. b) Representación del camino original y el modificado en el espacio formado por el espacio de configuración del robot modificado frente al camino parametrizado del otro robot.

3.4.1.2 Modificación tipo 2

La segunda modificación [99] consiste en apartar uno de los robots de su camino original en un punto determinado, dejar paso al otro robot y retomar de nuevo el camino original por el mismo punto. Esta modificación, que como se analizará posteriormente, es más complicada de obtener que la anterior, tiene su aplicación en aquellos casos en que debido al tipo de operación del robot sea imprescindible que recorra completamente su camino original. Al igual que en la modificación anterior, la utilización del método implica:

- Elección del robot primario.
- Elección del punto del camino del robot primario donde se realizará el abandono y el regreso al camino original.

- Búsqueda de un camino hasta un punto seguro, es decir una posición del robot primario que no colisione con el robot secundario en su movimiento.
- Garantizar que si se consigue realizar la modificación, el diagrama de coordinación resultante va a tener solución.

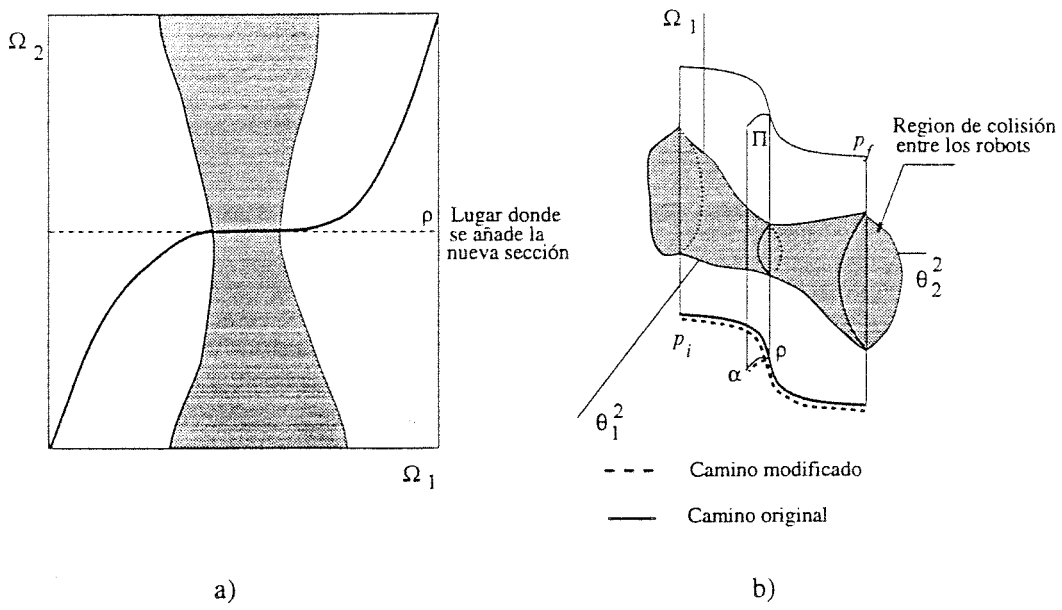

Figura 3.14 Modificación tipo 2. a) Diagrama de coordinación original. b) Representación de los caminos original y modificado en el espacio formado por el espacio de configuración del robot primario y el camino parametrizado del robot secundario.

Sea de nuevo el diagrama de coordinación sin solución de la Figura 3.14-a y el mismo camino de coordinación que en el caso anterior. En esta modificación, el robot primario será el robot 2. El objetivo es encontrar un camino desde el punto en el espacio de las configuraciones del robot primario $\hat{\rho}$ hasta un nuevo punto $\hat{\alpha}$ fuera del camino original que cumpla la condición (Ver Figura 3.14) de que al ser añadir al camino original los tramos $\hat{\rho}$-$\hat{\alpha}$ y $\hat{\alpha}$-$\hat{\rho}$ forme un diagrama de coordinación con solución. Se pretende por tanto añadir una nueva sección al diagrama original en la línea correspondiente a ρ que permita "romper" la zona de colisión. Nótese que al coincidir el camino de ida y el de vuelta, la nueva sección del diagrama de coordinación es simétrica respecto a la línea correspondiente a α. En la Figura 3.14-b se observa la nueva sección a añadir en el diagrama de coordinación, que se corresponde con la superficie Π y en la Figura 3.15-b el diagrama de coordinación resultante de aplicar esta modificación.

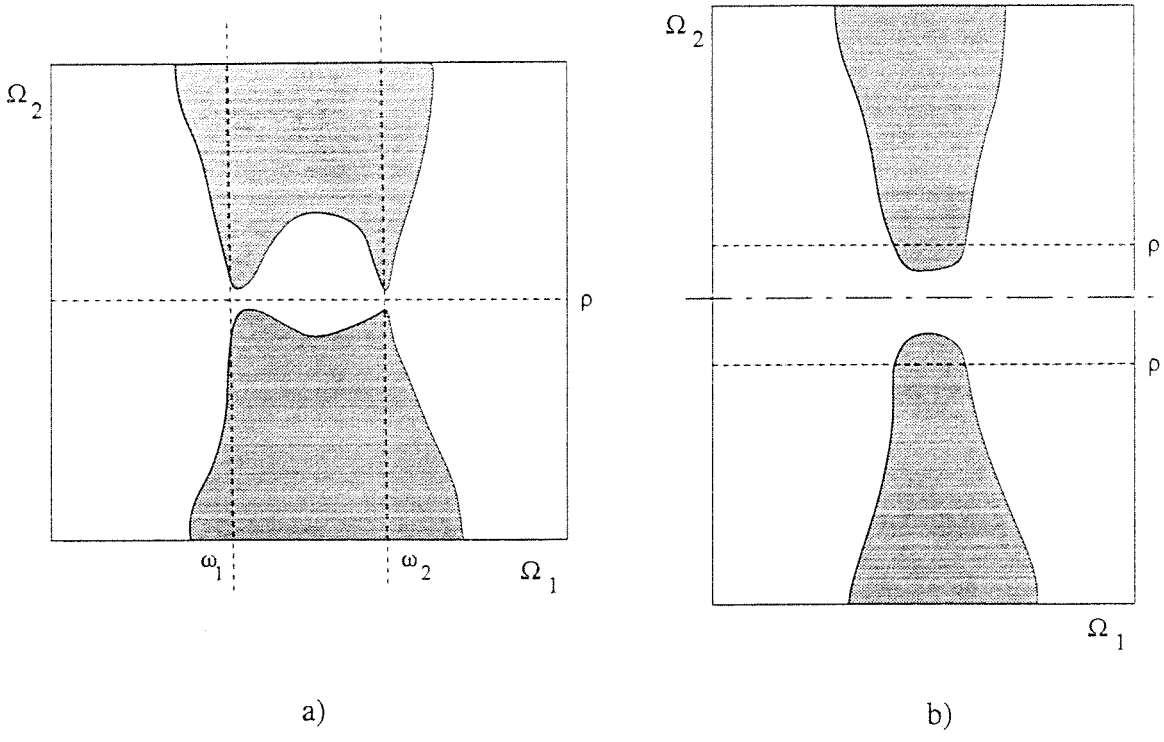

Figura 3.15 Diagramas de coordinación modificados. a) Modificación tipo 1. b) Modificación tipo 2.

3.4.1.3 Comparación con otras técnicas de planificación

La única técnica de planificación que puede asegurar el encontrar una solución si ésta existe es la *centralizada*. Sin embargo, para robots industriales, su utilización queda descartada por la gran dimensión del espacio de búsqueda. Una alternativa más viable puede ser la planificación con establecimiento de prioridad, que se puede enfocar como la búsqueda de un camino entre los puntos p_i y p_f de la Figura 3.14-b. Esta técnica, más global que las modificaciones propuestas en esta tesis, normalmente es capaz de resolver un mayor número de situaciones, pero a costa de un mayor esfuerzo de cálculo.

Sin embargo, es posible encontrar situaciones en las que un método de planificación con prioridad no puede encontrar solución, mientras que las modificaciones propuestas sí lo hacen. En efecto, sea una situación como la que se representa en la Figura 3.16-a donde dos robots planos con dos grados de libertad se mueven en un mismo entorno. En línea continua se indica el estado inicial de cada robot, y en discontinua el estado final. Este

problema no tiene solución con la planificación con prioridad ya que sería necesario la alteración de los caminos de los dos robots. Sin embargo, el carácter local de las modificaciones propuestas puede permitir resolver el problema.

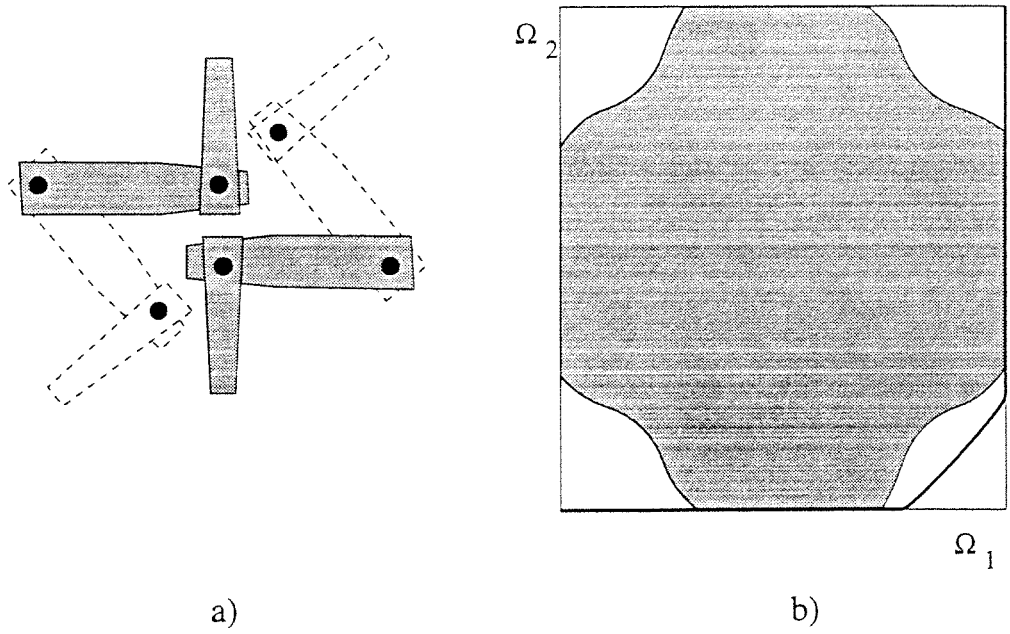

Figura 3.16 Problema que no puede resolverse mediante técnicas de planificación con prioridad. a) Representación de los robots. En línea discontinua el estado final. b) Diagrama de coordinación.

En la Figura 3.16-b se representa el diagrama de coordinación asociado al ejemplo anterior. Se puede comprobar que una única modificación no es capaz de dar lugar a un diagrama con solución. Sin embargo, el hecho de que las modificaciones planteadas sólo afecten a una parte del camino del robot primario hace que se puedan aplicar repetidamente sobre un mismo diagrama de coordinación. En efecto, un camino de coordinación como el dibujado en la figura necesitaría la aplicación de una modificación tipo 2 dos veces, cada una considerando como robot primario uno distinto. En el apartado de ejemplos de este mismo capítulo se muestra la resolución de un problema similar a éste.

3.4.2 Implementación de los métodos de modificación

La implementación de cada uno de los tipos de modificaciones propuestos se va a dividir en dos partes:

- **Selección de los parámetros previos a la modificación** : El objetivo es determinar el robot primario, y las posiciones claves de la modificación, esto es, posición del robot fijo y puntos de partida y vuelta al camino original en el robot primario si la modificación es de tipo 1. En la modificación tipo 2 se buscará el punto de separación del camino original en el robot primario, que coincidirá con el de vuelta. Esta selección se realizará de una forma prácticamente igual para los dos tipos de modificaciones.

- **Búsqueda del nuevo camino**: Será el encargado de encontrar el camino con el que se materializará la modificación. Este módulo es totalmente distinto para cada uno de los tipos de modificación.

3.4.2.1 Selección de los parámetros previos a la modificación

Como se deriva de los apartados anteriores, la determinación del robot primario y de las posiciones fundamentales de la modificación en un diagrama de coordinación sin solución pasan por la especificación de líneas de corte que atraviesen una región de colisión, conectando dos celdas libres. Por necesidad del método, estas líneas de corte estarán formadas por celdas consecutivas que formen o bien una línea vertical, o bien una línea horizontal. La dirección de la línea determinará el robot primario, mientras que la posición de la misma en el diagrama de coordinación fijará los puntos de comienzo y fin de la modificación, así como la posición fija del robot secundario si la modificación es de tipo 1 y el punto de separación del camino original si es de tipo 2. En algunos diagramas de coordinación es necesaria la especificación de varias líneas de corte, de forma que cada una de ellas dará lugar a una modificación distinta. Nótese que debido a la naturaleza del cambio que produce cada modificación tipo 1 sobre el diagrama de coordinación, éstas no interaccionan entre sí, de forma que el cambio que produce una sobre el diagrama, no afecta a la aplicabilidad de la siguiente (aunque como se verá posteriormente, puede hacerla innecesaria). Posteriormente se estudiará cómo en la modificación tipo 2 es necesario establecer unas condiciones para que no se vea afectada dicha aplicabilidad.

Por otro lado es importante hacer notar que la existencia de un camino entre la celda $\mathbb{C}_0$ y la primera celda de la línea de corte, y otro desde la última celda de la línea hasta $\mathbb{C}_{max}$, garantiza, en caso de que se encuentre una modificación, la obtención de un diagrama de coordinación con solución. Lo mismo ocurre para el caso de varias líneas de corte, si existe un camino que una la última celda de una línea de corte con la primera de

la siguiente (lo que obliga a establecer una ordenación entre las líneas de corte). En relación a la situación de las líneas de corte, también es importante que estén colocadas donde la modificación a realizar suponga un menor esfuerzo de cálculo. Esto es muy difícil de conocer a priori, por lo que se recurrirá a estimaciones heurísticas que evalúen la dificultad de realizar la modificación. Finalmente, en caso de existir varias posibilidades de modificación con la misma dificultad estimada, se deberá optar por aquella que se considere vaya a dar lugar a un camino coordinado en el que los robots recorran sus respectivos caminos con una mayor rapidez.

Por todo ello, el módulo de selección consistirá en la búsqueda de un camino coordinado formado por celdas libres o por celdas obstáculos, siempre que se atraviesen las distintas zonas de colisiones en línea recta. El método de búsqueda va a estar basado en un algoritmo del tipo A* (véase apéndice A). El objetivo es encontrar una secuencia de celdas consecutivas desde $\mathbb{C}_0$ hasta $\mathbb{C}_{max}$ que minimice una cierta función de evaluación asociada a los caminos. Como es bien conocido la función de evaluación se suele representar por:

$$f(C)=g(C)+h(C) \tag{3.17}$$

donde C es una celda del diagrama de coordinación, $g(C)$ es el coste acumulado desde la celda inicial a C, y $h(C)$ es una estimación del coste de C a la celda final. En este caso, la función de evaluación elegida se corresponde con el tiempo necesario para completar el movimiento coordinado de los dos robots. Para obtener este tiempo, se va a utilizar un modelo muy simple, ya que en esta etapa de la planificación se prefiere la rapidez en encontrar una solución a la precisión. Por tanto, se considerará velocidad constante en cada una de las articulaciones, despreciando los tiempos de aceleración y deceleración.

Dicha función de evaluación sólo tiene sentido para caminos formados por celdas libres de colisión (no cabe hablar de tiempo de ejecución, si los robots colisionan en su movimiento). Sin embargo en este problema todas la celdas, tanto de colisión como las libres, deberán poder formar parte del camino coordinado (de hecho el objetivo principal es encontrar una secuencia de celdas de tipo obstáculos, la línea de corte). Para solucionar el problema, se ha optado por dar un tratamiento especial a las celdas de colisión, penalizándolas con un coste adicional al que tendrían si fueran libres. Este coste será diferente para cada celda y se obtiene mediante una estimación heurística de la dificultad que representa evitar una colisión en la que está involucrada dicha celda. Esta estimación heurística está basada principalmente en dos conceptos:

- La naturaleza de la colisión: Normalmente es más fácil evitar una colisión producida con el elemento terminal del robot primario, que colisiones en las que está involucrado un enlace del robot próximo a la base, debido a que en la primera

existe un mayor número de articulaciones cuyo movimiento es capaz de evitar la colisión. Cabe destacar que sólo hay que considerar el tipo de colisión con el robot primario ya que el secundario se comporta de forma pasiva en cuanto a la evitación de colisiones.

- Los obstáculos fijos del entorno o los límites de las articulaciones próximos al camino del robot del robot primario. Un punto en el camino del robot primario rodeado por obstáculos fijos o cerca de los límites de alguna articulación debe tener asociado un coste mayor. La principal dificultad estriba en que la información que se dispone en este sentido es incompleta, ya que una gran parte de los algoritmos para la obtención de los caminos no necesitan la obtención explícita del espacio de las configuraciones.

Teniendo en cuenta estas consideraciones, la función $g(C)$ se ha definido como:

$$g(C)=g(C_{padre})+\max_{i=1,2}\left[\text{tiempo}_i(C_{padre},C)\right]+P(C,rp) \tag{3.18}$$

donde el primer sumando es la función g de la celda padre, el segundo representa el tiempo necesario para alcanzar la celda C desde C_{padre} para cada uno de los robots, y $P(C,rp)$ es la penalización heurística para celdas tipo obstáculo, siendo rp el robot primario. Esta penalización debe ser nula para celdas libres y debe ser lo suficientemente grande para celdas tipo obstáculo que garantice que si existe un camino formado exclusivamente por celdas libres, el algoritmo lo encuentre. Por tanto, se ha establecido que la mínima penalización se corresponda con el tiempo máximo que emplearían los robots si no existieran colisiones entre ellos, es decir:

$$P(C,RP)=\begin{cases} 0 & \text{si } C \text{ es una celda libre} \\ > \sum_{i=1,2} \text{tiempo}_i(\mathbb{C}_0,\mathbb{C}_{max}) & \text{si } C \text{ no es una celda libre} \end{cases} \tag{3.19}$$

donde $\text{tiempo}_i\ (\mathbb{C}_0,\mathbb{C}_{max})$ se define como el tiempo necesitado por el robot i para realizar el movimiento a lo largo de su camino completo. Los valores concretos para la función P dependen de la configuración y del tipo de modificación elegido. Así, en un robot tipo SCARA, una celda que suponga colisión del primer enlace del robot primario y una modificación tipo 1, debe tener un valor muy alto, ya que es prácticamente imposible evitar

dicha colisión. En general, siempre cumpliendo la restricción de la ecuación anterior, los valores de penalización serán más altos a medida que el enlace del robot primario donde se produce la colisión esté más cerca de la base.

Finalmente, la función heurística $h(C)$ será el tiempo invertido desde C hasta la celda final, suponiendo que no existen obstáculos por lo que el movimiento puede realizarse de forma simultánea sin restricciones, es decir:

$$h(C)=\max_{i=1,2}\left[\text{tiempo}_i(C,\mathbb{C}_{max})\right] \tag{3.20}$$

Esta función nunca sobreestima el coste de alcanzar la celda objetivo desde C, por lo que tiene la condición de admisibilidad de A* (Ver apéndice).

También es necesario considerar la obtención de los sucesores de las celdas. Debido a las características de las líneas de corte, es necesario que los sucesores de una celda se traten de distinta forma para celdas libres y celdas obstáculos. Así, mientras que para las primeras, cada celda podrá tener como sucesoras otras tres celdas (la que se encuentra encima, a la derecha y en la diagonal suparior derecha) , cada celda C de tipo colisión sólo tendrán un único sucesor, la celda que se encuentre en la misma dirección en la que se ha alcanzado dicha celda C.

En algunas ocasiones este método requiere la comprobación del estado de colisión de algunas celdas. En efecto, si fue aplicado el método propuesto en apartados anteriores para la obtención del diagrama de coordinación, se desconoce el estado real de todas aquellas celdas pertenecientes al interior de una zona de la clausura-SW de la región de colisión, por lo que ahora puede ser necesario obtenerlo.

3.4.2.2 Búsqueda del nuevo camino en la modificación tipo 1

En este módulo se busca un camino entre los puntos $\hat{\omega}_1$ y $\hat{\omega}_2$ en el espacio de las configuraciones del robot primario, teniendo como obstáculos los fijos del entorno y el robot secundario en la posición fija $\hat{\rho}$. El método utilizado será un procedimiento de búsqueda bidireccional mediante un algoritmo A* que se detalla en el apéndice B de esta tesis, exactamente igual al que se puede utilizar para planificar los caminos originales para cada uno de los robots.

3.4.2.3 Búsqueda del nuevo camino en la modificación tipo 2

Sea una línea de corte comprendida entre las celdas $(\omega_{1\rho},\rho)$ y $(\omega_{2\rho},\rho)$ de forma que ambas celdas sean libres (Ver Figura 3.17-a). La finalidad del módulo es encontrar un camino en el espacio de las configuraciones del robot primario desde el punto $\hat{\rho}$ hasta un punto $\hat{\alpha}$ donde se evite la colisión con el otro robot. Desde el punto de vista del diagrama de coordinación, se pretende ampliar éste con una nueva zona que "rompa" la región de colisión. Para ello se utilizará de nuevo el procedimiento que se describe a continuación.

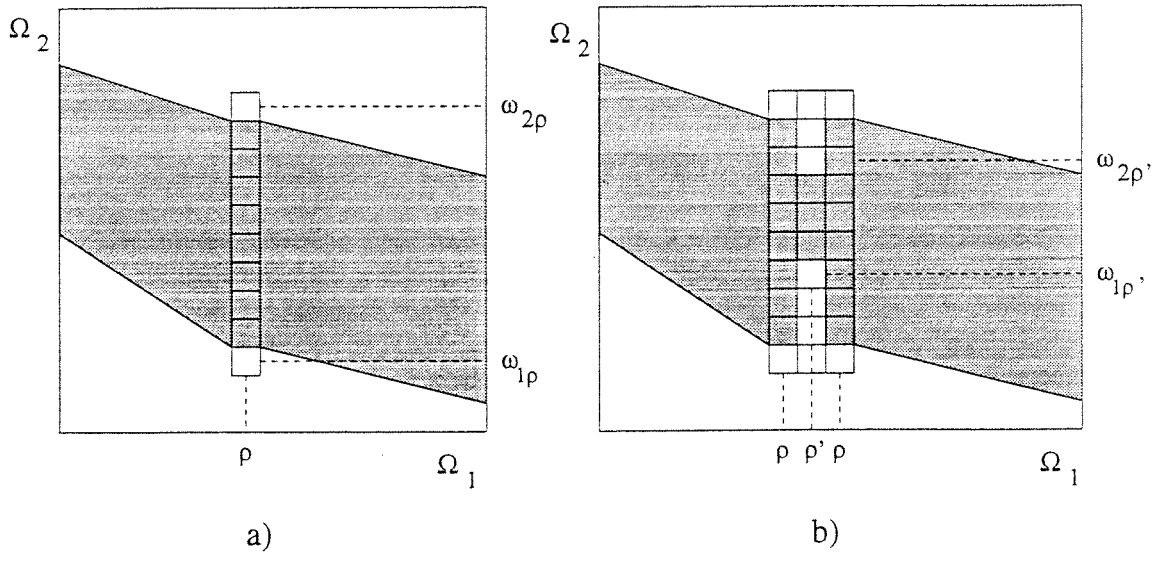

Figura 3.17 Módulo de modificación tipo 2. a) Diagrama de coordinación inicial. b) Diagrama de coordinación tras añadir los tramos ρ-ρ' y ρ'-ρ.

Consiste en un procedimiento de búsqueda de un camino en el espacio de las configuraciones discretizado del robot primario, que trata de encontrar un camino formado por celdas consecutivas partiendo de $\hat{\rho}$. Sin embargo, al ser necesario garantizar que el nuevo diagrama de coordinación tenga solución, cada vez que se considere un nuevo punto del camino será necesario comprobar la incidencia sobre el diagrama resultante. Sea $\hat{\rho}'$ una celda del espacio de las configuraciones del robot primario vecino de $\hat{\rho}$ y considérese el diagrama de coordinación resultante de ampliar el camino con los tramos $\hat{\rho}-\hat{\rho}'$ y $\hat{\rho}'-\hat{\rho}$, que se muestra en la Figura 3.17-b. El diagrama de configuración se habrá ampliado, de forma que para el intervalo ρ' la zona de colisión a evitar estará entre dos nuevos intervalos del camino secundario $\omega_{1\rho'}$ y $\omega_{2\rho'}$. El algoritmo pretende buscar puntos en el espacio de las configuraciones de forma que la amplitud de la banda entre los puntos ω a la que da lugar en el diagrama de coordinación se reduzca, hasta llegar a un punto α en que ésta desaparezca (sería un punto de finalización de la búsqueda). Por tanto, el algoritmo que

se propone buscará un camino desde $\hat{\rho}$ hasta $\hat{\alpha}$, de forma que el diagrama de coordinación resultante de ampliar el camino del robot primario con los tramos $\hat{\rho}$-$\hat{\alpha}$ y $\hat{\alpha}$-$\hat{\rho}$ tenga solución. En el desarrollo del algoritmo, sin pérdida de generalidad, se considerará que el robot primario es el número 1 (por tanto la línea de corte será vertical).

El algoritmo propuesto, que es del tipo A*, presenta algunas particularidades que es necesario considerar. Estas peculiaridades se encuentran fundamentalmente en tres conceptos:

- En las condiciones que debe cumplir un punto para poder ser considerado como *final.*
- En la determinación de los puntos que pueden ser sucesor de uno dado.
- En la función de evaluación.

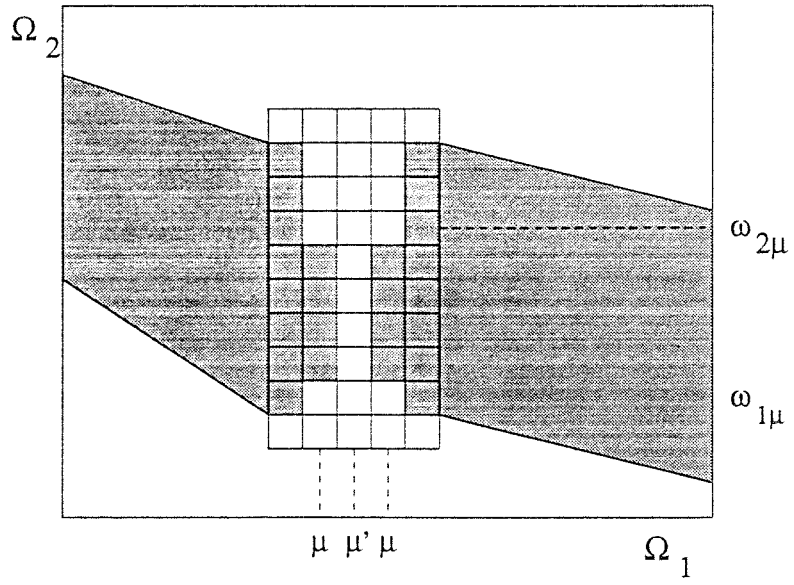

Figura 3.18 Representación de un punto μ' final.

Con relación al primer punto, sea $\hat{\mu}$ un punto del espacio de las configuraciones del robot primario y $\hat{\mu}'$ uno de sus posibles sucesores. Se dice que $\hat{\mu}'$ es un punto final si cumple la siguiente condición (Figura 3.18):

$$\forall i \;/\; \omega_{1\mu} \leq i \leq \omega_{2\mu} \quad \text{se cumple que } (\mu',i) \text{ es una celda libre} \tag{3.21}$$

Con respecto a los puntos en el espacio de las configuraciones que pueden ser sucesores de uno dado, se imponen unas condiciones adicionales con respecto a los sucesores considerados en el apéndice B para la obtención de caminos en el espacio de las configuraciones (recuérdese que se había realizado una discretización del espacio de configuración y que se consideraban dos sucesores por cada grado de libertad, resultado de mover la articulación asociada una cantidad fija en ambos sentidos, dejando fijas el resto de articulaciones). Estas nuevas condiciones están orientadas a impedir la aparición de diagramas de coordinación sin solución. En primer lugar se verán las condiciones a cumplir cuando la línea de corte es única, por lo que sólo hay que realizar una modificación. En este sentido, dado un punto del espacio de las configuraciones del robot primario $\hat{\mu}$ y un posible sucesor $\hat{\mu}'$, se dice que $\hat{\mu}'$ es un *punto válido* respecto a $\hat{\mu}$ si cumple las siguientes condiciones:

a) No colisiona con ningún obstáculo fijo del entorno.
b) Cumple alguna de las dos siguientes condiciones:
   b.1) Es un punto *final*.
   b.2) Si no es final, se deduce que existirán los intervalos del camino del robot secundario $\omega_{1\mu'}$ y $\omega_{2\mu'}$ que marcan el principio y el final de la nueva línea de corte. Dichos puntos deberán permitir la existencia un camino desde la celda inicial $\mathbb{C}_0$ a la celda $(\mu,\omega_{1\mu'})$ y otro de la celda $(\mu,\omega_{2\mu'})$ hasta $\mathbb{C}_{max}$.

La comprobación de la condición b.2 requiere un importante esfuerzo de cálculo, por lo que se ha optado por sustituirlas por unas nuevas condiciones más fáciles de calcular pero a su vez más restrictivas. Estas dos condiciones son:

- Deben existir las celdas libres $(\mu',\omega_{1\mu'})$ y $(\mu',\omega_{2\mu'})$, es decir $\omega_{1\mu'} \geq 1$ y $\omega_{2\mu'} \leq max_2$. La no existencia de una de estas celdas implica una situación tal como la de la Figura 3.19-a, en la que no existe $(\mu',\omega_{1\mu'})$.

- Si $\omega_{1\mu'} \geq \omega_{1\mu}$ y $\omega_{2\mu'} \leq \omega_{2\mu}$

  En este caso siempre hay camino, ya que existe un camino desde $(\mu,\omega_{1\mu})$ hasta $(\mu',\omega_{1\mu'})$, y al ser μ válido, deberá existir camino desde $\mathbb{C}_0$ hasta $(\mu,\omega_{1\mu})$. Un resultado similar se obtiene para el punto $(\mu',\omega_{2\mu'})$

  En caso contrario

Para este caso, más complejo, se utilizará el camino de coordinación previamente calculado en el módulo de selección. Si *CC* es dicho camino de coordinación, y las celdas $(x_1,\omega_{1\mu'})$ y $(x_2,\omega_{2\mu'})$ pertenecen a él, se deberá verificar:

$\forall i,j \,/\, x_1<i<\mu'$ y $\mu'<j<x_2$ se cumple que las celdas $(i,\omega_{1\mu'})$ y $(\omega_{2\mu'},j)$ sean celdas libres. (Ver Figura 3.19-b).

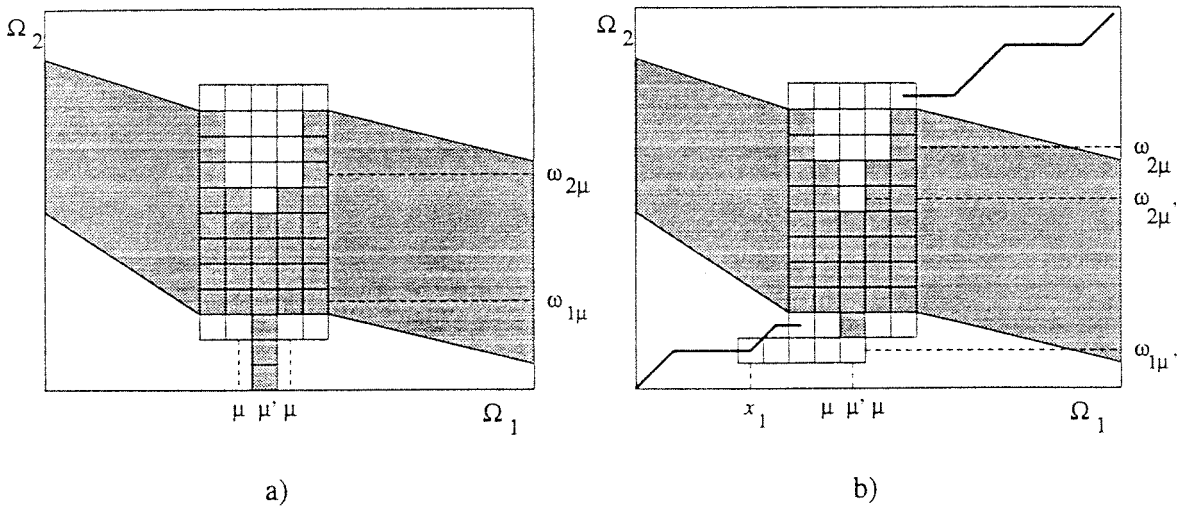

Figura 3.19 a) Representación del punto μ' no válido. b) Condiciones de μ' para ser un punto válido.

Caso de existir más de una línea de corte, hay que tener la precaución de que una modificación no cree un diagrama de coordinación tal que impida aplicar la siguiente. Supóngase un caso general, en que se está resolviendo una modificación, de manera que hay una anterior ya aplicada y una posterior sin aplicar. En primer lugar es necesario realizar una definición previa. Se define la *celda límite superior* de una modificación, como la celda:

$$cls_{\text{mod}}=(\mu_{\max},\omega_{2\mu_{\max}}) \;/\; \omega_{2\mu_{\max}} \geq \omega_{2\mu} \quad \forall\mu \quad \alpha \leq \mu,\mu_{\max} \leq \rho \tag{3.22}$$

La anterior se corresponde con una modificación realizada a partir de una línea de corte vertical. Resultados similares se obtendrían para una línea de corte horizontal. En la Figura 3.20 se muestran algunos ejemplos de celdas límites superiores.

Volviendo a las condiciones de validez con múltiples modificaciones, sea la modificación a realizar entre $(\rho,\omega_{1\rho})$ y $(\rho,\omega_{2\rho})$. Sea cls_{ant} la celda límite superior de la modificación anterior, y C_{pos} la celda inicial de la línea de corte de la modificación posterior. En esta circunstancias, las condiciones son similares a las establecidas para líneas de corte únicas, sólo que en la condición b.2) habrá que garantizar la existencia de un camino desde cls_{ant} hasta $(\rho,\omega_{1\rho})$ y otro desde $(\rho,\omega_{2\rho})$ hasta C_{pos}. Es decir, lo único que habrá que hacer es sustituir el papel de $\mathbb{C}_0$ por cls_{ant} y el de $\mathbb{C}_{max}$ por C_{pos}.

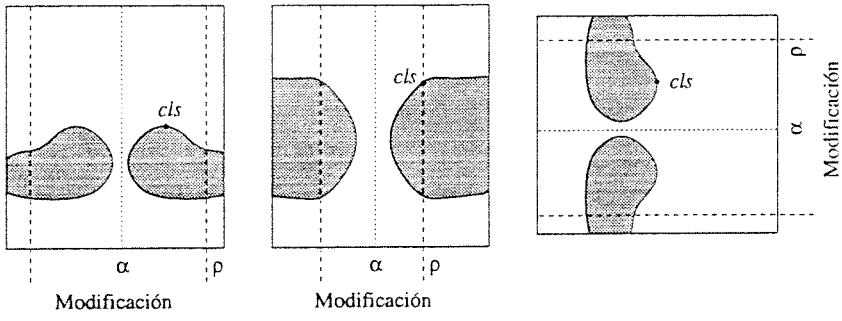

Figura 3.20 Distintos ejemplos donde se muestra la *celda límite superior* en el diagrama de coordinación resultante tras una modificación.

En la aplicación del algoritmo, se considerarán como sucesores de un punto del espacio de las configuraciones a aquellos que sean válidos respecto a dicho punto. El cumplimiento de las condiciones anteriores garantiza que si se encuentra el camino desde $\hat{\rho}$ hasta $\hat{\alpha}$ el diagrama de coordinación resultante va a tener solución.

Finalmente, como función de evaluación heurística se han obtenido buenos resultados para:

$$f(\hat{\mu})=\text{número de celdas entre } (\mu,\omega_{1\mu}) \text{ y } (\mu,\omega_{2\mu}) \tag{3.23}$$

3.4.2.4 Restricciones en la aplicación de los algoritmos

La filosofía de los algoritmos hace que no puedan ser aplicados a todos los diagramas de coordinación que no tengan solución. En efecto, el método exige que las zonas de

colisiones puedan ser atravesadas mediante una línea vertical u horizontal, lo cual no siempre es posible.

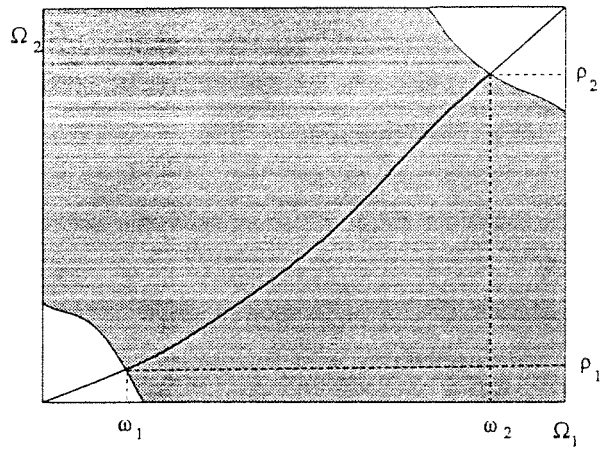

Figura 3.21 Diagrama de modificación al que no pueden aplicarse los métodos de modificación de caminos.

Una posible forma de abordar este tipo de problemas sería utilizando una extensión de la modificación tipo 1, en la que la selección no restringe la forma de atravesar las regiones de colisión a líneas. El problema puede ser resuelto realizando la búsqueda del nuevo tramo que define la modificación en el espacio de las configuraciones del primario entre los puntos $\hat{\omega}_1$ y $\hat{\omega}_2$, pero considerando como obstáculo fijo el volumen barrido por el robot secundario entre las posiciones $\hat{\rho}_1$ y $\hat{\rho}_2$.

Una segunda consideración hace referencia al tamaño del espacio en el que se realiza la búsqueda del nuevo tramo, que en principio es el espacio de las configuraciones completo del robot primario. Sin embargo, dado que el procedimiento no asegura la aparición de una solución y que está enfocado en general a pequeñas modificaciones de los caminos iniciales, para no alargar en demasía el tiempo de búsqueda, se hace aconsejable la limitación del espacio de búsqueda a un subespacio que incluya los puntos $\hat{\omega}_1$ y $\hat{\omega}_2$ en las modificaciones tipo 1 y el punto $\hat{\rho}$ en la modificación tipo 2. Una segunda alternativa es prefijar un tiempo máximo de búsqueda.

3.5 Ejemplos

La eficiencia de las técnicas utilizadas en este capítulo depende en gran medida del caso concreto al que se aplican. Así, la obtención de la clausura SW supondrá una mejora considerable en cuanto al tiempo necesario para la consecución del diagrama de coordinación para casos en los que aparezcan grandes regiones de colisión. Por otro lado, la aplicación del cálculo de colisiones jerárquico será mucho más efectiva frente al cálculo convencional en entornos en los que los robots no se encuentren demasiado próximos, lo que en general dará lugar a diagramas de coordinación con regiones de colisión reducidas.

A título de ejemplo se proporcionan resultados en cuanto a tiempo de C.P.U. necesarios para la obtención del diagrama de coordinación en dos casos extremos. En primer lugar para el diagrama de coordinación de 65×65 celdas mostrado en la Figura 3.22. El segundo se corresponde con un diagrama de coordinación de 68×46 sin ninguna región de colisión. Ambos casos se corresponden con un entorno con dos robots tipo SCARA, en los que sólo se consideran las dos primeras articulaciones. Para estas pruebas se ha utilizado una estación de trabajo RISC-6000 320-H de I.B.M.

En el primer ejemplo, el tiempo necesario para obtener el diagrama de coordinación completo, es decir realizando la detección de colisiones para todas y cada una de las celdas, utilizando siempre el modelo más exacto (CATIA) ha sido de 2m. 38s. Al utilizar el algoritmo para el cálculo de la clausura SW con el mismo modelo exacto, el tiempo se redujo a 59s. Y finalmente utilizando el modelo jerárquico para el cálculo de colisiones y la clausura SW el tiempo empleado fue de 8s.

Para el segundo ejemplo se realizaron las mismas pruebas, obteniendo respectivamente 1m.46s., 1m.48s. y 7s. En este caso se observa que la utilización del algoritmo de la clausura SW no supone ninguna ventaja, pero tampoco supone prácticamente ningún retraso. En ambos casos se pone de manifiesto la importancia de realizar el cálculo de colisiones de forma jerárquica.

En relación a los métodos de modificación de caminos se muestran los resultados de la planificación de dos robots tipo SCARA, en los que se han considerado los dos primeros grados de libertad. La Figura 3.24-1 y Figura 3.24-8 muestran respectivamente los estados iniciales y finales. Los caminos originales para cada uno de los robots consiste en un giro de 180° de la segunda articulación. El diagrama de coordinación obtenido se muestra en la Figura 3.22, que requiere la aplicación de dos modificaciones, apareciendo en el dibujo las correspondientes líneas de corte.

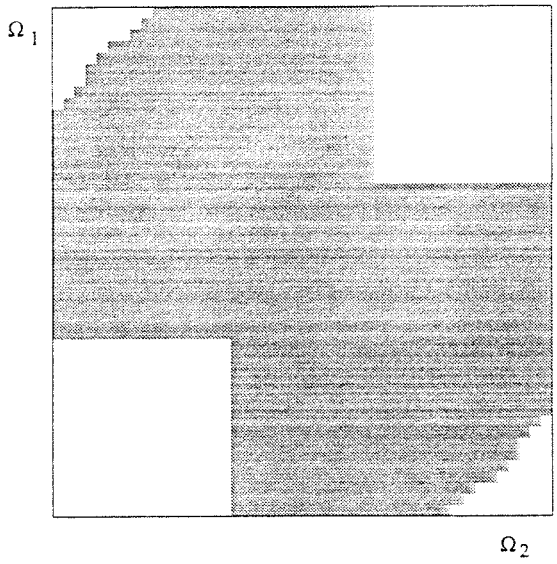

Figura 3.22 Diagrama de coordinación para los dos robots SCARA

En primer lugar se procedió a realizar la modificación tipo 1. En este caso se da la circunstancia de que la realización de la modificación asociada a la línea de corte lc_1 da lugar a un diagrama de coordinación que hace innecesaria la otra modificación. Este resultado no se puede generalizar, y así por ejemplo la aplicación en primer lugar de la modificación lc_2 no daría lugar a un diagrama de coordinación con solución. Para estos casos caben dos soluciones: o bien comprobar si el diagrama obtenido después de cada modificación tiene solución, o bien realizar todas las modificaciones, asegurándose de esta forma la existencia de un diagrama de coordinación con solución. En la Figura 3.23 se muestra el diagrama de coordinación obtenido después de la modificación lc_1, mientras que en la Figura 3.24 se muestra una secuencia de pasos del movimento coordinado a que da lugar. Los puntos numerados de la Figura 3.23 se corresponden con cada una de estas secuencias. La Figura 3.27 presenta el espacio de configuración de cada robot, con una representación de los caminos original y modificado.

Los tiempos aproximados de C.P.U. en una RISC-6000 320-H para cada uno de las fases han sido los que se muestran en la siguiente tabla. El tiempo necesario para la obtención del diagrama de coordinación es tan pequeño debido al tamaño tan considerable que adquiere la clausura SW en este ejemplo. De hecho, las únicas celdas que se evalúan son las celdas libres que aparecen en la esquina superior derecha de la Figura 3.22. Sin embargo esto incide negativamente en el módulo de selección, donde va a ser necesario

evaluar el estado de numerosas celdas no evaluadas anteriormente, de aquí el tiempo necesario para éste módulo.

MÓDULO	Tiempo
Diagrama de coordinación	3 s.
Módulo de selección	17 s.
Modificación lc_1	4 s.
Modificación lc_2[1]	3 s.

A continuación se resolvió el mismo problema aplicando modificaciones del tipo 2. En este caso, la única forma de obtener un diagrama de coordinación con solución es realizando todas y cada una de las modificaciones. En la Figura 3.25 se muestra el diagrama de coordinación resultante después de realizar las dos modificaciones, mientras que en la Figura 3.26 aparece una secuencia de pasos del movimiento coordinado de los dos robots, y en Figura 4.28 una representación de los caminos originales y modificados.

Los tiempos de C.P.U obtenidos fueron en este caso los siguientes:

MÓDULO	Tiempo
Diagrama de coordinación	3 s.
Módulo de selección	17 s.
Modificación lc_1	4 s.
Modificación lc_2	3 s.

[1] Aunque no es necesario realizar esta modificación, se ha evaluado el tiempo de C.P.U. que requiere.

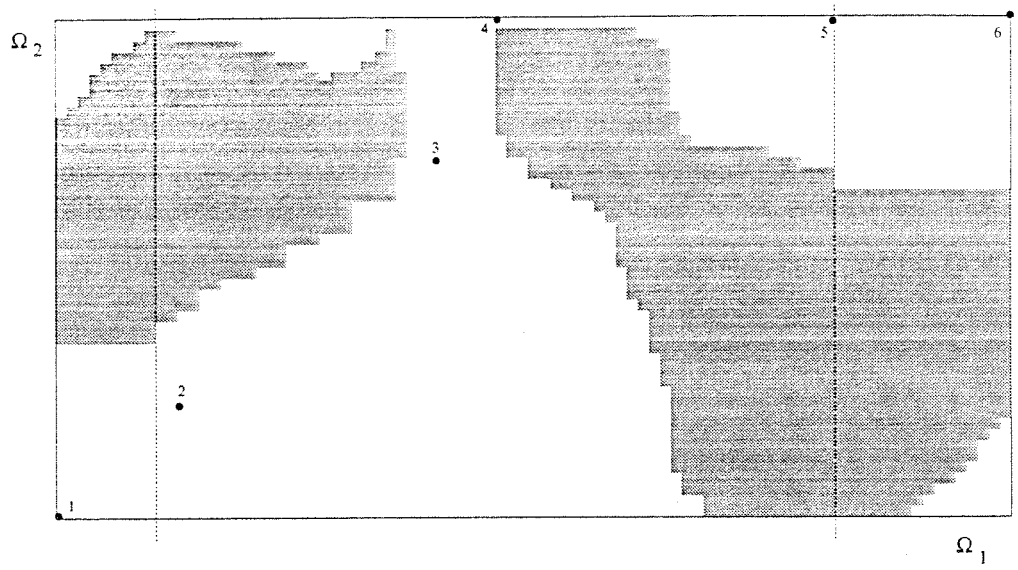

Figura 3.23 Diagrama de coordinación después de la modificación asociada a lc_1

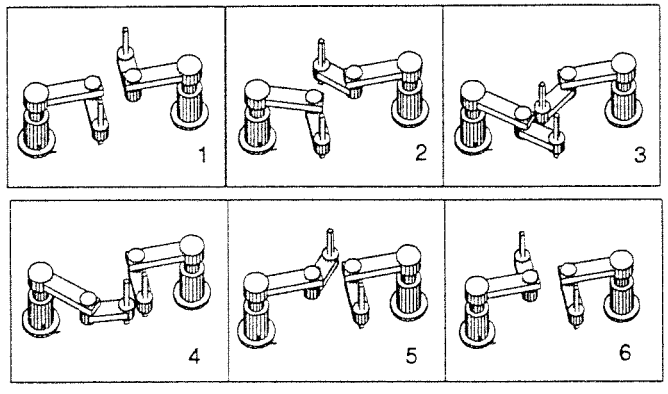

Figura 3.24 Secuencia de pasos después de la modificación tipo 1

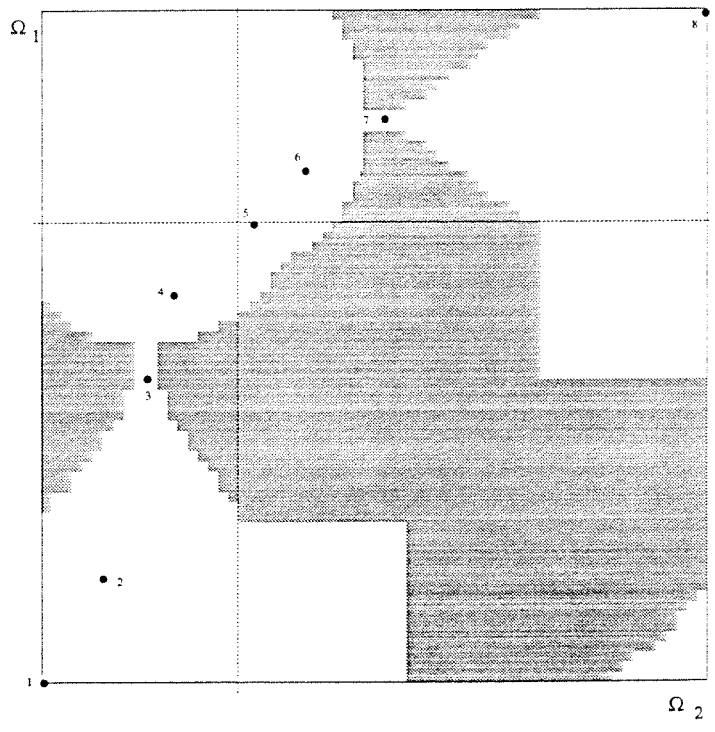

Figura 3.25 Diagrama de coordinación después de realizar las modificaciones tipo 2.

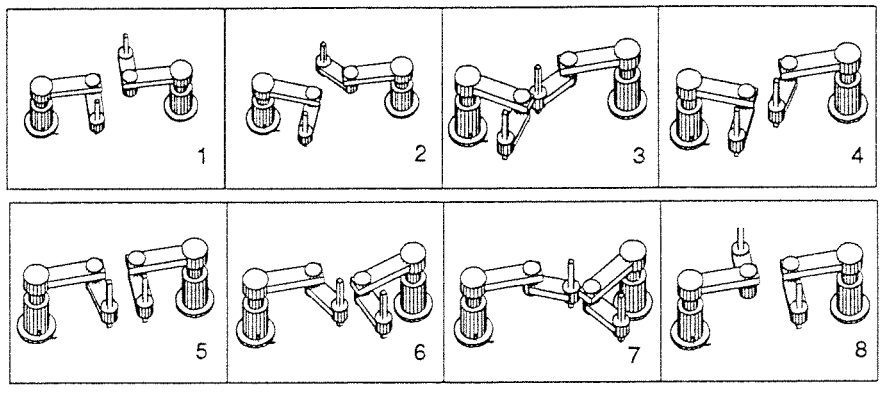

Figura 3.26 Secuencia de pasos después de realizar las modificaciones tipo 2.

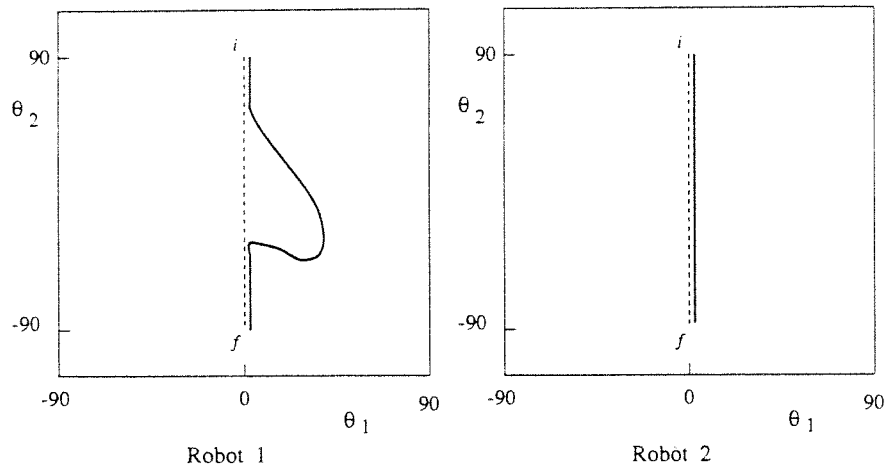

Figura 4.27 Caminos en el espacio de las configuraciones de los robots, tras una modificación tipo 1. Con línea continua el camino modificado y con línea de trazos el camino original.

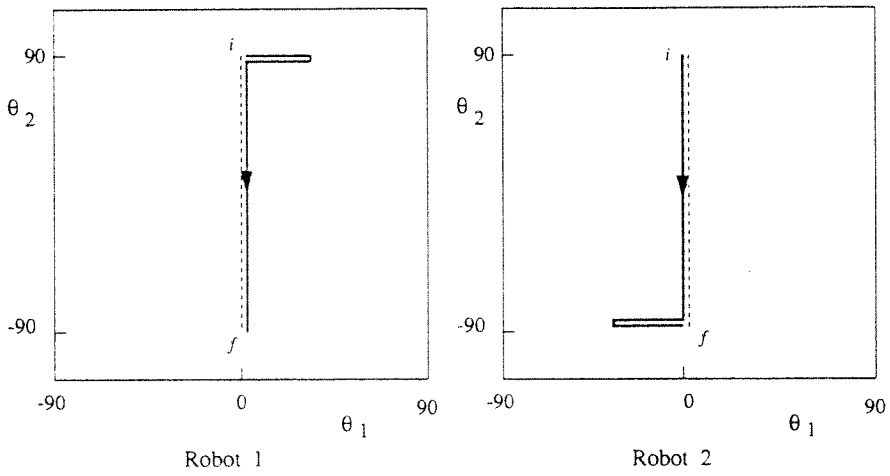

Figura 4.28 Caminos tras una modificación tipo 2. En línea discontinua el camino original, y en línea continua el camino modificado.

Capítulo 4

Generación automática de programas para la coordinación del movimiento de robots

4.1 Introducción

En el presente capítulo se describen una serie de métodos basados en técnicas de planificación desacoplada que permiten generar de forma automática programas para la ejecución del movimiento coordinado de varios robots, utilizando para ello lenguajes de programación comerciales para robots industriales, como por ejemplo VAL II. Como se describió en el capítulo anterior, la utilización de técnicas de planificación desacoplada supone la descomposición del problema global en dos subproblemas. Por un lado, la obtención, de forma independiente para cada robot, de un camino que evite los obstáculos fijos del entorno. El segundo subproblema consiste en establecer la forma en que se recorrerán dichos caminos para evitar colisiones entre los distintos robots. Este capítulo aborda el segundo subproblema, considerando resuelto el primero.

Por tanto, se parte de la existencia previa de un camino para cada uno de los robots, que no colisiona con ninguno de los obstáculos fijos del medio. En este capítulo se supondrá que dichos caminos vienen especificados mediante una sucesión de puntos en el espacio de las articulaciones del robot y se corresponden con secuencias de tramos rectos en dicho espacio. Este tipo de camino tiene la ventaja de poder ser fácilmente especificado en cualquier lenguaje de programación. Sin embargo, cualquier otro tipo de camino que se pueda especificar mediante instrucciones propias de un lenguaje para robots también podría ser utilizado.

A partir de estos caminos se puede construir el diagrama de coordinación, utilizando los procedimientos descritos con anterioridad. Otra suposición será considerar que el

diagrama de coordinación tiene soulución. Caso de que no la tenga, se aplicará alguna de las modificaciones propuestas en el capítulo anterior, o algún método alternativo de planificación. El objetivo es encontrar una trayectoria para cada robot, de forma que su representación en el diagrama de coordinación (camino coordinado) no atraviese las zonas de colisiones de dicho diagrama, es decir, la posición en todo instante de los robots se debe corresponder con celdas libres del diagrama, con lo que se garantiza la no existencia de colisiones entre los robots.

4.2 Sincronización de los robots

El movimiento de cada robot vendrá especificado en un programa realizado en el lenguaje de programación propio de su controlador. Dicho programa deberá incluir, por un lado, las instrucciones necesarias para que el manipulador recorra el camino especificado y por otro lado las instrucciones necesarias para realizar la coordinación con los otros robots. En el trabajo descrito en este capítulo las sincronizaciones necesarias se realizan a nivel de ejecución, es decir, están implementadas en los programas generados, realizándose una coordinación débil. En este apartado se aborda el estudio del tipo se sincronización requerida y la forma de llevarla a la práctica.

Un *punto de sincronización* define una posición en el camino de cada uno de los manipuladores, esto es, un punto en el diagrama de coordinación. Cuando un robot llega a dicho lugar, detiene su movimiento hasta que el resto de los robots lleguen a sus respectivas posiciones. Una vez que todos los robots están en dichos puntos el movimiento se reanuda. La existencia de puntos de sincronización supone una restricción en el movimiento de los robots en el sentido que se obliga a que cualquier camino coordinado incluya dicho punto.

La utilización de puntos de sincronización permite "modelar" los caminos coordinados de forma que, estableciendo una secuencia de ellos, se pretenderá evitar la región de colisión. El número de ellos y su posición en el diagrama de coordinación determinarán las características del movimiento coordinado, y por tanto, como parámetro mas importante, el tiempo total de ejecución de dicho movimiento. En la Figura 4.1 se muestra cómo la colocación adecuada de puntos de sincronización puede producir un movimiento coordinado libre de colisiones.

La principal ventaja de la utilización de puntos de sincronización radica en que pueden ser implementados de una forma muy simple en cualquir lenguaje de programación para robots. En la Figura 4.2 se muestra un fragmento de un programa con estas características para el lenguaje VAL II. Las tres primeras instrucciones se corresponden con

el seguimiento de un camino descrito por una serie de puntos en el espacio de las articulaciones, mientras que las tres últimas realizan la sincronización con un segundo robot.

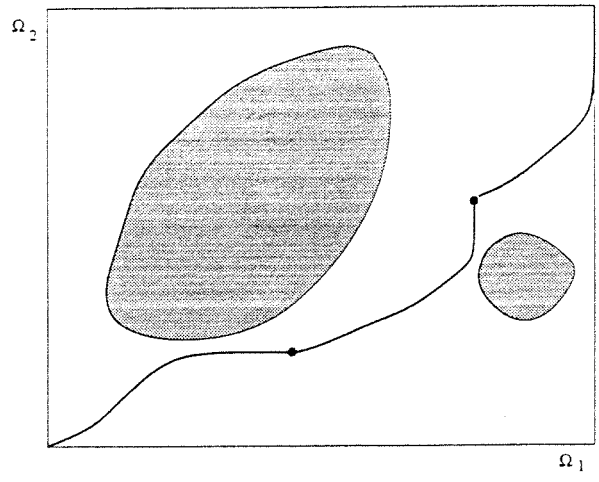

Figura 4.1 Camino coordinado libre de colisiones obtenido mediante la utilización de puntos de sincronización.

```
PROGRAMA ROBOT 1


FOR i=0 TO N
  MOVES pts[i]
END
SIGNAL 1
WAIT SIG(1001)
SIGNAL -1
```

Figura 4.2 Código en VAL II para implementar la coordinación de dos robots.

Otro tema importante es la arquitectura necesaria para llevar a la práctica las técnicas de coordinación propuesta. En este sentido se ha pretendido que fuera lo más simple posible. De cualquier modo, la existencia de puntos de sincronización obliga a establecer una comunicación entre los controladores de los robots. La gran ventaja que presenta este método estriba en la facilidad para llevar a cabo esta comunicación, ya que sólo es necesario conectar una salida digital de cada uno de los controladores, con una entrada digital en cada uno de los demás controladores. La Figura 4.3 muestra un esquema general del sistema.

Por tanto, el problema de especificar el movimiento coordinado de varios robots se puede reducir a la obtención de una *secuencia de puntos de sincronización* sobre el *diagrama de coordinación* de modo que se minimice el tiempo total de ejecución del movimiento de los robots. En el siguiente apartado se proponen algunos métodos para lograr este objetivo.

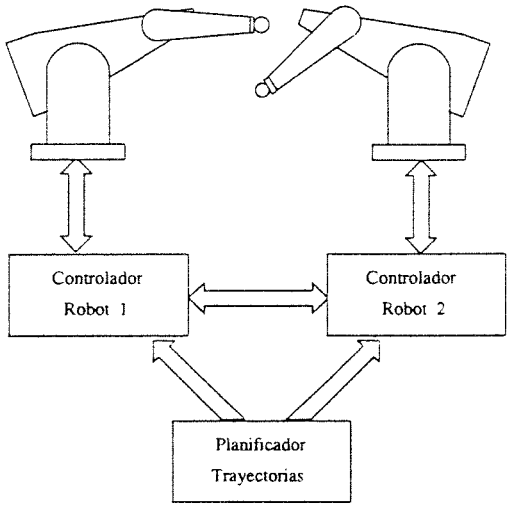

Figura 4.3 Estructura general del sistema de planificación de trayectorias.

4.3 Métodos para la obtención de secuencias de Puntos de Sincronización

En esta sección se presentan dos métodos para obtener una secuencia de puntos de sincronización que minimice el tiempo total de ejecución. La diferencia entre los métodos se encuentra en el distinto grado de seguridad que presentan en relación a la evitación de colisiones. Así, en el primer método los puntos de sincronización se elegirán de manera que

nunca exista colisión entre los robots, sea cual sea la forma en que cada uno realice el movimiento entre dos puntos de sincronización. En el segundo método, la obtención de los puntos de sincronización está basada en un modelo del movimiento de los robots, presuponiendo que la discrepancia entre el comportamiento real de los robots y el modelado es suficientemente pequeña, lo cual no impide que, si la trayectoria real de alguno de los robots se separa en exceso de la obtenida con el modelo, pudieran colisionar los robots.

4.3.1 Método de los rectángulos

Considérese el problema de coordinación del movimiento para dos robots. Sea un rectángulo definido en el diagrama de coordinación en el que se pretende estudiar el movimiento de los robots desde el punto correspondiente a la esquina inferior izquierda hasta la esquina superior derecha de dicho rectángulo. Este rectángulo delimita el espacio de los caminos coordinados posibles, determinados por las distintas trayectorias de los robots. Así, un camino coordinado que recorra el perímetro del rectángulo se corresponderá con un movimiento secuencial de los dos robots, es decir, primero realizaría el movimiento completo uno de los robots, y una vez terminado éste, lo realizaría el segundo. Por otro lado, el paso del camino coordinado por el interior del rectángulo implicará algún tipo de simultaneidad en el movimiento.

Si un rectángulo está formado exclusivamente por celdas libres, implicará que sea cual sea la trayectoria definida para cada uno de los robots, nunca existirá colisión entre ellos en el movimiento considerado anteriormente. Este tipo de rectángulo se denominará *rectángulo libre de colisiones*. Sea una secuencia de rectángulos libre de colisiones conectados de forma que el extremo superior derecho de un rectángulo coincide con el extremo inferior izquierdo del siguiente. Además el extremo inferior izquierdo del primero coincidirá con el extremo inferior izquierdo del diagrama de coordinación y del mismo modo, el extremo superior derecho del último rectángulo coincidirá con el extremo superior derecho del diagrama de coordinación, tal como se ve en la Figura 4.4.

Si se restringe el movimiento de los robots, de modo que el camino coordinado resultante esté siempre en alguno de estos rectángulos libres, se habrá conseguido un camino de coordinación libre de colisiones. Nótese que la implementación de este método en un programa para robots es inmediata, ya que bastará con establecer un *punto de sincronización* en aquellos puntos que se correspondan con los vértices de conexión entre los rectángulos.

El método propuesto busca una secuencia de rectángulos libres en la que se minimice el tiempo en que los robots alcancen sus posiciones finales. Para ello los factores fundamentales a considerar son el número y la posición de los puntos de sincronización,

que dependerán principalmente de la forma geométrica de la región de colisiones en el diagrama de coordinación.

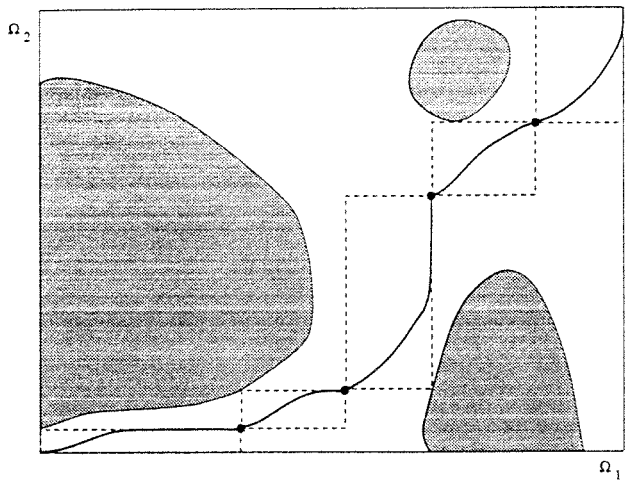

Figura 4.4 Una secuencia de rectángulos libres.

Este procedimiento se puede generalizar al movimiento coordinado de R robots, apareciendo el concepto de *hipercubo.* Sean dos puntos del diagrama de coordinación $x=(x_1,...,x_R)$ e $y=(y_1,...,y_R)$, donde $x_j,y_j \in [1,max_j]$. Formalmente, un hipercubo entre dos puntos del diagrama de coordinación se define como el siguiente conjunto de celdas:

$$H_{x,y}=\{ (z_1,...,z_R) \ / \ z_j \in [x_j,y_j] \ \ 1 \leq j \leq R \} \tag{4.1}$$

Un hipercubo $H_{x,y}$ está *libre de colisiones* si $\forall z \in H_{x,y}$, z es una celda libre de colisiones. Por otro lado, una secuencia de M hipercubos libres de colisiones se define como:

$$SHL=\{H_{x_i,y_i} \ \ 1 \leq i \leq M \ / \ y_j=x_{j+1} \ \ 1 \leq j \leq M-1 \ ; \ x_0=\mathbb{C}_0 \ ; \ y_M=\mathbb{C}_{max}\} \tag{4.2}$$

En el algoritmo que se presenta, es importante conocer el tiempo necesario para la ejecución del movimiento coordinado con las restricciones asociadas a una secuencia de

hipercubos *SHL*, que se denominará *tiempo(SHL)*. Para obtenerlo es necesario disponer de un modelo del generador de trayectorias de los robots que se esté usando. Este modelo será utilizado intensivamente debido a que el algoritmo que se presenta para la obtención de los puntos de sincronización está basado en un proceso de búsqueda, por lo que se debe exigir al modelo la realización de pocos cálculos.

Por todo ello, se ha optado por un modelo basado en los trabajos de Paul [87], Luh [75],[74] y Tondu [119], donde se estudia el movimiento en tramos rectilíneos en el espacio de las articulaciones. Dicho modelo considera velocidad constante, con transiciones suaves entre los tramos, considerando un perfil de aceleraciones contínuos. Este modelo se describe con detalle en el apéndice 3. La utilización de un modelo más complejo, por ejemplo con la utilización de las ecuaciones dinámicas del robot, se ha descartado por el tiempo de cálculo necesario. En cualquier caso, el algoritmo para la obtención de la secuencia de puntos de sincronicación es independiente del modelo.

Por tanto, el problema consiste en encontrar una secuencia de hipercubos libres de colisiones *SHL* que minimice la función *tiempo(SHL)*. Para resolver este problema se han utilizado dos metodologías distintas. La primera, que se describe en este mismo capítulo, utiliza técnicas heurísticas de búsqueda en grafos implícitos. La segunda, desarrollada en el siguiente capítulo, está basada en Algoritmos Evolutivos. Un estudio comparativo de ambos algoritmos se realiza para algunos ejemplos en el Capítulo 6.

4.3.2 Método de la banda de la trayectoria

Sean dos puntos P_1 y P_2 en un diagrama de coordinación (ver Figura 4.5). Si el movimiento se realiza con unas características preestablecidas (aceleraciones y velocidades), existirá un camino de coordinación CC_T asociado a las trayectorias de los robots en el movimiento entre dichos puntos, que se obtendrá utilizando algún modelo de la trayectoria de los robots. Se define la *banda de la trayectoria* como la intersección entre el hipercubo definido entre los dos puntos P_1 y P_2 y el conjunto de celdas del diagrama de coordinación que cumplen la condición de distar menos que una cierta constante K respecto al camino coordinado CC_T.

$$B^{K}_{P_1P_2}=H_{P_1P_2} \cap \{x \ / \ \text{dist}(x,CC_T)\leq K \text{ con } x\in DC\} \tag{4.3}$$

donde **dist**(x,CC_T) se define como el mínimo número de celdas entre x y cualquiera de las celdas que componen el camino CC_T. Si la banda de una trayectoria está formada exclusivamente por celdas libres, se dice que es una *banda libre de colisiones*.

En este segundo método, el objetivo será encontrar la secuencia de puntos de sincronización unidos por bandas libres de colisiones que minimice el tiempo de ejecución del movimiento coordinado. En el método propuesto la optimización se consigue definiendo el número y posición de los puntos de sincronización, suponiendose conocidas las velocidades y aceleraciones de cada tramo. En un trabajo futuro estas magnitudes se considerarán también como variables que afectan a la optimización.

La diferencia fundamental con el método de los rectángulos se encuentra en la forma de establecer cuándo dos puntos de sincronización pueden ser consecutivos. Si en el método de los rectángulos dos puntos son consecutivos atendiendo simplemente a consideraciones geométricas dentro del diagrama de coordinación, en este nuevo método depende además de la trayectoria modelada de los robots. En cualquier caso, el objetivo continua siendo el mismo, obtener una secuencia de puntos de sincronización con un tiempo de ejecución mínimo. Lógicamente la condición que impone el método de la banda de la trayectoria es menos restrictiva que la correspondiente al método de los rectángulos, lo que implica que dos puntos de sincronización que pueden ser consecutivos utilizando el método de la banda, no lo sean con el método anterior. Así, en la Figura 4.5, el punto de sincronización PS_j es sucesor de PS_i según este algoritmo, pero el rectángulo definido por dichos puntos no es libre.

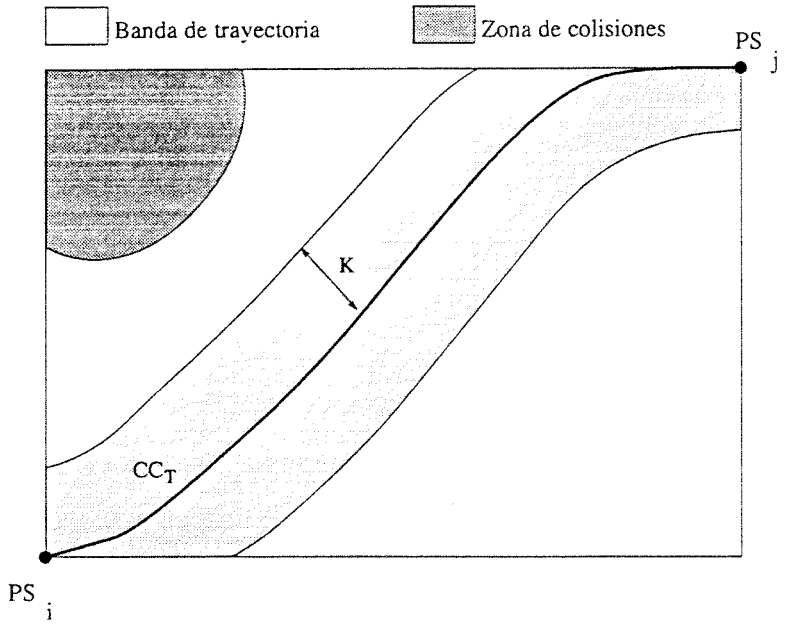

Figura 4.5 Banda de trayectoria.

Es importante destacar que este método está basado en la suposición de que el comportamiento modelado no va a discrepar en exceso del real, lo que supone disponer de un modelo razonablemente exacto del generador de trayectoria de los robots.

La utilización de este nuevo método presenta una serie de ventajas e inconvenientes frente al anterior. La principal ventaja consiste en la obtención de mejores soluciones debido principalmente al hecho de utilizar restricciones más suaves en relación a los puntos de sincronización consecutivos, que hace que las secuencias obtenidas por este método puedan tener menos puntos de sincronización. Así por ejemplo, si el diagrama de coordinación tiene algún obstáculo, el método de los rectángulos obtiene soluciones con al menos un punto de sincronización. En este nuevo método, incluso con obstáculos en el diagrama, pueden aparecer soluciones sin ninguno de estos puntos.

Entre los inconvenientes cabe citar:

- Aumento del espacio de búsqueda. El número de secuencias de puntos de sincronización cumpliendo las restricciones sin colisiones aumenta.

- Mayor complejidad en los cálculos: La obtención de puntos de sincronización consecutivos, que en el algoritmo anterior era cuestión puramente geométrica dentro del diagrama de coordinación, necesita ahora de la utilización del modelo de las trayectorias para la obtención del camino de coordinación entre los puntos de sincronización.

- Seguridad ante las colisiones: Tal como ocurría en la segunda variante del algoritmo anterior, puede ocurrir que debido a discrepancias entre comportamiento del modelo y el real, se produzcan colisiones no previstas entre los robots.

4.4 Coordinación de robots mediante técnicas heurísticas de búsqueda en grafos

En este apartado se desarrolla la implementación de los métodos descritos en los apartados anteriores mediante algoritmos heurísticos de búsqueda en grafos implícitos. En concreto, se utilizarán algoritmos del tipo A^*, utilizando tanto heurísticas admisibles, como semiadmisibles e incluso no admisibles. Una descripción de este tipo de algoritmos y sus propiedades se detallan en el Apéndice A.

4.4.1 Implementación del método de los rectángulos

Como se describió anteriormente, el objetivo de este método consiste en la obtención de una secuencia de hipercubos libres de colisiones, de forma que el tiempo total de ejecución del movimiento coordinado sea mínimo, es decir, un camino formado por puntos del diagrama de coordinación que den lugar a una secuencia de hipercubos libres de colisión.

Para resolver este problema se va a utilizar un algoritmo del tipo A^* que tendrá como espacio de búsqueda el diagrama de coordinación y como nodo inicial la celda $\mathbb{C}_0$. En primer lugar habrá que establecer la función de evaluación, que será la bien conocida:

$$f(C)=g(C)+h(C) \tag{4.4}$$

donde C es una celda, g(C) es el coste desde la celda inicial a C y h(C) es una estimación del coste desde C hasta la celda correspondiente al final de los dos caminos.

Sea C la celda actual. La función de evaluación deberá considerar el tiempo necesario para realizar el movimiento coordinado, suponiendo que en C exista un punto de sincronización. Supóngase que en el camino encontrado desde la celda inicial hasta C existen n-1 puntos de sincronización, de forma que PS_i es el i-ésimo punto de sincronización, PS_0 se corresponde con el vértice inferior izquierdo del diagrama de coordinación ($\mathbb{C}_0$) y el punto de sincronización que se sitúa en C es PS_n. Con estas consideraciones, la función g(C) se define del siguiente modo:

$$g(C)=\sum_{i=0}^{n-1} \text{tiempo}(PS_i,PS_{i+1}) \tag{4.5}$$

donde la función **tiempo**(PS_i , PS_j) se define como el tiempo necesario para realizar el movimiento entre los puntos del diagrama de coordinación PS_i y PS_j partiendo del estado de reposo hasta la detención de los robots en PS_j.

Por otra parte, la función heurística $h(C)$ se define del siguiente modo:

$$h(C)=\text{tiempo}(PS_n, \mathbb{C}_{max}) \quad \text{con} \quad \mathbb{C}_{max}=(max_1,..max_N) \tag{4.6}$$

es decir, el tiempo necesario para realizar el movimiento desde el punto PS_n hasta el extremo superior derecho del diagrama de coordinación, sin considerar las zonas de colisiones definidas en el hipercubo definido entre estos dos puntos, esto es, considerando que dicho hipercubo es libre.

Como se puede comprobar, la función $h(C)$ nunca sobreestima el coste de alcanzar un nodo final; por tanto, tiene la condición de admisibilidad del algoritmo A*. Lógicamente, la función óptima $h^*(C)$ coincidirá con $h(C)$ si el hipercubo definido entre los puntos PS_n y el extremo del diagrama de coordinación esté libre de colisiones. En caso contrario, será necesario restringir el movimiento mediante la inclusión de algún punto de sincronización intermedio que hará que el valor de $h^*(C)$ sea mayor que $h(C)$. Además, la función $h(C)$ definida en la ecuación (4.6) también es monótona (ver Apéndice A). En efecto, sea PS' un sucesor de PS, es decir, dichos puntos cumplen las condiciones para poder ser consecutivos. En estas condiciones, para que la función heurística sea monótona se deberá cumplir:

$$\text{tiempo}(PS, \mathbb{C}_{\max}) < \text{tiempo}(PS, PS') + \text{tiempo}(PS', \mathbb{C}_{\max}) \tag{4.7}$$

En efecto, esta desigualdad se cumple, ya que en el caso más favorable (suponiendo que todos los robots llegan simultáneamente a PS'), es necesario considerar al menos el tiempo de deceleración de los robots para detenerse en PS' y el posterior tiempo de aceleración para continuar el movimiento.

Debido a estas propiedades de la función heurística, una celda puede considerarse final si el hipercubo desde dicha celda hasta el extremo superior derecho del diagrama de coordinación es libre. En efecto, al ser la función heurística monótona, para cualquier nodo expandido C se cumplirá que $g(C)=g^*(C)$, y por ser libre el hipercubo hasta la celda superior derecha del diagrama de coordinación $h(C)=h^*(C)$. Por tanto se trata de una secuencia libre de colisiones, que es la óptima con la restricción de tener como punto de sincronización la celda C. Además, por el hecho de haber sido seleccionada para ser expandida, significa que $f(C) \leq f(C') \ \forall C' \in$ ABIERTOS. Por tanto:

$$f^*(C)=f(C) \leq f(C') \leq f^*(C') \quad \forall C' \in \text{ABIERTOS} \tag{4.8}$$

La principal dificultad de la implementación se encuentra en la expansión de nodos. El número de sucesores potenciales que puede tener cada nodo y que es necesario obtener cada vez que se expande un nodo puede ser muy elevado. En principio, un nodo A' será sucesor de otro A si el hipercubo definido entre dichos puntos es libre. En la práctica el número de puntos A' que cumplen esta condición puede ser elevado, haciendo el proceso de expansión de nodos computacionalmente lento. Para disminuir este número de sucesores se han utilizado varios procedimientos que se describen en los siguientes apartados.

4.4.1.1 Sustitución de las zonas de colisiones por su clausura-SW

La clausura-SW de un diagrama de coordinación fue descrita anteriormente, siendo su objetivo considerar celdas tipo obstáculo, aquellas que aún siendo libres no pueden formar parte de ningún camino coordinado libre de colisiones. En la Figura 4.6 aparecen reflejadas las nuevas zonas de colisión que aparecen al calcular la clausura-SW.

En el capítulo 3 se decribió un método para la obtención de la clausura-SW en el que no es necesario obtener previamente de forma explícita las regiones de colisión.

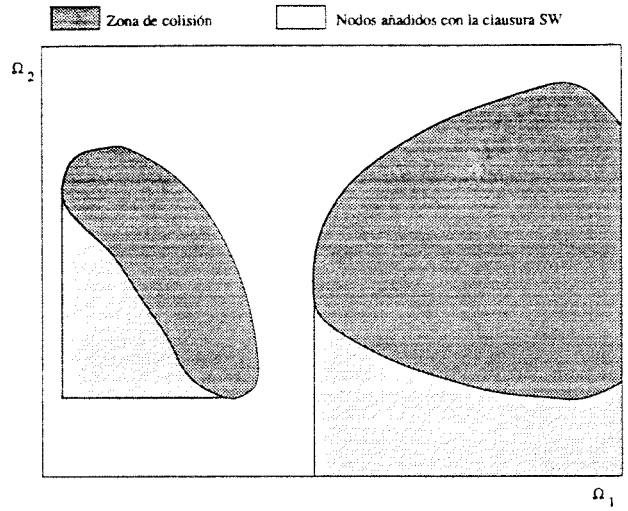

Figura 4.6 Nuevas zonas de colisión al obtener la clausura-SW.

4.4.1.2 Eliminación de puntos de sincronización redundantes

Dados tres puntos de sincronización consecutivos *A*, *B* y *C*, como se ve en la Figura 4.7 para el caso bidimensional, se dice que *B* es un *punto de sincronización redundante* respecto a *A* y *C* si este último punto es sucesor válido de A. Teniendo en cuenta la forma de obtener los sucesores en el método de los rectángulos, *B* será redundante respecto a *A* y *C* si los rectángulos R_S y R_I son rectángulos libres.

La existencia de puntos redundantes supone la intercalación entre dos puntos de sincronización que forman una secuencia válida, de un tercero. Lógicamente, el tiempo necesario para realizar el movimiento entre los puntos *A* y *C* sin restricciones nunca será mayor al necesario para realizarlo con la restricción de pasar por un punto intermedio *B*.

$$\text{tiempo}(A,C) \leq \text{tiempo}(A,B) + \text{tiempo}(B,C) \tag{4.9}$$

Por tanto, en estas condiciones no será necesario considerar *C* como sucesor de *B*, ya que se puede afirmar que previamente se habrá encontrado un camino hasta *C* con un coste menor al camino ahora encontrado. Efectivamente, el expandir el nodo *A*, tuvo que aparecer *C* como su sucesor, por lo que ya existe un camino hasta dicho nodo.

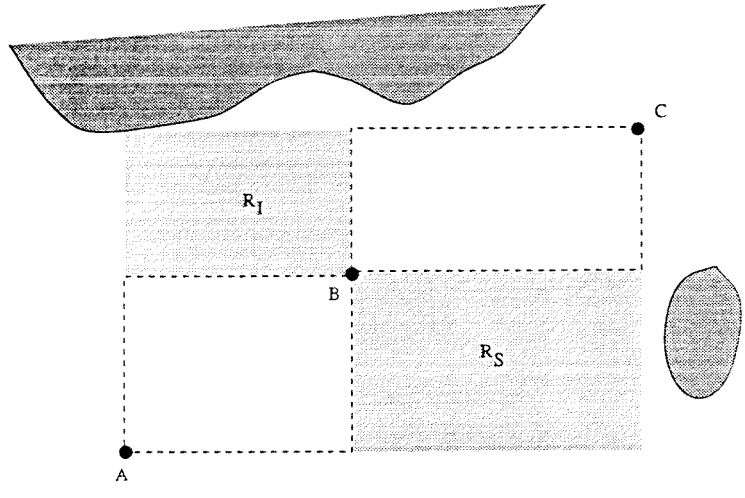

Figura 4.7 Puntos de sincronización redundantes.

4.4.1.3 Eliminación de puntos de sincronización innecesarios

Sea un punto de sincronización A y un sucesor de éste B. Se dice que B es un *punto de sincronización innecesario* respecto a A, si para todo sucesor C de B, se cumple que B es redundante en relación a A y C, es decir, si todos los sucesores de B también lo son de A. Por tanto, si en cualquier caso B va a ser redundante, no es necesario considerarlo como sucesor de A.

En este apartado de presenta un método rápido para obtener puntos de sincronización redundantes sin necesidad de obtener todos sus sucesores. Para esto es necesario realizar unas definiciones previas.

Dada una celda $A=(a_1,...,a_j,...,a_R)$ se define la *distancia a obstáculo* de A en dirección j (DO_A^j), como el número de celdas desde A hasta la celda obstáculo más próxima o en su defecto hasta el final del camino, considerando solamente el movimiento de avance del robot j según su camino, mientras todos los demás permanecen detenidos. Formalmente:

$$DO_A^j=\min\left(\text{dist}(A,A_{col}^j),\text{dist}(A,A_{\max}^j)+1\right) \tag{4.10}$$

donde

$$A_{col}^j=(a_1,..,col_j,..a_R)\in RC \quad \text{y} \quad \forall k_j \text{ con } a_j\le k_j<col_j \quad (a_1,..,k_j,..a_R)\notin RC \tag{4.11}$$

$$A_{\max}^j=(a_1,...,\max_j,...,a_R) \tag{4.12}$$

Sea un punto de sincronización $A=(a_1,...a_R)$ y uno de sus sucesores $B=(b_1,...,b_R)$. Como condición suficiente para que todos los sucesores de B sean a su vez sucesores de A, se tiene que cumplir:

$$\forall j \quad DO_B^j\le\min_{\forall D}(DO_D^j) \quad / \quad D=(d_1,...,b_j,...,d_R) \text{ con } a_k\le d_k<b_k \tag{4.13}$$

En la Figura 4.8 se muestra un punto de sincronización innecesario B respecto a otro A para el caso de un diagrama de coordinación con dos dimensiones. En este caso, la condición anterior se reduce a comparar la distancia a obstáculo desde el punto B según los ejes horizontal y vertical con las celdas que componen el lado derecho y superior respectivamente del rectángulo formado por los puntos de sincronización A y B.

Como comprobación de la condición dada en la ecuación anterior, se considerará el caso bidimensional representado en la Figura 4.8. Sea C un sucesor de B. Por definición, el rectángulo formado por los puntos de sincronización B y C deben ser libres. Por tanto, cualquier sucesor C de B debe cumplir:

$$C \subseteq \left\{ (c_1, c_2) \;/\; b_j \leq c_j < DO_B^j \right\} \tag{4.14}$$

pero teniendo en cuenta (4.13), los rectángulos R_S y R_I son libres, y por tanto B es redundante con respecto a A y C, para cualquier C que sea sucesor de B.

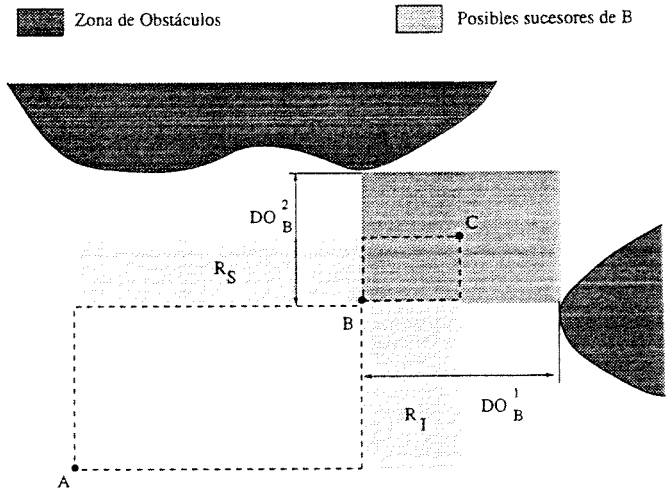

Figura 4.8 Puntos de sincronización innecesarios.

La utilización de estos tres criterios de eliminación de sucesores, hace que el árbol de búsqueda se reduzca drásticamente, haciendo viable el algoritmo. En la Figura 4.9 se muestran los sucesores potenciales de un nodo frente a los que realmente se consideran.

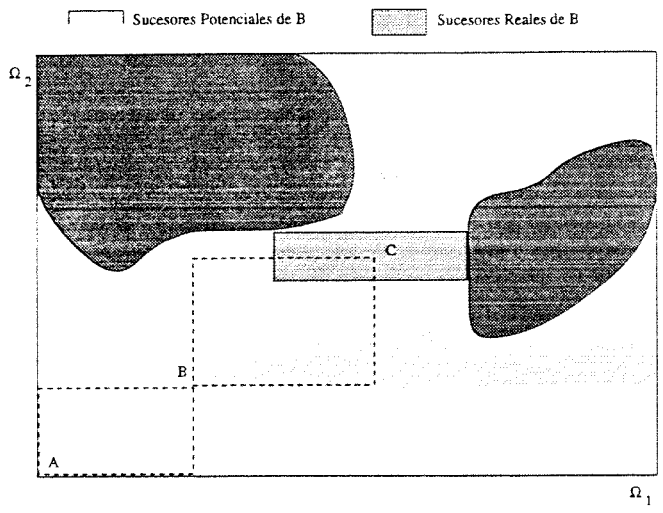

Figura 4.9 Sucesores considerados de un punto de sincronización con relación a todos los sucesores posibles.

Una vez definidos los métodos para reducir el árbol de búsqueda mediante la eliminación de sucesores, se puede definir el algoritmo general para la obtención de una secuencia óptima de puntos de sincronización utilizando el método de los rectángulos. El algoritmo completo puede describirse del siguiente modo:

```
Calcular la clausura SW
Seleccionar la celda inicial
Mientras  la celda no sea  final
      Seleccionar la celda no expandida C que minimice la función
      objetivo
      Para cada sucesor posible C' de C
            Si C' no es redundante ni innecesario respecto a C
                  Considerar  C' como es sucesor de C
            Fin Si
      Fin Para
Fin Mientras
```

4.4.2 Implementación del método de la banda de la trayectoria

El algoritmo propuesto para la resolución del problema de coordinación utilizando el método de la banda de la trayectoria es conceptualmente igual al formulado para el método de los rectángulos. Se utilizará un método basado en el algoritmo A^* utilizando la misma función objetivo, basada en los tiempos de ejecución del movimiento coordinado.

Al igual que en el caso anterior el espacio de búsqueda será el diagrama de coordinación, el nodo inicial será la celda $\mathbb{C}_0$, y una celda será final si el extremo superior derecho del diagrama de coordinación es un sucesor de dicho punto de sincronización.

En la implementación del algoritmo es necesario plantear un nuevo método para la obtención de sucesores, en primer lugar porque se obtienen atendiendo a criterios distintos y en segundo porque alguno de los mecanismos propuestos para la eliminación de sucesores en el caso anterior dejan de ser válidos. En la práctica, la mayor complejidad de los cálculos en la obtención de sucesores y el mayor número de ellos complica la aplicación del algoritmo en problemas de gran tamaño, por lo que se ha optado por limitar la aplicación del algoritmo a la coordinación de dos robots.

En relación a la obtención de sucesores, dado un punto de sincronización *PS*, será necesario obtener el correspondiente camino coordinado (para lo cual será necesario utilizar el modelo anteriormente comentado) y calcular la intersección de la zona de colisiones con la correspondiente banda. Para simplificar los cálculos, se propone un método aproximado que presenta un buen comportamiento para los valores de velocidades y aceleraciones propios de los robots industriales. Sea *PS* un punto de sincronización y *K* la constante que define la anchura de la banda de la trayectoria. En primer lugar se obtiene el camino de coordinación desde *PS* hasta que el primero de los robots llegue al final de su movimiento (CC_{PS}). Por otra parte, se define $PSM_{PS,K}$ como el punto de CC_{PS} más alejado de *PS*, de forma que la banda de la trayectoria entre *PS* y $PSM_{PS,K}$ esté libre de colisiones. Finalmente, se denomina *Camino Coordinado de Profundidad Máxima* de *PS* y banda *K* , que se representará por $CCPM_{PS,K}$, al fragmento del camino coordinado desde el punto *PS* y $PSM_{PS,K}$. En la Figura 4.10 se muestran algunos ejemplos de este camino.

Formalmente, sea $CC_{PS}=\{x^i=(x_1^i,x_2^i) \quad 0\leq i\leq n\}$ con $x^i\in DC$, una sucesión monótona creciente de celdas, de forma que x^i y x^{i+1} sean consecutivas, con $x^0=PS$ y $x^n=(x_1^n,x_2^n)$ con $x_1^n=max_1$ ó $x_2^n=max_2$. En estas condiciones se define $PSM_{PS,K}$ del siguiente modo:

$$PSM_{PS,K}=x^j\in CC_{PS} \;/\; B_{PS\,x^m}\cap RC=\varnothing \quad \text{con} \quad 0\leq m\leq j \quad \text{y} \quad B_{PS\,x^{j+1}}\cap RC\neq\varnothing \qquad (4.15)$$

Como aproximación se supondrá que toda celda x que pertenezca a $CCPM_{PS,K}$ será también sucesor de *PS*. Esta hipótesis será cierta si el proceso de deceleración producido para detener los robots en un punto x perteneciente a $CCPM_{PS,K}$ no modifica de forma cualitativa el camino coordinado. Para los tamaños de celdas utilizados (del orden de 4 grados) y para las aceleraciones existentes en robots industriales, el asumir esta hipótesis no presenta problemas.

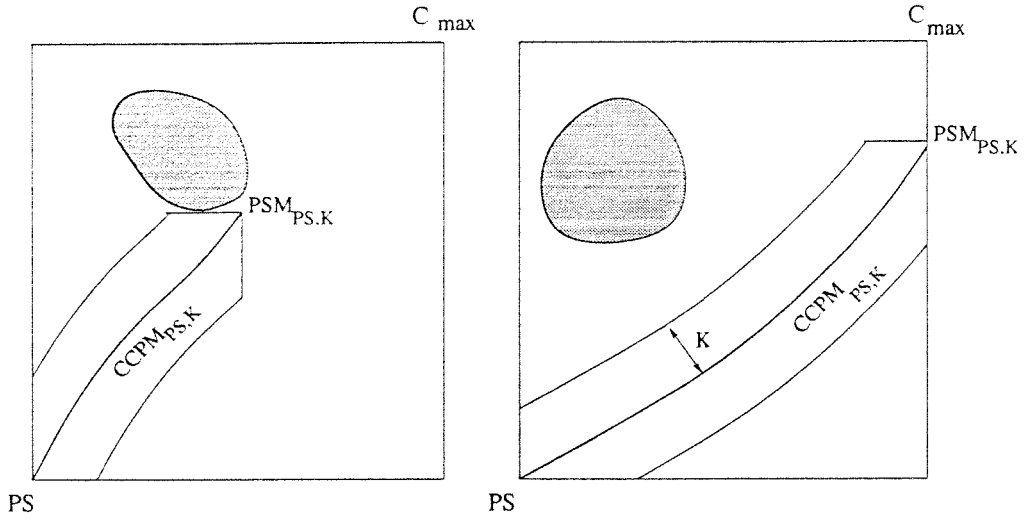

Figura 4.10 Ejemplos de distintos caminos coordinados de profundidad máxima.

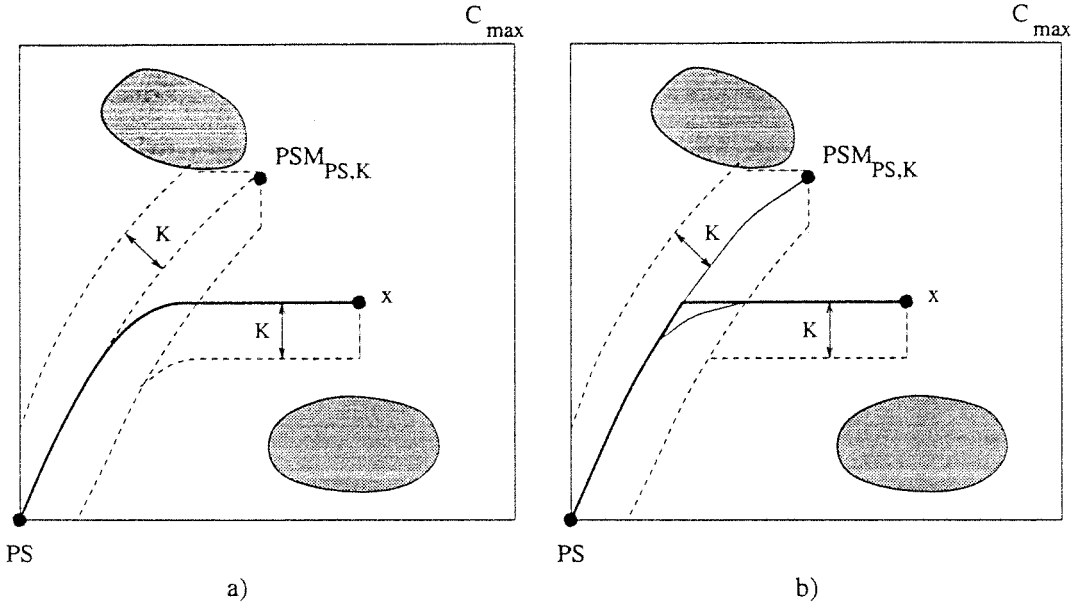

Figura 4.11 Aproximación para la obtención de sucesores.

Además también habrá que incluir como sucesores aquellas celdas que se alcancen recorriendo parte de $CCPM_{PS,K}$ y deteniendo uno de los dos robots como se muestra en la Figura 4.11-a, donde x también debe ser considerado como sucesor de PS. Estrictamente hablando, sería necesario obtener el camino coordinado desde PS hasta x, y a partir de él obtener la banda de la trayectoria. Sin embargo, se realizará la siguiente hipótesis simplificativa: se supondrá que la detención de dicho robot será instantánea con lo que el camino coordinado para alcanzar estos puntos desde PS será tal como aparece en la Figura 4.11-b. De esta forma, la obtención del resto de los sucesores se realiza teniendo en cuenta aspectos puramente geométricos, simplificándose los cálculos. Se distinguirán dos zonas de sucesores: aquellos que resultan al detener el primer robot tras recorrer parte de $CCPM_{PS,K}$, que se corresponde con la zona $R^1_{PS,K}$ en la figura Figura 4.12, y del mismo modo los puntos de la zona $R^2_{PS,K}$ se obtienen de manera similar pero deteniendo el robot 2. Fuera de esta zona no puede haber otros sucesores.

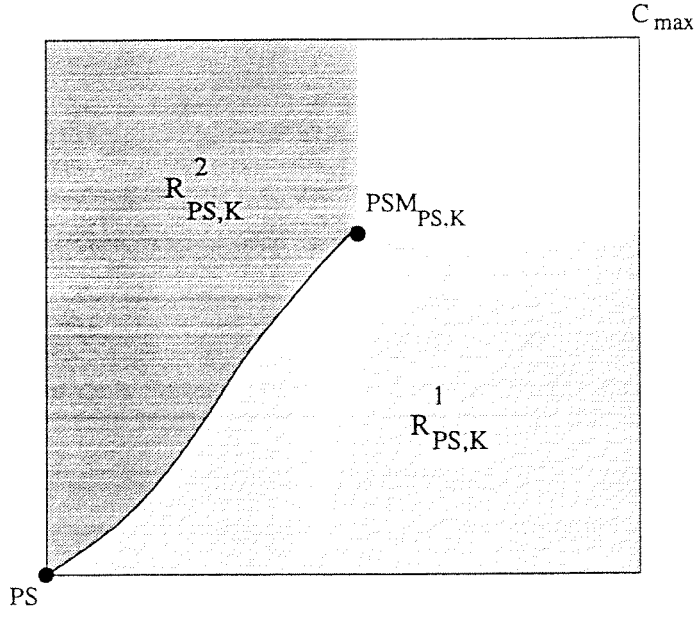

Figura 4.12 Regiones de posibles sucesores.

Además de la pertenencia a estas zonas, un sucesor deberá de cumplir una segunda condición que garantice que en el tramo correspondiente al movimiento de un solo robot (horizontal o vertical), su banda asociada también sea libre. Es la siguiente:

Sea $PS=(ps_1,ps_2)$, $x=(x_1,x_2)\in R^1_{PS,K}$ e $y=(y_1,y_2)\in CCPM_{PS,K}$ tal que $y_2=x_2$. Para que x sea sucesor de PS se deberá cumplir:

$$\forall z=(z_1,z_2) \;/\; y_1\leq z_1\leq x_1,\; x_2-K\leq z_2\leq x_2 \text{ y } z_2\geq ps_2 \text{ se cumple } z\notin RC \qquad (4.16)$$

Si por el contrario $x=(x_1,x_2)\in R^2_{PS,K}$ existirá $y=(y_1,y_2)\in CCPM_{PS,K}$ tal que $y_1=x_1$ de forma que la condición ahora se formula:

$$\forall z=(z_1,z_2) \;/\; y_2\leq z_2\leq x_2,\; x_1-K\leq z_1\leq x_1 \text{ y } z_1\geq ps_1 \text{ se cumple } z\notin RC \qquad (4.17)$$

De nuevo en este caso, la principal dificultad práctica es el gran número de sucesores (mayor incluso que en al caso anterior), con lo cual será de nuevo necesario recurrir a métodos para reducirlos. Evidentemente seguirá siendo posible la sustitución de la región de colisiones por su clausura-SW, ya que este procedimiento sólo elimina zonas del diagrama de coordinación que no podrán pertenecer a ningún camino de coordinación libre de colisiones, y por tanto es independiente de la forma en que se obtenga dicho camino de coordinación.

El procedimiento propuesto anteriormente para la eliminación de Puntos de Sincronización Redundantes deja de ser válido. En este caso sólo se puede aplicar el método general, es decir, dados tres puntos de sincronización consecutivos A, B y C, se dice que B es redundante respecto a A y C si este último es sucesor de A. En cualquier caso, esta comprobación requiere un tiempo considerable de cálculo. Sin embargo, debido a que los sucesores al aplicar el método de la banda de la trayectoria engloban a los sucesores obtenidos según el método basado en los rectángulos, los puntos de sincronización redundantes según este último método también lo seguirán siéndo para el anterior. Por tanto se ha optado por mantener el mismo criterio de eliminación de puntos de sincronización redundantes, aunque no se consideren redundantes ciertos puntos que en realidad lo son.

El método de eliminación de puntos de sincronización innecesarios también deja de ser válido. En este caso los puntos de sincronización que cumplen las condiciones impuestas en las ecuaciones (4.10) a (4.12) no pueden ser eliminados, debido a que no se puede garantizar que todos los sucesores de dichos puntos den lugar a puntos de sincronización redundantes, por lo que es necesario recurrir a nuevas técnicas.

Sea x un punto de sincronización y PS su antecesor. Para comprobar si x es innecesario se utilizará el siguiente método aproximado:

En primer lugar se obtienen $CCMP_{PS,K}$ y $CCMP_{x,K}$. Sean $PSM_{PS,K}=(p_1,p_2)$ y $PSM_{x,K}=(q_1,q_2)$ la última celda de cada uno de los caminos coordinados (Ver Figura 4.13). Una primera condición que se debe cumplir para que x sea innecesario es que alguna de las dos coordenadas de $PSM_{x,K}$ sean menores o iguales que las correspondientes de $PSM_{PS,K}$. En caso contrario, al menos el punto $PSM_{x,K}$ no pertenecería a los sucesores de PS, ya que quedaría fuera de las zonas de posibles sucesores $R^1_{PS,K}$ y $R^2_{PS,K}$. Por tanto se habría encontrado al menos un punto sucesor de x que no lo es de PS.

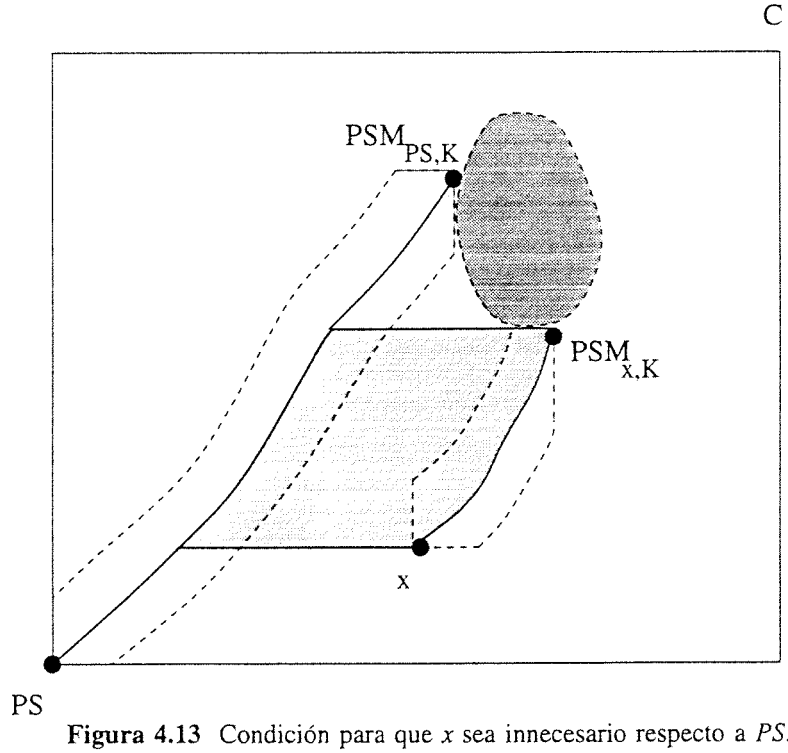

Figura 4.13 Condición para que x sea innecesario respecto a PS.

Sin embargo es necesario establecer una segunda condición para asegurar que todos los sucesores de x también lo son de PS. Ésta consiste en validar que la zona del diagrama de coordinación entre los dos caminos coordinados (gris claro en la Figura 4.13) esté formada exclusivamente por celdas libres. Por tanto, las dos condiciones que se deben verificar para que x sea innecesario respecto a PS se reducen a:

$$\forall y \in CCPM_{x,K} \Rightarrow y \in R^1_{PS,K} \cup R^2_{PS,K} \tag{4.18}$$

Además si $y \in R^1_{PS,K}$ por definición deberá existir una celda $z=(z_1,z_2) \in CCPM_{PS,K}$ con $z_1=y_1$, debiendose cumplir para estas celdas:

$$\forall w=(w_1,w_2) \;/\; z_1 \leq w_1 \leq y_1 \text{ y } w_2=z_2 \Rightarrow w \notin CR \tag{4.19}$$

Una condición simétrica se deberá cumplir para celdas pertenecientes a $R^2_{PS,K}$

4.5 Heurísticas no admisibles

Hasta este punto se han estudiado varias metodologías y algoritmos que aseguran encontrar una solución óptima. En este apartado se realiza un estudio sobre la aplicación de algoritmos de búsqueda no admisibles. Estas técnicas no aseguran encontrar el óptimo, pero se pretende con ellas encontrar soluciones razonablemente buenas reduciendo el tiempo de búsqueda. Estas técnicas se pueden aplicar a cualquiera de los métodos estudiados con anterioridad. Un estudio más detallado de heurísticas no admisibles se expone en el apéndice A.

4.5.1 Asignación de pesos a *g* y *h*

La utilización de pesos transforma la función de evaluación estudiada en los apartados anteriores en:

$$f(N)=(1-w)\cdot g(N)+w\cdot h(N) \tag{4.20}$$

Los algoritmos anteriores se corresponden a la utilización de w=0.5. Para valores inferiores la búsqueda sigue siendo admisible, aunque tendiendo hacia una búsqueda a lo ancho (w=0). Esto da lugar a estrategias que necesitan en general un mayor tiempo de cálculo que las anteriormente descritas, por lo que no aportan ninguna ventaja.

Para valores superiores se obtienen búsquedas más rápidas, que van obteniendo soluciones cada vez peores. En pruebas experimentales se ha comprobado que para valores de w entre 0.55 y 0.6 se encuentran soluciones aceptables (no óptimas en general) conseguidas en un tiempo de cálculo menor que en los algoritmos admisibles.

La mayor ventaja de este método es la facilidad para implementarlo, mientras que el principal problema que se ha encontrado con este algoritmo consiste en la imposibilidad de encontrar una cota para el error en relación a la solución óptima.

4.5.2 Heurísticas semiadmisibles

Dada una constante ϵ, se denomina heurística ϵ-admisible aquella que garantiza que el valor de la solución encontrada no se separa del valor óptimo en una cantidad mayor que ϵ. Se han estudiado dos métodos: Ponderación dinámica y el algoritmo A_ϵ^*. Estos dos métodos se describen con detalle en el apéndice A.

La ponderación dinámica consiste en asignar pesos a las funciones f y g. Estos pesos se iran modificando a lo largo de la búsqueda, de forma que en el peso de la función h va aumentando a medida que lo hace la búsqueda. El función objetivo utilizada propuesta en [93] es:

$$f(N)=g(N)+h(N)+\epsilon\left[1-\frac{d(N)}{D}\right]h(N) \tag{4.21}$$

donde $d(N)$ es la profundidad del nodo N y D es una cota máxima de la profundidad de la solución obtenida. Este método no se ha desarrollado debido a la dificultad de encontrar una buena estimación de D de forma rápida. Sin embargo, para ver la eficacia del método se han hecho estudios para ejemplos en los que se conoce el valor de D, obteniendo en general mejores resultados que en el algoritmo A_ϵ^* que se describe a continuación. En los resultados para los ejemplos que se presentan en próximos apartados se pone de manifiesto este hecho. Sería pues interesante la búsqueda de un método eficiente para obtener una estimación de dicha cota superior.

El algoritmo A_ϵ^*, comentado con más extensión en el apéndice A, es una variación del algoritmo A*, que no asegura la obtención de la solución óptima, pero sí que el valor de la solución obtenida no supera en una cierta constante ϵ el valor óptimo.

El algoritmo es fundamentalmente igual al A*, diferenciándose en la forma de seleccionar el nodo a ser expandido. En este caso no se selecciona el que presente una

mejor función objetivo, sino el que se estime que encontrará una solución de una forma más fácil, de entre aquellos nodos cuya función objetivo no se separe de la mejor encontrada en cada momento un valor superior a ϵ.

Para la aplicación de este algoritmo, es necesario obtener una nueva función heurística h_F que seleccione entre los mejores nodos, aquel que se expandirá. El objetivo de dicha función será evaluar el coste computacional estimado para llegar a un nodo final.

Para la implementación del algoritmo se han utilizado las mismas funciones *f*, *g* y *h* que en el algoritmo A*. Para la nueva función heurística se ha tomado $h_F=h$ (nótese que la función *h* proporciona valores más pequeños a medida que la celda se va acercando a la celda final y por tanto, de alguna forma estima el esfuerzo de cálculo necesario para alcanzar el nodo final). Esta función presenta la ventaja de no necesitar cálculos adicionales para la obtención de una nueva función.

Las experiencias realizadas con este algoritmo de búsqueda en el problema aquí tratado han demostrado que se trata de un procedimiento bastante "conservador". En general las soluciones obtenidas distan mucho de acercarse al límite permitido, lo cual es contrario al objetivo deseado de encontrar soluciones rápidas. De hecho en general, se ha comprobado que existen soluciones con un valor mayor dentro de los límites tolerados que necesitan menos tiempo de cálculo.

4.6 Ejemplos

Un este apartado se muestran los resultados obtenidos al aplicar los distintos algoritmos a tres ejemplos. En la Figura 4.14 se muestran los diagramas de coordinación de dichos ejemplos. Los dos primeros se corresponden con la planificación del movimiento de dos robots tipo SCARA, mientras que el tercero pertenece al movimiento de dos robots PUMA. La discretización realizada es de cuatro grados por celda, y los tamaños de los distintos diagramas de coordinación son de 82×68, 65×65 y 105×80 respectivamente. En la Figura 4-15 se muestran las posiciones iniciales y finales para este último ejemplo.

En las tablas siguientes se muestran, para las distintas pruebas realizadas, el tiempo de ejecución del movimiento coordinado para la solución encontrada (un asterisco indica solución óptima), el número de puntos de sincronización en la solución y el tiempo de C.P.U. necesario para encontrar la solución en una estación de trabajo RISC-6000 320H. Dichos tiempos corresponden exclusivamente al algoritmo de búsqueda, sin considerar el tiempo de obtención del diagrama de coordinación. En la Tabla 4.1 se muestran los resultados obtenidos para los ejemplos anteriores, utilizando el método de los rectángulos.

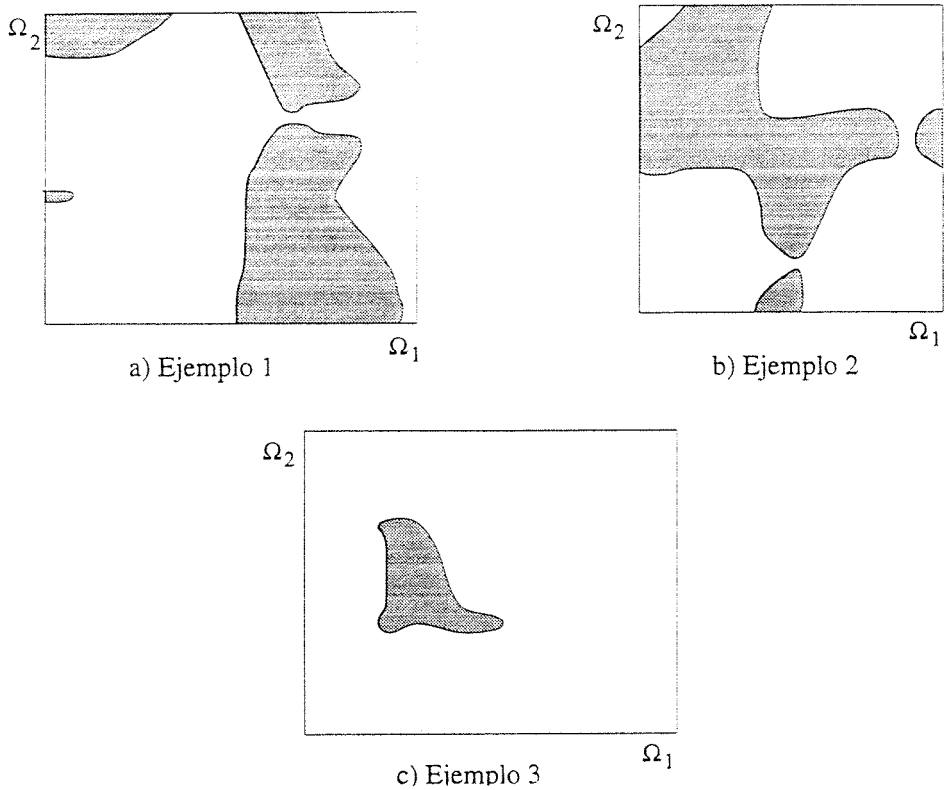

Figura 4.14 Diagramas de coordinación correspondientes a las aplicaciones realizadas.

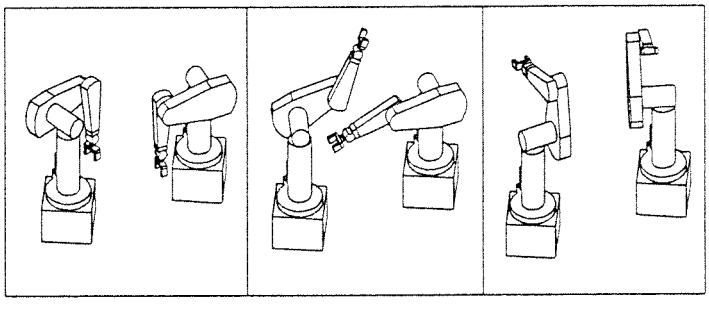

Figura 4.15 Posiciones inicial, en el punto de sincronización y final de los robots PUMAS del ejemplo 3.

En dicha tabla se comprueba que el tiempo de C.P.U. es totalmente dependiente del ejemplo tratado, sobre todo del tamaño del diagrama de coordinación y de la posición y forma de las zonas de colisión. Cabe destacar que el ejemplo 1 es un caso particularmente desfavorable debido a la posición de las zonas de colisión. En la mayoría de los ejemplos tratados, para diagramas de coordinación de tamaño similar a los aquí presentados, los tiempos de C.P.U. oscilan entre 1 y 10 segundos.

Tabla 4.1 Método de los rectángulos

Ejemplo	Resultado (seg.)	N° P.S.	C.P.U. (seg.)
1	5.2505*	4	177.84
2	7.4365*	5	15.06
3	3.2986*	1	7.17

La siguiente prueba trata de valorar la importancia de calcular previamente la clausura-SW del diagrama de coordinación. En la Tabla 4.2 se muestran los tiempos de C.P.U. para los mismos ejemplos y utilizando el mismo algoritmo pero sin obtener previamente la clausura SW de los diagramas de coordinación. Como se puede comprobar se obtienen tiempos de C.P.U. considerablemente superiores, sobre todo en el primer ejemplo.

Tabla 4.2 Método de los rectángulos sin clausura SW

Ejemplo	Resultado (seg.)	N° P.S.	C.P.U. (seg.)
1	5.2505*	4	533.12
2	7.4365*	5	25.73
3	3.2986*	1	10.44

En la Tabla 4.3 se exponen los resultados del método de los rectángulos cuando se ha utilizado una heurística no admisible basado en la utilización de pesos en la función objetivo. Los mejores resultados se obtienen para valores de w comprendidos entre 0,55 y 0,6. Para valores superiores se obtienen soluciones rápidas pero alejadas del óptimo. La utilización de w=0.6 proporciona un excelente método para obtener unos resultados aceptables con tiempos de cálculo reducidos frente al necesario para obtener el óptimo.

Tabla 4.3 Método de los rectángulos con heurística no admisible utilizando pesos

Ejem.	w	Result. (s.)	Nº P.S.	C.P.U. (s)
1	0.55	5.2506*	4	119.72
	0.6	5.2506*	4	11.33
	0.7	5.3855	4	3.82
2	0.55	7.4365*	6	9.57
	0.6	7.5221	7	6.71
	0.7	7.7924	8	6.18
3	0.55	3.2986*	1	6.77
	0.6	3.2986*	1	6.39
	0.7	3.3656	2	5.72

Los resultados obtenidos con heurísticas semiadmisibles se muestran a continuación. (Tabla 4.4). En primer lugar se muestran algunos resultados utilizando ponderación dinámica. Como se comentó anteriormente, este algoritmo no se ha implementado completamente por no haberse obtenido de forma general una buena estimación para el factor D. En la tabla se muestran los resultados para un valor de D igual al doble de la profundidad del árbol para la solución óptima.

Por otro lado, los resultados de la utilización de la heurística semiadmisible A_ϵ^* se muestran en la Tabla 4.5. Los resultados en las dos búsquedas semiadmisibles demuestran que los valores de las soluciones obtenidas distan mucho de los topes permitidos. Así para ϵ=0.1 segundos siempre se ha encontrado el óptimo, con unos tiempos de C.P.U. algo inferiores a los requeridos por el algoritmo admisible. Para ϵ=1 segundo en ninguno de los

ejemplos tratados se ha obtenido una separación del óptimo mayor de 4 décimas de segundo. También se pueden comprobar los mejores resultados obtenidos para la ponderación dinámica tanto en resultados como en tiempo de C.P.U. a pesar de utilizar una estimación de D no muy ajustada.

Tabla 4.4 Método de los rectángulos con ponderación dinámica

Ejem.	ε	Result. (s.)	N°P.S	D	C.P.U. (s)
1	0.1	5.2506*	4	8	161.40
	0.5	5.2506*	4	8	14.44
	1	5.3855	4	8	4.41
2	0.1	7.4365*	5	10	10.50
	0.5	7.5210	7	10	6.63
	1	7.6768	8	10	6.17
3	0.1	3.2986*	1	2	6.96
	0.5	3.2986*	1	2	6.66
	1	3.3656	1	2	6.40

Finalmente en la Tabla 4.6 se muestran los resultados obtenidos al aplicar el método de la banda de la trayectoria. Se presentan los resultados en función de la anchura de la banda que viene especificado en número de celdas. Se observan en general resultados bastante mejores a los obtenidos por el método basado en rectángulos. En general, el tiempo invertido para obtener una solución es mayor en este método debido a la mayor complejidad de los cálculos. Sin embargo se observa en la tabla que a veces ese tiempo es inferior al correspondiente utilizado por el algoritmo basado en rectángulos. Debido al mayor número de sucesores que en general presenta cada punto de sucesor, es posible encontrar soluciones que requieren una menor profundidad en el árbol de búsqueda, y por tanto es posible que requieran expandir menos nodos.

Tabla 4.5 Método de los rectángulos con el algoritmo A_ϵ^*

Ejem.	ε	Result. (s.)	N° P.S.	C.P.U. (s)
1	0.1	5.2506*	4	175.61
	0.5	5.2506*	4	168.05
	1	5.3855	4	4.15
2	0.1	7.4567	5	14.57
	0.5	7.4567	5	14.21
	1	7.5909	8	9.35
3	0.1	3.2986*	1	6.93
	0.5	3.3656	2	5.96
	1	3.3656	2	5.90

Tabla 4.6 Método de la banda de la trayectoria

Ejem.	Banda	Result. (s.)	N° P.S.	C.P.U. (s)
1	1	4.8349	1	26.02
	5	5.0818	2	17.43
	10	5.2505	4	194.80
	20	5.2505	4	202.01
2	1	6.7312	2	57.66
	5	7.0223	4	88.18
	10	7.2100	5	150.12
	20	7.2100	5	205.02
3	1	3.1400	1	45.68
	5	3.2232	1	59.81
	10	3.2986	1	114.24
	20	3.2986	1	117.71

Capítulo 5

Generación automática de programas y Algoritmos Evolutivos

5.1 Algoritmos Evolutivos

5.1.1 Introducción

Los algoritmos de optimización global que imitan ciertos principios que se presentan en la naturaleza, tales como el *Enfriamiento Simulado, Redes Neuronales* y *Computación Evolutiva* han demostrado su utilidad en diversos dominios. La Computación Evolutiva está basada en principios biológicos: la evolución natural y la supervivencia de los individuos más aptos [52],[41]. Los algoritmos basados en estas observaciones se denominan *Algoritmos Evolutivos.*

Los *Algoritmos Evolutivos* son una familia de modelos computacionales basados en los procesos evolutivos que se presentan en la naturaleza. Estos algoritmos intentan simular los mecanismos de evolución natural que se producen en una población de individuos capaces de reproducirse. Estos individuos presentan entre ellos algunas diferencias que hacen que algunos estén más adaptados a las condiciones de vida que otros. Los individuos más adaptados son los que más se reproducen y más tiempo sobreviven. A lo largo del tiempo y de las generaciones, las características de los individuos de la población cambian hacia una mejor adaptación al medio.

Basados en estos principios de evolución natural surgen varias técnicas de aplicación a la resolución de problemas. Todos ellos comparten un mismo principio: la simulación de

la evolución de estructuras (*individuos*) componentes de una *población* a lo largo de sucesivas *generaciones*, mediante la aplicación de *operadores genéticos*. Cada uno de estos individuos representa una solución al problema. La bondad de cada una de estas soluciones con respecto a lo requerido (frecuentemente optimización de parámetros) se define mediante la *función de adaptación*.

A grandes rasgos, los algoritmos evolutivos parten de un conjunto inicial de individuos, obtenidos en general de forma aleatoria. En cada iteración (generación) se evalúa la adaptación de cada uno de estos individuos, posteriormente se selecciona un conjunto de individuos de la población de la última generación para ser reproducidos de acuerdo con su función de adaptación. Finalmente a partir de estos individuos se obtiene la siguiente generación (descendencia) mediante la aplicación de diversos operadores genéticos. La Figura 5.1 muestra la estructura general de los algoritmo de este tipo.

Algoritmo AE

- *Crear aleatoriamente una población inicial de individuos.*
- *Evaluar la población inicial.*
- *Mientras no se cumpla el criterio de terminación*
    - *Seleccionar una subpoblación para generar la descendencia.*
    - *Recombinar algunos de estos individuos.*
    - *Perturbar alguno de los individuos*
    - *Evaluar la función de adaptación de estos individuos*
    - *Seleccionar los individuos de la descendencia*
- *Fin Mientras*

Figura 5.1 Pseudocódigo de los Algoritmos Evolutivos

Los dos operadores más frecuentemente utilizados son la *Recombinación* o *Cruce* y la *Mutación*. Un cruce consiste en obtener un nuevo individuo a partir de las características de otros dos (*padres*). La forma de implementar un cruce depende principalmente de la estructura que defina al individuo. En la Figura 5.2 se muestra el tipo de cruce más característico, que se utiliza cuando los individuos están formados por vectores.

Una mutación consiste en alterar alguna componente de uno de los individuos de una forma más o menos leve. La Figura 5.3 muestra una posible mutación para una estructura del mismo tipo.

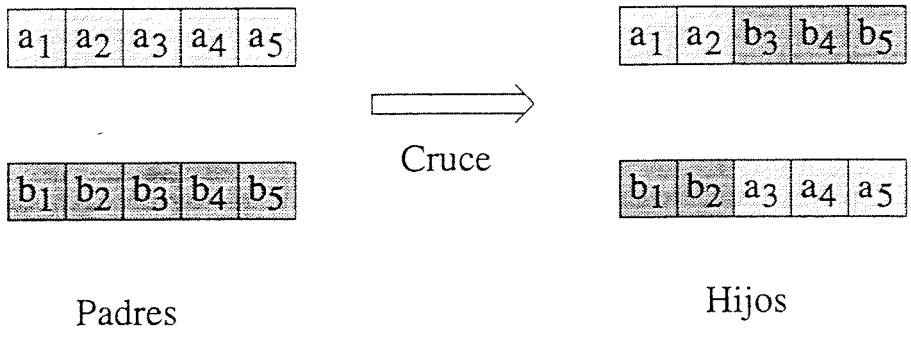

Figura 5.2 Cruce de dos individuos cuya estructura es un vector, dando lugar a dos hijos

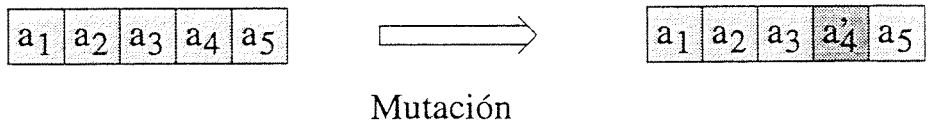

Figura 5.3 Mutación de un individuo definido por un vector

Diferentes variantes de Algoritmos Evolutivos han surgido, algunos de ellos desarrollados simultánea pero independientemente, entre ellos cabe destacar los siguientes:

- *Algoritmos Genéticos:* Fueron desarrollados por Holland [52] en Estados Unidos durante los años sesenta, siendo utilizados inicialmente para optimización discreta. Posteriormente De Jong [22] continuó los trabajos, realizando una aplicación para dominios continuos. La representación de los individuos se realiza mediante cadenas de bits, siendo el *cruce* el operador fundamental, frente a un papel más secundario que representa la *mutación*.

- *Estrategias Evolutivas:* Aunque en cierto modo similares a los Algoritmos Genéticos, las Estrategias Evolutivas aparecen también durante los años sesenta en Alemania, gracias a los trabajos de Rechemberg [97]. Los trabajos iniciales de Rechemberg, aplicados a ciertos problemas hidrodinámicos, no contemplaban el concepto de población de individuos. Un sólo individuo iba evolucionando mediante la aplicación de mutaciones. Posteriormente, las Estrategias Evolutivas fueron extendidas a poblaciones de más individuos por Schwefel [104] incluyendo además nuevos operadores como la *recombinación* o *cruce*. Como características fundamentales de estos algoritmos está el uso de vectores reales para representar individuos, y el uso del operador de *mutación* como operador principal. Este operador también se diferencia del similar en Algoritmos Genéticos en que la mutación se realiza según una distribución de probabilidad normal asociada a cada individuo, y que también evoluciona como una parte más del individuo.

- *Programación Evolutiva:* Tiene también su origen en los años sesenta en una serie de trabajos realizados por Fogel [30] en los Estados Unidos, realizados también de forma independiente de los anteriores, con los que pretendía crear máquinas inteligentes. Al igual que en las Estrategias Evolutivas, la Programación Evolutiva también representa los individuos mediante valores reales, utiliza la mutación como operador genético principal y estas mutaciones también siguen una distribución de probabilidad normal. La diferencia fundamental con los anteriores consiste en la no utilización de un operador de *cruce*, así como diferencias en la aplicación de otros operadores genéticos y en la obtención de la función de adaptación. Una comparación experimental para distintas funciones [1] muestra un comportamiento superior para las Estrategias Evolutivas frente a la Programación Evolutiva.

Desde el campo de los Algoritmos Genéticos surgieron otras dos grandes líneas de investigación:

- *Programación Genética:* El paradigma de la programación genética [61],[43] difiere bastante de las anteriores variaciones, sobre todo en la representación de los individuos. En este caso, los individuos son programas que resuelven el problema que se plantea. Estos individuos son representados mediante árboles sintácticos, donde los nodos interiores son operadores y los nodos hoja son constantes. El

operador fundamental es el *cruce*, donde dos individuos se recombinan intercambiando subárboles.

- *Sistemas de Clasificación:* Fueron creados por Holland como una aplicación de los Algoritmos Genéticos. Los Sistemas de Clasificación son sistemas de aprendizaje, capaces de aprender cadenas de reglas sintácticamente simples, utilizadas para regir su comportamiento en un entorno arbitrario [41].

El problema que se pretende resolver en esta tesis mediante Algoritmos Evolutivos se adapta mejor a las tres primeras variantes presentadas. En esta tesis no se pretende seguir fielmente los conceptos que aparecen en cada uno de ellos ni realizar un estudio comparativo, sino que se ha optado por tomar de cada una de ellas lo que mejor se adapte al problema considerado, siendo por tanto complicado definir a qué variante pertenecen los algoritmos propuestos. Principalmente se han considerado aportaciones de los Algoritmos Genéticos y de las Estrategias Evolutivas, que en los siguientes apartados serán introducidas individualmente.

5.1.2 Introducción a los Algoritmos Genéticos

Formalmente se puede definir un Algoritmo Genético [51] como el conjunto:

$$AG=(P^0, \lambda, l, s, \rho, \Omega, f, t) \tag{5.1}$$

donde

$P^0=(a_1^0,...,a_\lambda^0) \in I^\lambda \quad I=\{0,1\}^l$	Población inicial
$\lambda \in \mathbb{N}$	Tamaño de la población
$l \in \mathbb{N}$	Longitud de cada individuo
$s : I^\lambda \rightarrow I^\lambda$	Operador de Selección
$\rho : I \rightarrow \Omega$	Función de determinación de operadores
$\Omega \subseteq \{ \omega : I \times I \rightarrow \wp \rightarrow I \}$[1]	Conjunto de Operadores Genéticos
$f : I \rightarrow \mathbb{R}$	Función de adaptación
$t : I^\lambda \rightarrow \{0,1\}$	Criterio de terminación

[1] $\wp$ indica que la función se aplica según una cierta función de probabilidad

La forma más usual de representar la estructura de los individuos en los Algoritmos Genéticos es mediante cadenas de longitud fija (l en (5.1)). Estas cadenas codificadas en binario se denominan *genotipos* o *cromosomas*. Cada uno de los bits que forman la cadena se denomina *gen*, mientras que λ representa el número de individuos que componen cada una de las generaciones, y P^0 es la población inicial, que se obtiene de forma aleatoria.

El operador de selección s produce en una generación t, a partir de los individuos de dicha generación $P^t=(a_1^t,a_2^t,....,a_\lambda^t)$ una población intermedia $P^{t'}$ mediante la realización de copias de los individuos de P^t. Estos individuos serán copiados a la población intermedia de acuerdo con la siguiente probabilidad:

$$p_s(a_i^t)=\frac{f(a_i^t)}{\sum_{j=1}^{\lambda} f(a_j^t)} \qquad (5.2)$$

por tanto, probabilisticamente, los individuos con un valor de la función de adaptación mayor aparecerán un mayor número de veces en la población intermedia. Este método de selección se denomina *Selección Proporcional.* Si el mejor (o mejores) individuos de cada generación se copian en la generación siguiente, se dice que la selección es *elitista.*

Una vez que la población intermedia ha sido creada, se procede a la aplicación de los operadores genéticos sobre esta nueva población. Los operadores fundamentales son el cruce y la mutación; sin embargo es posible definir otros nuevos operadores como *cruces múltiples, inversión,* etc. La función ρ determina el operador a aplicar a cada uno de los individuos de la población (en el Algoritmo Genético básico, el operador a aplicar es el mismo para todos ellos, y consiste en un operador compuesto de un cruce seguido de una mutación).

El operador cruce sobre un individuo $a_1 \in P^{t'}$ consta de los siguientes pasos:

a) Seleccionar aleatoriamente un individuo $a_2 \in P^t$
b) Seleccionar aleatoriamente un punto de cruce entre 1 y l-1.
c) Formar dos nuevos individuos (Ver Figura 5.2)
d) Seleccionar aleatoriamente un individuo entre los dos generados.

Este operador se aplica con una probabilidad p_c (es frecuente que sea del orden de 0,6). El cruce es el operador fundamental usado en Algoritmos Genéticos.

Por otro lado la mutación es un operador secundario en los Algoritmos Genéticos. Se aplica con una probabilidad baja p_m (normalmente por debajo de 0.01) y con él se pretende evitar la pérdida de un gen en una determinada posición, que no sería recuperable por medio de cruces, dando lugar a una reducción del espacio de búsqueda. La mutación consiste en cambiar el valor de cada gen del individuo con probabilidad p_m.

La población de la siguiente generación P^{t+1} estará compuesta por los individuos de P'' una vez aplicados los operadores genéticos (Figura 5.4).

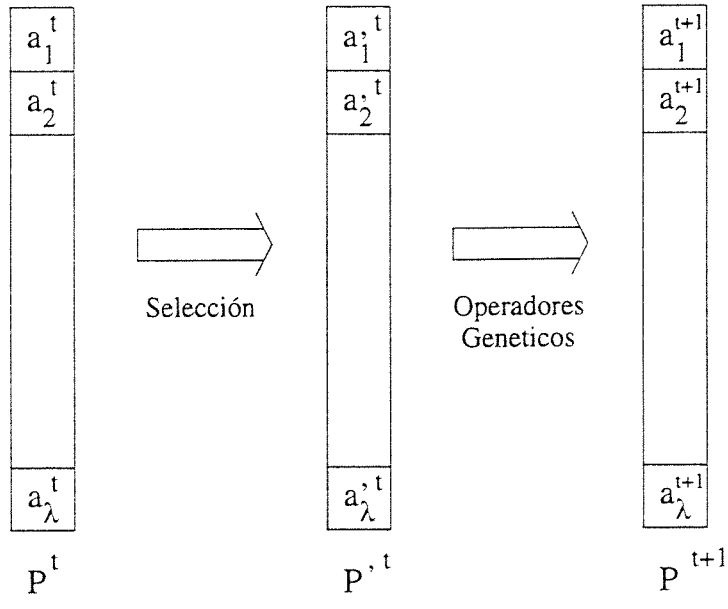

Figura 5.4 Esquema general de los Algoritmos Genéticos.

Numerosos estudios teóricos han demostrado que los Algoritmos Genéticos son un método robusto de optimización, aplicables a un gran número de problemas, incluyendo aquellos con numerosos mínimos locales o funciones no continuas o derivables. De cualquier forma es importante destacar que los Algoritmos Genéticos en general no garantizan el encontrar el óptimo, y si lo encontraran no habría ninguna información que indicara que esto es así. Mediante la introducción del concepto de *esquema* [41], se ha llegado a estimar que aunque los Algoritmos Genéticos procesan λ estructuras en cada generación, en realidad están procesando del orden de λ^3 esquemas (*Paralelismo Implícito*) [41],[124]. Incluso bajo ciertas condiciones [51],[48] son capaces de garantizar el encontrar el óptimo (Teorema de Convergencia Global).

5.1.3 Introducción a las Estrategias Evolutivas

Las Estrategias Evolutivas fueron introducidas por Rechemberg [97]. Los primeros modelos no incluían el concepto de población, sino que un sólo individuo iba evolucionando dando lugar a un sólo individuo mediante mutaciones afectadas por una distribución normal de probabilidad. Este tipo de estrategias se conoce como (1+1)-ES. Posteriormente, Rechemberg incluyó el concepto de población en las *Estrategias Evolutivas con múltiples miembros,* a las que Schwefel denominó (μ+1)-ES, en la que los μ componentes de la población daban lugar a un único descendiente. Los μ individuos de la siguiente generación se seleccionaban entre esos μ+1 individuos (μ padres y un descendiente).

Posteriormente estos algoritmos fueron extendidos por Schwefel para dar lugar a lo que se denominó (μ+λ)-ES y (μ,λ)-ES. En estos casos, los μ componentes de la población daban lugar a λ descendientes, que posteriormente eran reducidos de nuevo a μ individuos para formar la siguiente generación. En el caso de (μ+λ)-ES los componentes de la siguiente generación se forman entre los μ componentes de la generación actual y los λ sucesores creados a partir de ellos. En la variante (μ,λ)-ES, para obtener los μ componentes de una generación sólo se tienen en cuenta los λ individuos creados en la generación anterior.

La descripción formal de (μ+λ)-ES es la siguiente [51],[2]:

$$(\mu+\lambda)\text{-ES}=(P^0,\mu,\lambda,r,m,s,\Delta\sigma,f,g,t) \tag{5.3}$$

donde

$P^0=(a_1^0,\ldots,a_\mu^0) \in I^\mu \; ; \; I=\mathbb{R}^n\times\mathbb{R}^n$	Población inicial
$\mu \in \mathbb{N}$	Número de individuos en la población
$\lambda \in \mathbb{N}$	Número de descendientes
$r : I^\mu \rightarrow I$	Operador de recombinación
$m : I \rightarrow I$	Operador de mutación
$s : I^{\lambda+\mu} \rightarrow I^\mu$	Operador de selección
$\Delta\sigma \in \mathbb{R}$	Paso de control para mutaciones
$f : \mathbb{R}^n \rightarrow \mathbb{R}$	Función objetivo
$g_j : \mathbb{R}^n \rightarrow \mathbb{R}$	Funciones de restricción
$t : I^\mu \rightarrow \{0,1\}$	Criterio de terminación

Con respecto a la estrategia (μ,λ)-ES, ésta consiste en establecer una función de selección distinta, donde $s : I^\lambda \rightarrow I^\mu$, garantizándose además que λ>μ.

La primera característica importante se refiere a la estructura de los individuos, que en este caso están definidos como vectores de números reales. El operador básico en las Estrategias Evolutivas es la mutación. Además también se realiza de una forma distinta a la vista en Algoritmos Genéticos. Cada individuo tiene dos componentes, cada uno de ellos consiste en un vector de reales: el primer componente es la información del propio individuo y el segundo un parámetro interno que determina cómo será la mutación que afecte a cada componente del individuo. El operador de mutación se define del siguiente modo:

$$\begin{aligned} a'=m(a) \quad & a=(x,\sigma)\ ,\ a'=(x',\sigma') \in I \\ & \sigma'=\sigma\cdot\exp N_o(\Delta\sigma) \\ & x'=x+N_o(\sigma') \end{aligned} \tag{5.4}$$

donde $\mathbf{N}_0(m)$ indica un número aleatorio que sigue una distribución normal de media 0 y varianza m. Como se puede comprobar, el parámetro de estrategia σ forma parte del individuo y resulta afectado también por la mutación.

Por otra parte, en las Estrategias Evolutivas, la recombinación o cruce se define de la siguiente manera:

$$\begin{aligned} a'=r(P^t) \quad & a'=(x',\sigma') \in I \\ x_i'=\frac{1}{2}(x_{a,i}+x_{b,i}) \quad & i=1,\ldots,n \quad a,b \in I \\ & \sigma_i'=\frac{1}{2}(\sigma_{a,i}+\sigma_{b,i}) \end{aligned} \tag{5.5}$$

El algoritmo básico parte de una población inicial aleatoriamente seleccionada. Los μ individuos de cada generación producen λ descendientes mediante los operadores de recombinación y mutación dando lugar a una población intermedia formada sólo por los λ descendientes en las estrategias (μ,λ)-ES; y por los λ descendientes a los que se añaden los μ padres en la $(\mu+\lambda)$-ES. Finalmente, el operador de selección reduce la población intermedia a los μ individuos que formarán la siguiente generación, siguiendo una distribución uniforme que afecta a los μ elementos de la población intermedia que tengan un valor más elevado de la función de adaptación, es decir:

$$p_s(a_i')=\begin{cases}1/\mu & , 1\le i\le \mu \\ 0 & , \mu\le i\le x\end{cases} \quad x=\lambda \text{ en } (\mu,\lambda)\text{-ES y } x=\mu+\lambda \text{ en } (\mu+\lambda)\text{-ES} \tag{5.6}$$

Este tipo de selección se denomina *Ordenación Uniforme*. La Figura 5.5 muestra el esquema general de las Estrategias Evolutivas.

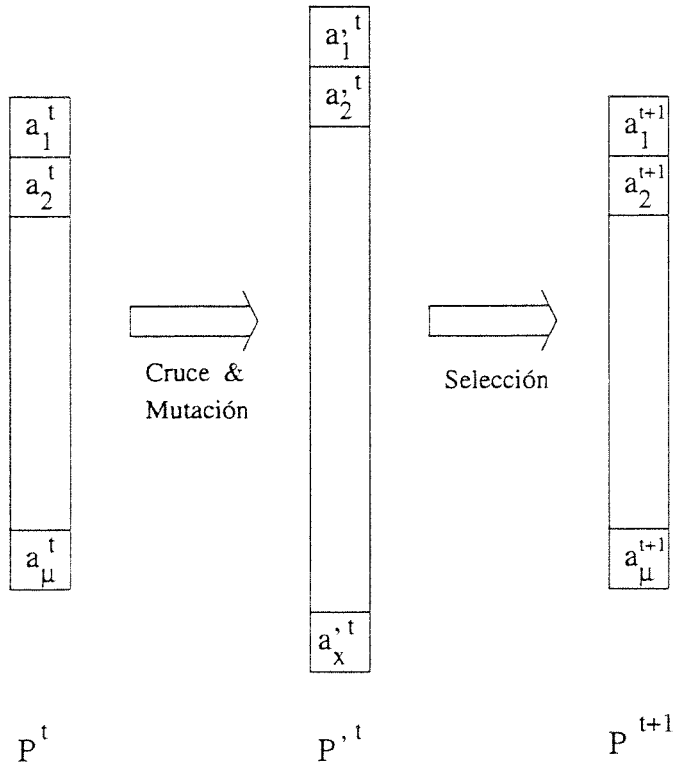

Figura 5.5 Esquema general de las Estrategias de Evolución. $x=\lambda$ para (μ,λ)-ES y $x=\mu+\lambda$ para $(\mu+\lambda)$-ES.

También en el caso de las Estrategias Evolutivas se han descrito desarrollos teóricos sobre la convergencia para las estrategias (1+1)-ES y (μ+1)-ES.

5.1.4 Comparación de los algoritmos

Los Algoritmos Genéticos y las Estrategias Evolutivas tienen numerosos puntos en común. En primer lugar, se tratan de algoritmos de optimización globales basados en la evolución natural, compartiendo la idea de la obtención de una población de individuos a lo largo de sucesivas generaciones que evolucionan de acuerdo a su adaptación, estimada mediante una función de adaptación. También hay una gran similitud en el tipo de operadores genéticos utilizados, principalmente la mutación y la recombinación o cruce.

También existen grandes diferencias entre ellos. En primer lugar, el tipo de estructura utilizado para codificar los individuos, codificación binaria en los Algoritmos Genéticos frente a codificación real en las Estrategias Evolutivas. Esta representación real tiene algunas ventajas como son la posibilidad de usar grandes dominios para las variables, difíciles de conseguir con cadenas binarias de longitud fija, además se facilita la utilización de restricciones en las soluciones. Finalmente, este tipo de representación supone una formulación más natural del problema, lo que facilita la introducción de una heurística propia de cada problema, que se traduce en la utilización de operadores específicos para un determinado problema. En cualquier caso, la codificación real no es exclusiva de las Estrategias Evolutivas, ya que se han realizado numerosos estudios sobre Algoritmos Genéticos con codificación distinta a la binaria, en particular codificación real, definiendo incluso diferentes tipos de cruces y mutaciones [21],[50]. El tipo de problemas en los que más se utiliza esta codificación real es el optimización paramétrica.

Por otra parte, estos algoritmos se diferencian en la forma de utilizar los operadores genéticos, en especial de la mutación. En los Algoritmos Genéticos el operador fundamental es el cruce, utilizando esporádicamente la mutación para recuperar genes perdidos. En las Estrategias Evolutivas la mutación es el operador principal, pudiendo asemejarse a un *método de escalada*. Sin embargo, tal vez la diferencia principal consista en el carácter autoadaptativo de las mutaciones en las Estrategias Evolutivas. Al incluir los parámetros de adaptación σ dentro de la representación del individuo y estar sometido a los operadores genéticos se pretende que el paso de la mutación se adecue a las circunstancias, permitiendo, por ejemplo, mutaciones grandes al principio de la búsqueda, y mutaciones pequeñas en las cercanías del óptimo.

Estudios experimentales comparativos realizados en [51], donde se pretendía encontrar el óptimo en distintas funciones de prueba, llegan a la conclusión de que las Estrategias Evolutivas tienen un gran comportamiento ante búsquedas locales, por lo que consiguen un grado alto de convergencia y una gran robustez ante problemas con pocos mínimos locales. Con los Algoritmos Genéticos, comparativamente se obtienen mejores resultados en funciones muy complejas con numerosos mínimos locales.

En cualquier caso, últimamente se muestra una tendencia a incorporar a cada tipo de algoritmo lo que es beneficioso del otro, por lo que muchas veces es difícil catalogar una determinada implementación dentro de un tipo u otro.

5.2 Obtención de Puntos de Sincronización mediante Algoritmos Evolutivos

En este apartado se estudiará la implementación propuesta para la obtención de una secuencia de puntos de sincronización que definan un movimiento sin colisiones entre varios robots, de forma que el tiempo de ejecución de dicho movimiento sea mínimo [100]. Todo lo desarrollado estará referido al método de los rectángulos, aunque sería igualmente válido para el método de la banda de la trayectoria. A continuación se estudiará la aplicación de los algoritmos evolutivos al problema propuesto. Para ello habrá que considerar los distintos aspectos que definen este tipo de algoritmos para problemas de optimización, que son [114]:

- Estructura cromosómica que forma cada individuo.
- Formas de crear la población inicial.
- Definición de la función de adaptación.
- Operadores Genéticos.
- Restricciones sobre la descendencia.
- Parámetros de control.

Los siguientes apartados definen estos aspectos, aplicados al problema que se intenta resolver en esta tesis, así como distintas modificaciones de cada uno de ellos y los resultados obtenidos en diversas funciones de prueba.

5.2.1 Representación cromosómica de los individuos

Cada individuo deberá representar una sucesión creciente de puntos de sincronización situados sobre el diagrama de coordinación. Teniendo en cuenta que este diagrama es un espacio discreto, el modo más natural de representación de los individuos será mediante vectores de números enteros.

La mayor parte de las problemas resueltos mediante algoritmos evolutivos consideran longitud fija de los cromosomas, es decir, el número de componentes del vector permanece constante a lo largo del algoritmo. Sin embargo, en este caso considerar fijo el número de componentes del vector (número de puntos de sincronización) no parece

adecuado, ya que dicho número afecta significativamente al tiempo total de ejecución del movimiento coordinado, y es por tanto un parámetro a optimizar del problema.

La solución adoptada para representar los individuos ha sido considerar una longitud cromosómica fija, es decir, todas las secuencias tienen el mismo número de puntos de sincronización reales, pero permitiendo que varios puntos de sincronización tengan las mismas coordenadas (denominados *Puntos Múltiples*). Por tanto, la longitud del cromosoma define el número máximo de puntos de sincronización permitidos.

Esta longitud cromosómica es un parámetro importante en el buen funcionamiento del algoritmo. Con un número demasiado bajo de este parámetro, puede ocurrir que se pierdan soluciones. Por otro lado, un valor alto incrementa sustancialmente el espacio de búsqueda. Además con un valor alto es más difícil encontrar soluciones iniciales válidas cuando existen numerosas zonas de colisiones. De cualquier forma, como se comprobará posteriormente, el algoritmo presenta un buen comportamiento para longitudes cromosómicas grandes, debido principalmente a la utilización de operadores genéticos específicos a este problema.

Sea N la longitud del cromosoma y R el número de robots. El punto inicial de la secuencia es el origen de los caminos de cada uno de los robots $\mathbb{C}_0=(1,1,...,1)$, mientras que el último punto es el punto final de cada uno de los caminos, es decir $\mathbb{C}_{max}=(max_1,max_2,...,max_R)$. Entre estos dos puntos, cada individuo estará compuesto por una secuencia de puntos de sincronización $\{P^i\}$ con $1 \le i \le N-1$, $P^i=(x_1^i, x_2^i,..., x_R^i)$ y x_j^i es un intervalo perteneciente al camino del manipulador j. El hecho de que un robot no se mueve hacia atrás a lo largo de su camino supone una restricción a añadir en la configuración del individuo: las coordenadas de puntos de sincronización sucesivos deberán de ser monótonas crecientes, es decir $x_j^i \le x_j^{i+1} \ \forall\ i, j$. Finalmente, todos los puntos deberán estar sobre el diagrama de coordinación, por tanto $x_j^i \ge 0$, $x_j^i \le \max_j \ \forall\ i, j$. Por tanto, formalmente, cada individuo será una secuencia de puntos de sincronización que cumpla:

$$\{ (x_1^i,..., x_R^i) \in \Omega_1 \times ... \times \Omega_R \ / \ \forall\ i,j \ x_j^i \le x_j^{i+1} \} \quad 0 \le i \le N$$
$$\text{con} \quad (x_1^0,...,x_R^0)=\mathbb{C}_0 \quad \text{y} \quad (x_1^N,...,x_R^N)=\mathbb{C}_{max} \tag{5.7}$$

En la Figura 5.6 se representa un individuo, suponiendo que el número máximo de puntos de sincronización es N y que el número de robots es R. Como se puede comprobar, es un vector de números enteros de $(N+1) \times R$ componentes, de forma que aparecen las R componentes del primer punto de sincronización, seguido de las del segundo, hasta llegar a las R componentes del último punto de sincronización.

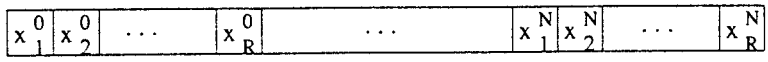

Figura 5.6 Representación de un individuo.

Entre los individuos es posible distinguir dos clases distintas:

- *Individuos Válidos*: Un individuo es *válido* si forma una secuencia creciente de hipercubos libres, y por tanto da lugar a camino de coordinación libre de colisiones. (Figura 5.7).

- *Individuos no válidos*: Un individuo no es válido si existe al menos un hipercubo que no es libre en la secuencia. En este caso es posible que se produzca una colisión entre los robots. Estos individuos no se consideran soluciones del problema. (Figura 5.8).

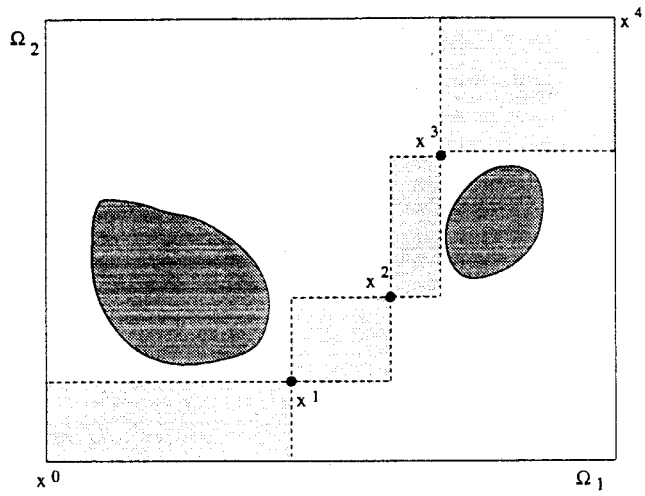

Figura 5.7 Individuo válido

Finalmente, cuando se aplican operadores genéticos, es posible que se obtengan secuencias de puntos de sincronización no crecientes. Estrictamente hablando, no se trata de individuos, pero en esta tesis serán denominados *individuos no aceptables*. (Figura 5.9).

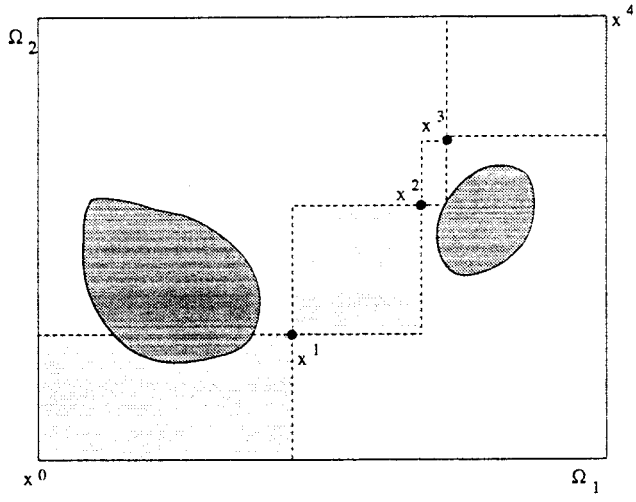

Figura 5.8 Individuo no válido

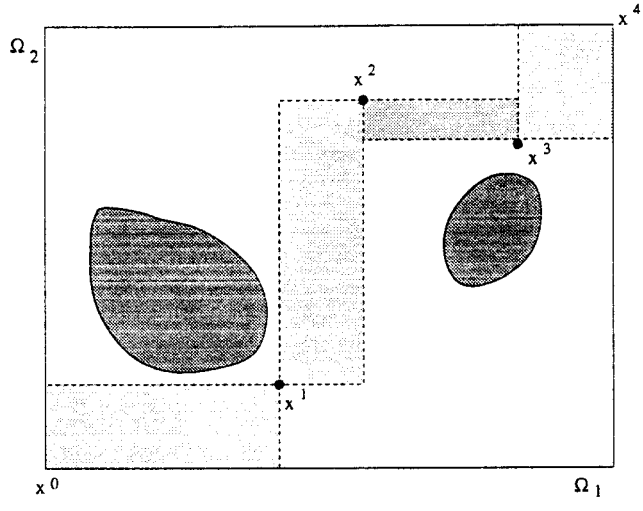

Figura 5.9 Individuo no aceptable. Se observa que el *punto de sincronización* x^3 tiene una componente decreciente respecto a x^2.

5.2.2 Función de adaptación

En este apartado se pretende definir la función de adaptación usada en la implementación de los algoritmos. La función de adaptación deberá tener en cuenta los dos tipos de individuos que pueden aparecer, según se consideró en el apartado anterior: individuos válidos e individuos no válidos.

Para individuos válidos, la función de adaptación deberá coincidir con el valor a optimizar: el tiempo total de ejecución que necesitan los robots para completar sus respectivos caminos cuando los puntos de sincronización se encuentran en las posiciones definidas por las especificaciones del individuo. Evidentemente, para obtener este tiempo será necesario utilizar un modelo que describa los movimientos de los robots, que coincidirá con el utilizado en el capítulo anterior y que se decribe en el Apéndice C.

En la Figura 5.10 se representa un ejemplo de los valores de la función de adaptación para un caso muy simple. Se trata de un diagrama de coordinación para representar el movimiento coordinado de dos robots, en el cual no existe ninguna región de colisión. Las funciones de adaptación representadas suponen un sólo punto de sincronización, de forma que la figura representa el valor de la función de adaptación si éste se encontrara en cada uno de los puntos del diagrama de coordinación.

Nótese el comportamiento de esta función en forma de "valle", en que los dos puntos con un valor máximo en la función de adaptación se corresponden con la ejecución secuencial de los dos robots, uno tras otro. Nótese también que el mínimo se corresponde con el punto de sincronización en el origen de coordenadas, lo cuál es lógico, ya que contempla un caso en que no existe ningún punto de sincronización efectivo. Un mínimo similar aparecerá en la celda correspondiente al final de ambos caminos.

Para los individuos no válidos, la función de adaptación definida es distinta. En este caso no interesa el tiempo de realización del movimiento, debido a que no son solución al problema. Dos posibles alternativas han sido propuestas:

- Una función de adaptación en la que se realice una estimación de lo lejos que se encuentra de un individuo válido. Por supuesto, en cualquier caso la función de adaptación para estos individuos deberá de ser de un orden de magnitud mayor que la utilizada para individuos válidos. La función propuesta es:

$$f(x)=K+nco \tag{5.8}$$

donde K es un valor alto con respecto a los tiempos característicos de realización del movimiento coordinado y nco es el número de celdas de tipo colisión que aparezcan en la secuencia de hipercubos que caractericen el individuo.

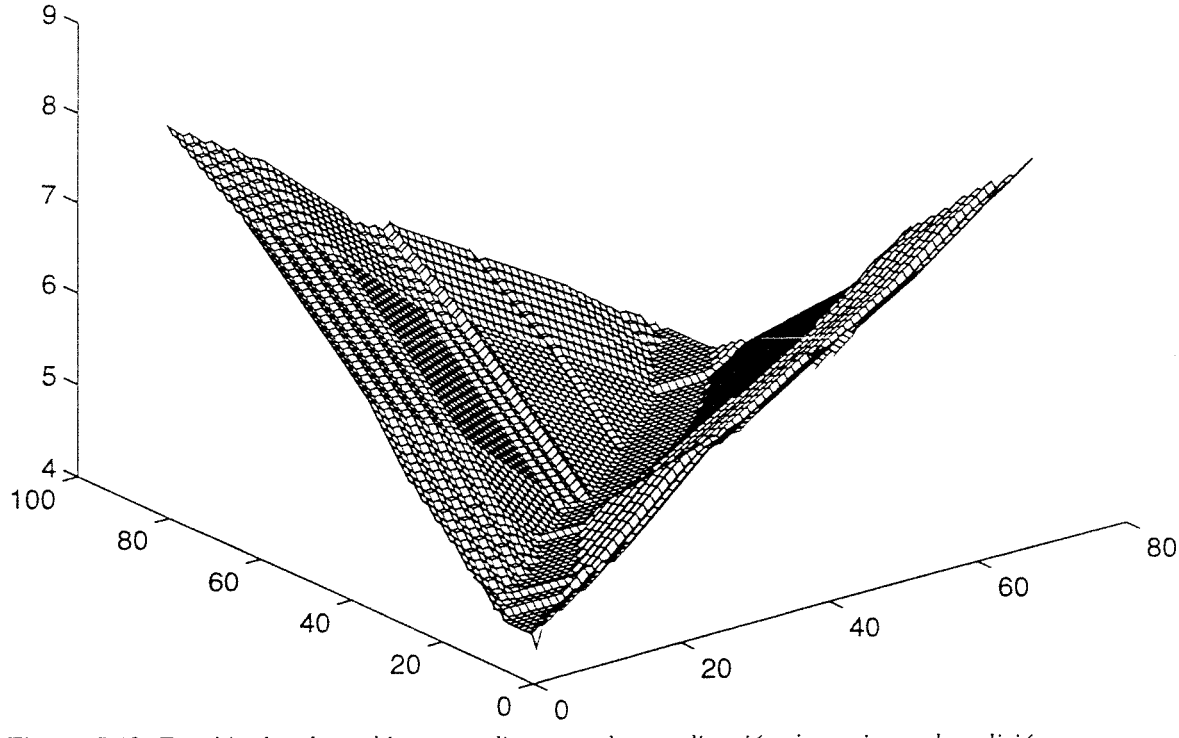

Figura 5.10 Función de adaptación en un *diagrama de coordinación* sin *regiones de colisión*, considerando un único *punto de sincronización*.

Este tipo de individuos, debido a la filosofía los Algoritmos Evolutivos tienden a desaparecer rápidamente de las poblaciones. Sin embargo, esta función de adaptación es útil para diagramas de coordinación con numerosas regiones de colisión, en los que es difícil obtener en la población inicial una serie de individuos válidos de forma totalmente aleatoria. Si todos los individuos de la población inicial son no válidos, esta función permitirá obtener elementos válidos que sean capaces de evolucionar hacia el óptimo. Esta función tiene también el inconveniente del tiempo de cálculo necesario para evaluarla junto.

- La segunda alternativa considera la función de adaptación $f(x)=K$, lo que hace necesario utilizar alguna alternativa para la obtención de poblaciones iniciales en diagramas de configuración complejos. Esta alternativa puede consistir en basarse en alguna solución

conocida, como la estudiada en el Capítulo 3. Sin embargo estas técnicas tienden a reducir la dispersidad de la población.

La solución implementada consiste en utilizar la segunda alternativa, salvo en el caso que en la población inicial el número de individuos válidos sea demasiado pequeño. De esta forma se pretende obviar los inconvenientes de la primera alternativa.

5.2.3 Generación de la población inicial

La generación de la población inicial en los Algoritmos Evolutivos se realiza de forma aleatoria. Sin embargo, en este caso hay que considerar las restricciones asociadas a los individuos, que dan lugar a los distintos tipos de individuos que se comentaron anteriormente.

De cualquier forma, la población inicial que se ha decidido considerar estará formada por individuos válidos y no válidos, por tanto la única restricción que se les impondrá será la de formar una sucesión monótona creciente (no se permiten individuos no aceptables). Así, a menos que resulta muy costoso su obtención, se ha comprobado que se obtienen mejores resultados si al menos una parte de estos individuos son válidos.

5.2.4 Operadores Genéticos

Teniendo en cuenta la configuración característica de los individuos en este problema, será necesario modificar los operadores genéticos utilizados en la literatura para aplicaciones con individuos implementados de forma ordinaria. En este apartado se describen los distintos operadores genéticos que han sido implementados en el algoritmo, incluyendo operadores de cruce, mutación y un operador específico al problema que se está tratando, que se ha denominado *Operador de Reducción.*

5.2.4.1 Operadores de cruce

Se han definido cuatro tipos distintos de cruces, la mayor parte de ellos modificaciones de cruces existentes para el caso de Algoritmos de Genéticos con codificación real. Estos cruces definidos son los siguientes:

Cruce Rectangular

Es una adaptación del cruce denominado *plano* [96],[50]. El cruce de dos individuos $X=\{ (x_1^i, x_2^i,..., x_R^i) / 0 \leq i \leq N \}$ e $Y=\{ (y_1^i, y_2^i,..., y_R^i) / 0 \leq i \leq N \}$ dará lugar a un hijo

$H=\{ (h_1^i, h_2^i,..., h_R^i) / 0 \leq i \leq N \}$ de forma que h_j^i es un valor aleatorio entero según una distribución uniforme en el intervalo $[x_j^i, y_j^i]$. En la Figura 5.11 se muestra un cruce rectangular para un ejemplo de coordinaciónde dos robots. Cada uno de los puntos de sincronización del descendiente deberá estar situado sobre las zonas rectangulares que aparecen en color gris.

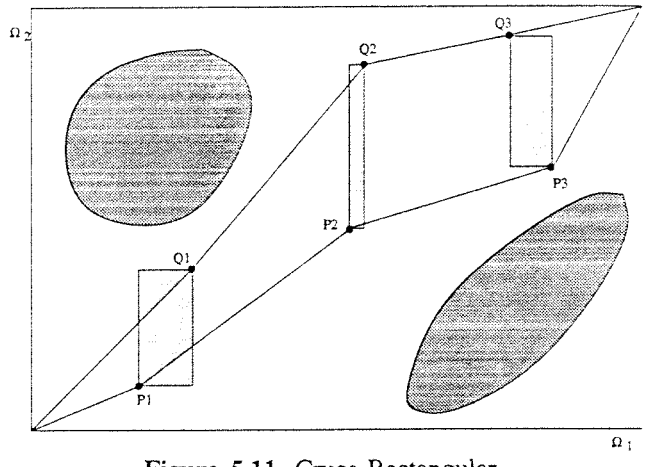

Figura 5.11 Cruce Rectangular

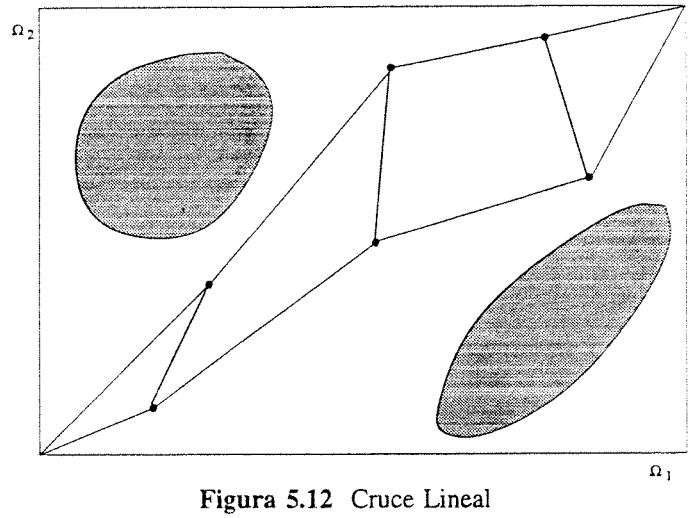

Figura 5.12 Cruce Lineal

Cruce Lineal

El cruce de dos individuos X e Y tal como fueron definidos anteriormente, dará lugar a un descendiente $H=\{ (h_1^i, h_2^i,..., h_R^i) / 0 \le i \le N \}$ tal que $h_{j_i}^i$ será un valor aleatorio en el segmento definido entre los puntos $(x_1^i, x_2^i,..., x_R^i)$ y $(y_1^i, y_2^i,..., y_R^i)$. La Figura 5.12 muestra este tipo de cruce. Cada uno de los puntos de sincronización que definen al descendiente deberá estar sobre las líneas que aparecen con trazo grueso.

Cruce Lineal Extendido

Es una ampliación del cruce anterior. Al utilizar el cruce lineal se están buscando individuos cuyos puntos de sincronización se encuentren entre los correspondientes de los padres. Sin embargo estos individuos no tienen siempre por qué ser mejores. En el *cruce lineal extendido* se pretende ampliar esta búsqueda. En este caso, cada punto de sincronización h_j^i del descendiente H se encontrará sobre el segmento extendido, es decir, considerando también la prolongación, en un porcentaje fijo de su longitud total, del segmento definido entre los puntos $(x_1^i, x_2^i,..., x_R^i)$ y $(y_1^i, y_2^i,..., y_R^i)$. En la Figura 5.13 se muestra este nuevo tipo de cruce.

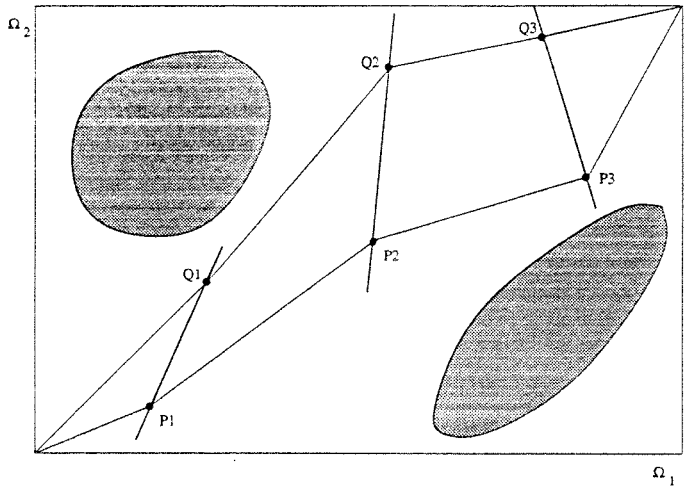

Figura 5.13 Cruce Lineal Extendido

Cruce Simple

Es una extensión del cruce más común en Algoritmos Genéticos. Se selecciona un valor entero aleatorio k tal que $1 \le k \le N-1$, creándose dos nuevos individuos. Uno, H, compuesto

por los primeros k puntos de sincronización del padre X y el resto pertenecientes a Y, mientras que el segundo sucesor G estará formado por los k primeros puntos de sincronización de Y, y el resto de X. De estos dos descendientes, se selecciona el más prometedor. En la Figura 5.14 aparecen representados dos posibles cruces de $\{P_1,P_2,P_3\}$ y $\{Q_1,Q_2,Q_3\}$. Si k=1, uno de los descendientes será $\{Q_1,P_2,P_3\}$, mientras que el otro será $\{P_1,Q_2,Q_3\}$.

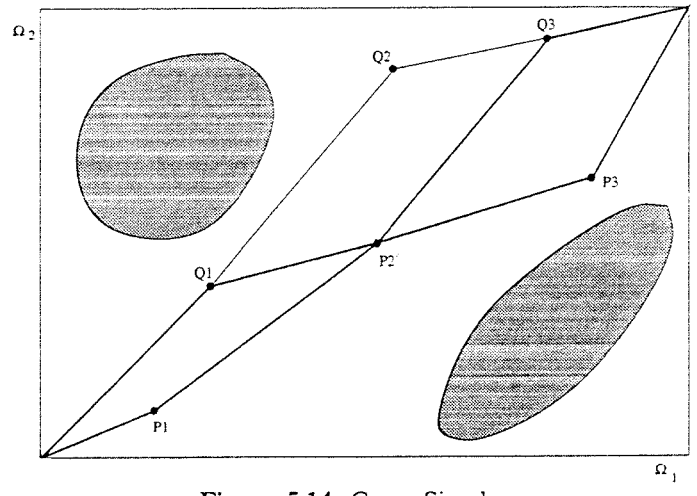

Figura 5.14 Cruce Simple

5.2.4.2 Operadores de Mutación

En general, una mutación produce una modificación en un gen de un individuo. De nuevo, debido a las características particulares de los individuos definidos en esta aplicación, es necesario realizar una serie de modificaciones sobre las mutaciones utilizadas normalmente, realizándose las mutaciones sobre puntos de sincronización. Los siguientes párrafos describen los distintos tipos de mutaciones que se han considerado.

Mutación Proporcional

Dado un individuo $X=\{ (x_1^i, x_2^i,..., x_R^i) / 0\leq i\leq N \}$, se selecciona de forma aleatoria un valor entero k tal que $1\leq k\leq N-1$, que representará el punto de sincronización que será modificado. La forma de realizar la mutación será la siguiente: Cada coordenada x_j^k del punto $(x_1^k,...,x_R^k)$ será sustituida por un valor aleatorio en el intervalo $[x_j^{k-1},x_j^{k+1}]$. En la Figura 5.15 aparece esta mutación. El nuevo valor del *punto de sincronización* deberá estar

sobre el rectángulo gris. La principal ventaja de este tipo de mutación es que nunca genera individuos no aceptables si el individuo de partida no lo era.

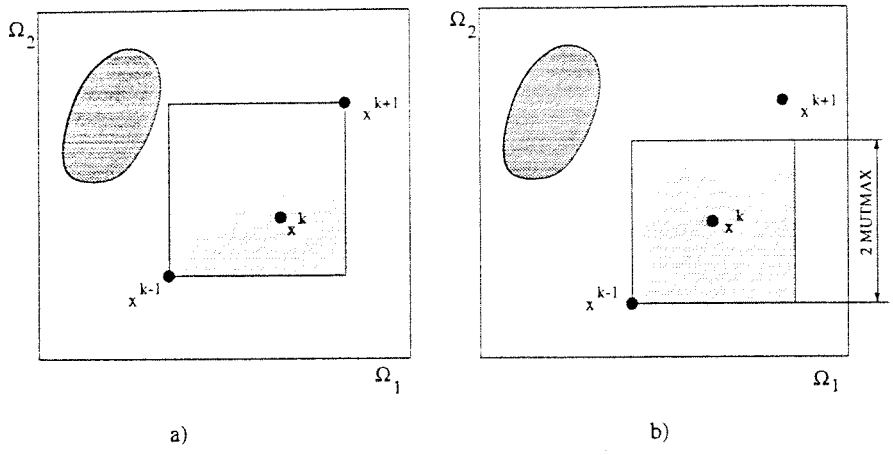

Figura 5.15 Tipos de mutaciones propuestas. a) Mutación proporcional. b) Mutación con límites fijos.

Una posible variación de esta mutación consiste en alterar sólo una de las coordenadas del punto de sincronización seleccionado, escogida de forma aleatoria. Esta variación se ha denominado *Mutación Proporcional Simple.*

Mutación con Límites Fijos

Dado un individuo $X=\{ (x_1^i, x_2^i,..., x_R^i) \ / \ 0 \leq i \leq N \}$, se seleccionan dos valores aleatorios; el primero de ellos, k, determina el punto de sincronización sobre el que se realizará la mutación, mientras que el segundo m determina el valor que tendrá la modificación realizada. Si se define *MUTMAX* como el valor máximo de la mutación realizada, se debe cumplir que $1 \leq m \leq MUTMAX$. Por tanto, la mutación consistirá en sustituir el punto de sincronización $(x_1^k,...,x_R^k)$ por $(x_1^k+m,...,x_R^k+m)$. En este caso también se pueden definir diferentes variaciones:

- *Mutación dependiente de la coordenada*: consiste en determinar diferentes valores de m para cada una de las coordenadas.

- *Mutación Simple:* en este caso, la mutación sólo afecta a una de las coordenadas del punto de sincronización elegido.

- *Mutación no uniforme*: esta tercera variante se obtiene cuando *MUTMAX* varía en función del número de generaciones ejecutadas. La idea de esta modificación consiste en realizar mutaciones grandes al principio de la ejecución del algoritmo, con lo que se consigue aumentar la dispersidad de la población, y en las últimas generaciones, cuando se supone que existen individuos cerca del óptimo, realizar mutaciones pequeñas, con objeto de buscar el óptimo por los alrededores de los puntos existentes. Este tipo de mutación es un primer paso hacia las mutaciones adaptativas utilizadas en las Estrategias Evolutivas.

En estos tipos de mutaciones, al contrario de lo que ocurría con las mutaciones proporcionales, los individuos obtenidos pueden ser no aceptables. En próximos apartados se verán los métodos propuestos para convertir estos individuos en aceptables.

Mutación Multipunto

Las *mutaciones multipuntos* no son nuevos tipos de mutaciones, sino una posible modificación que puede ser aplicable a cualquiera de los tipos anteriores. En este caso, la mutación no afecta solamente al punto de sincronización seleccionado, sino que se extiende a todos aquellos puntos de sincronización con las mismas coordenadas que dicho punto. Con esta modificación se pretende no aumentar el número de puntos de sincronización efectivos cuando se realiza una mutación.

5.2.4.3 Operador de Reducción

Este operador es específico de la aplicación que se está realizando y con él se pretende aprovechar el conocimiento que se tiene del problema para conseguir individuos con una mejor función de adaptación.

Como se describe en el capítulo anterior, un punto de sincronización *B* es redundante respecto a *A* y *C*, si está colocado en la secuencia entre otros dos puntos de sincronización A y *C* que pueden ser consecutivos (Ver Figura 5.16). Esto significa que el punto *B* puede ser eliminado, obteniendo una nueva secuencia con una función de adaptación que será menor que en la anterior (en el caso más desfavorable será igual a la secuencia anterior). En efecto, el tiempo necesario para realizar el movimiento entre los puntos *A* y *C* sin ninguna restricción no puede ser mayor que el tiempo invertido en realizar el movimiento entre *A* y *C* con la restricción de pasar por el punto intermedio *B*.

Con el *operador de reducción* se pretende modificar un individuo con puntos redundantes con objeto de obtener uno mejor. Este operador modifica la posición del punto de sincronización B desde su posición primitiva hasta las posiciones *A* o *C*. Es decir, se

elimina el punto intermedio, para conseguir un individuo con un número menor de puntos de sincronización efectivos.

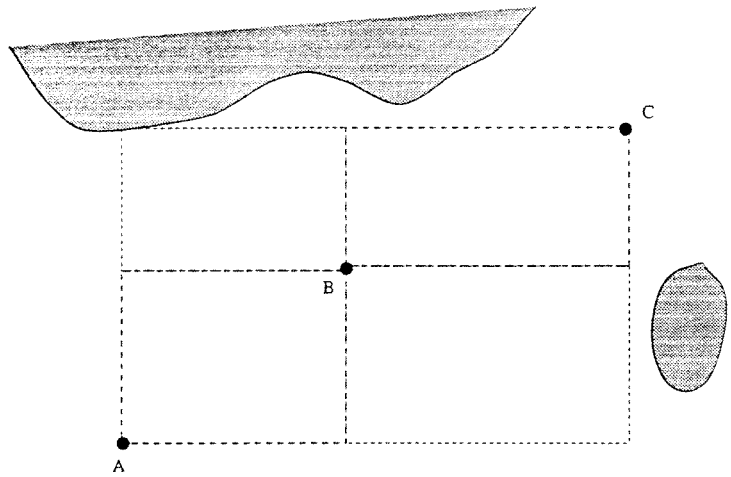

Figura 5.16 Puntos de sincronización redundantes

El uso de este operador específico tiende a reducir el número de puntos de sincronización efectivos en los individuos de la población. De cualquier forma, se ha comprobado un mejor comportamiento del algoritmo cuando este operador se aplica solamente a algunos individuos (seleccionados aleatoriamente), ya que de este forma se evita una disminución excesiva en la dispersidad de la población. En cierto modo, este operador específico se asemeja en su modo de aplicación a una mutación.

5.2.5 Tratamiento de los individuos no aceptables

Cuando se aplican los operadores genéticos vistos en el apartado anterior, los individuos resultantes pueden no verificar las condiciones requeridas para ser individuos aceptables, es decir, dado un individuo y un punto de sincronización de dicho individuo $P^k=(x_1^k,..,x_R^k)$, se cumple que dicho punto no forma una secuencia creciente con P_{k-1}. Estos individuos no deben de formar parte de la siguiente generación, y por tanto es necesario realizar acciones para resolver estos conflictos. En este apartado se especifican las acciones propuestas, que son:

a) *Rechazar el individuo*: El individuo es rechazado y sustituido, simplemente llevando a cabo otras mutaciones o cruces hasta que se obtenga un individuo aceptable.

b) *Mover las coordenadas conflictivas*: Esta solución modifica solamente las coordenadas de P que cumplan la condición anterior, dejando las demás inalteradas, es decir:

$$\forall\, j,k \;/\; x_j^{\,k} > x_j^{\,k+1} \quad \begin{cases} x_j^{\,k+1} = x_j^{\,k} & 1 \le k \le N \\ x_j^{\,N} = x_j^{\,N+1} \end{cases} \tag{5.9}$$

Una alternativa a este método consiste en:

$$\forall\, j,k \;/\; x_j^{\,k} > x_j^{\,k+1} \quad \begin{cases} x_j^{\,k} = x_j^{\,k+1} & 1 \le k \le N \\ x_j^{\,N} = x_j^{\,N+1} \end{cases} \tag{5.10}$$

De este modo no se modifica el punto conflictivo sino el siguiente. La Figura 5.17 muestra las dos variaciones de esta solución. Supóngase que como resultado de un cruce o una mutación se genera la secuencia no aceptable $\{P^1,P^2,P^3\}$. Siguiendo el tipo de solución expresado en (5.9) la secuencia obtenida será $\{P^{1'},P^2,P^3\}$, mientras que se generará $\{P^1,P^{2'},P^3\}$ según (5.10).

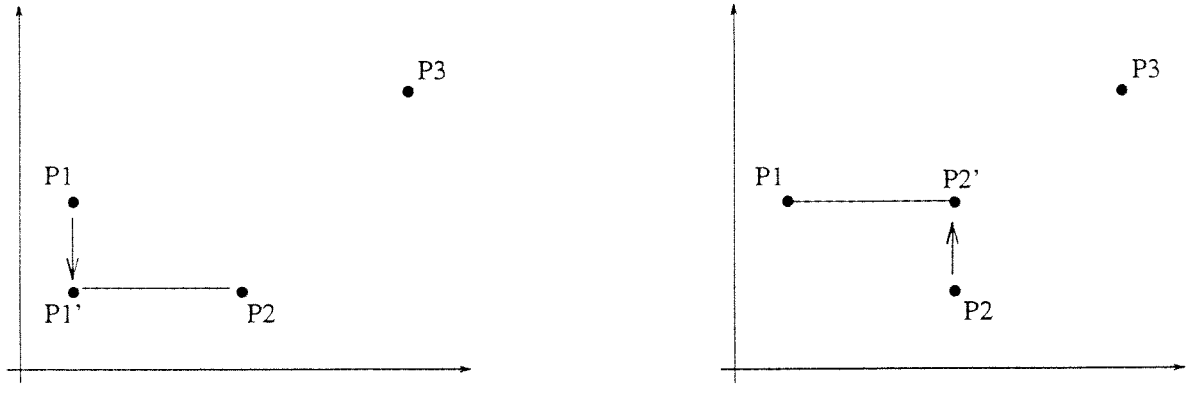

Figura 5.17 Resolución de conflictos modificando coordenadas.

c) *Unir puntos*: Esta última solución consiste en modificar todas las coordenadas del punto de sincronización conflictivo, hasta hacerlo coincidir con el siguiente. De esta forma se consigue formar un punto múltiple. Formalmente:

$$\exists\, j,k \;/\; x_j^k > x_j^{k+1} \quad \begin{cases} P^{k+1}=P^k & 1 \le k \le N \\ P^N=P^{N+1} \end{cases} \tag{5.11}$$

De forma similar se podría resolver el conflicto haciendo $P^k=P^{k+1}$. La ? muestra de forma gráfica el tipo de resolución expresado en (5.11).

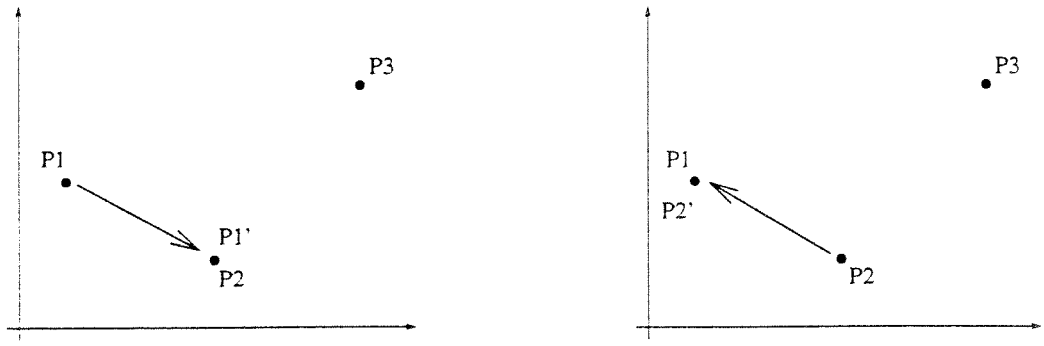

Figura 5.18 Resolución de conflictos uniendo puntos.

5.3 Ejemplos

Un este apartado se presenta la aplicación del algoritmo a varios ejemplos. Se ha pretendido estudiar el comportamiento del algoritmo para varios tipos de diagramas de coordinación con objeto de estudiar el funcionamiento comparado entre los distintos operadores genéticos propuestos.

Los tres casos presentados corresponden al estudio del movimiento coordinado de dos robots. En el primer caso se presenta el caso de un diagrama de coordinación con numerosas zonas de colisiones, y con fuertes restricciones en la solución, en concreto, cualquier solución debe pasar por un estrecho pasillo, como se puede comprobar en la Figura 5.19-a. Este tipo de diagrama de coordinación aparece frecuentemente cuando se

intenta evitar la colisión entre dos robots mediante un movimiento de ida y vuelta en uno de los robots a partir de su camino inicial (Ver Capítulo 3). El segundo de los diagramas de coordinación presenta varias zonas de colisión de pequeño tamaño, repartidas por el diagrama. Finalmente, el último se corresponde con un diagrama de coordinación con una única zona de obstáculos. Posiblemente sea el tipo de diagramas que aparece más frecuentemente en la práctica.

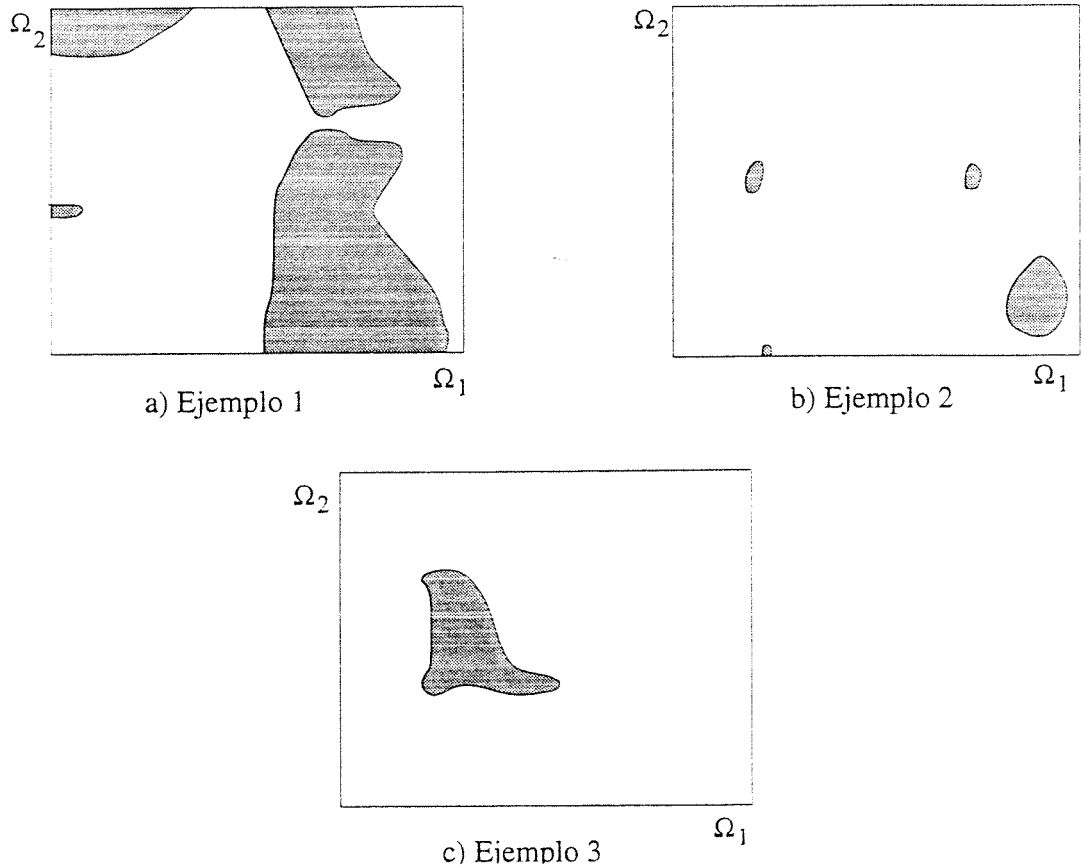

Figura 5.19 Diagramas de coordinación correspondientes a los tres ejemplos tratados.

Se ha realizado un estudio comparativo entre los distintos métodos de selección, y los diferentes tipos de cruces, mutaciones y parámetros genéticos. En general, se ha comprobado que estos algoritmos son bastante robustos frente a estas modificaciones, obteniendo en la mayoría de los casos buenos resultados. Estos resultados se presentan en las tablas que aparecen en los próximos párrafos, en las que se muestra la media, desviación típica y el valor mínimo obtenido para la función de adaptación (un asterisco indica un mínimo global) para las diferentes pruebas realizadas. El número de simulaciones

realizado en cada prueba ha sido de 50, siendo 200 el número de generaciones consideradas para cada simulación.

5.3.1 Tipo de selección

En primer lugar se consideró la eficiencia del algoritmo frente al tipo de selección. Se consideraron tres tipos de selección. El primero de ellos es un método elitista con probabilidad de selección proporcional al valor de la función de adaptación (mecanismo de selección propio de los algoritmos genéticos). Los otros dos se corresponden con métodos propios de Estrategias Evolutivas ($\mu+\lambda$)-ES y (μ,λ-ES). El tamaño de la población para el criterio de selección proporcional ha sido de 25 individuos. En las otras estrategias los valores escogidos han sido $\lambda=5$ y $\mu=25$. Con respecto al valor mínimo encontrado para la función objetivo, se han encontrado buenos comportamientos para la selección proporcional y (μ,λ) (similares, pero ligeramente superior en el caso la selección proporcional. Algo peor es el comportamiento para la selección ($\mu+\lambda$), comprobándose una relativamente rápida reducción de la dispersidad de la población, quedando atrapada en numerosas ocasiones en mínimos locales. La Tabla 5.1 muestra estos resultados.

Tabla 5.1 Valores de la función de adaptación (en segundos) para distintos tipos de selección

Ejem.	Tipo Sel.	Media	Desv. Típ.	Mínimo
1	Proporc.	5.4664	0.020	5.4060
	(μ,λ)	5.4693	0.0278	5.3930*
	($\mu+\lambda$)	5.5165	0.075	5.4060
2	Proporc.	4.7457	0.0146	4.7422
	(μ,λ)	4.8662	0.2479	4.7421*
	($\mu+\lambda$)	4.9094	0.1889	4.7421*
3	Proporc.	3.4810	0.0762	3.4651*
	(μ,λ)	3.4815	0.0771	3.4651*
	($\mu+\lambda$)	3.6857	0.1024	3.4651*

Si bien estos resultados hacen referencia a los obtenidos para la función de adaptación después de 200 generaciones, importantes conclusiones pueden obtenerse en cuanto a la velocidad de convergencia. Para tener una referencia de ésta, se ha considerado el número de generaciones necesarias para alcanzar un valor que se diferencie del mínimo global (conocido por el método de resolución del problema propuesto en el Capítulo 4) en menos de una décima de segundo por un lado, y una centésima de segundo por otro. La Tabla 5.2 muestra estos resultados. En esta tabla no aparecen reflejados los resultados de la selección ($\mu+\lambda$) porque una precisión de una décima sólo se alcanza en el 25% de las simulaciones. En la tabla no se especifica ningún valor (en la tabla aparece el símbolo -) cuando después de las 200 iteraciones no se alcanzado el nivel de precisión deseado.

Tabla 5.2 Generación en la que se alcanza una precisión de una décima y una centésima de segundo

Ejem.	Tipo Sel.	Media		Mínimo		Máximo	
		dec.	cen.	dec.	cen.	dec.	cen.
1	Proporc.	88	-	3	-	180	-
	(μ,λ)	50	-	7	-	-	-
2	Proporc.	22	62	6	10	38	190
	(μ,λ)	14	35	5	9	-	-
3	Proporc.	23	55	3	9	-	-
	(μ,λ)	11	18	3	7	-	-

En general se ha obtenido una velocidad de convergencia mayor cuando se utiliza una selección (μ,λ). Sin embargo este método de selección también ha quedado atrapado en más ocasiones en mínimos locales, sin haber conseguido llegar a la precisión exigida. En concreto para el ejemplo 2, la selección proporcional llegó a una precisión de una décima en todas las simulaciones y solamente en una no llegó a una precisión de centésimas. Sin embargo, con la selección (μ,λ) no se alcanzó una precisión de décimas en el 15% de las simulaciones y el de centésimas en un 18% de las simulaciones. Diferencias menos significativas aparecieron en los otros dos ejemplos.

En resumen se ha comprobado un resultado similar entre las estrategias proporcional y (μ,λ), presentando esta última una mayor velocidad en la convergencia pero una menor fiabilidad en la obtención de valores próximos al mínimo. En el resto de las pruebas que se realizan en este capítulo se ha considerado selección proporcional.

5.3.2 Mutaciones

En este apartado se presentan los resultados obtenidos al modificar los distintos parámetros que afectan a las mutaciones. En concreto se han contemplado la modificación de la probabilidad de mutación y los distintos tipos de mutaciones definidos en el apartado 5.2.4.2 de este capítulo.

La Tabla 5.3 muestra los resultados para distinta probabilidad de mutación. No se han encontrado diferencias significativas en este punto. Sin embargo, los mejores resultados se obtienen para probabilidad de mutación alta, del orden del 50%. Por tanto en esta implementación el operador de mutación juega un papel de más importancia que el secundario que tiene en Algoritmos Genéticos. Esto se debe a la forma en que han sido implementados estos operadores, de forma que su efecto no se reduce a obtener a partir de un individuo otro totalmente distinto para mantener la diversidad de la población, sino que son capaces de hacer una búsqueda en las cercanías de un individuo, de forma similar a como se realiza en las estrategias evolutivas.

Tabla 5.3 Valores de la función de adaptación (en segundos) para distintas probabilidades de mutación.

Ejem.	%	Media	Desv.Típ.	Mínimo
1	5	5.5207	0.0694	5.4607
	20	5.4689	0.0149	5.4060
	50	5.4537	0.0317	5.4060
	80	5.4605	0.0273	5.4060
2	5	4.8420	0.1892	4.7422
	20	4.7661	0.0322	4.7422
	50	4.7457	0.0147	4.7421*
	80	4.7459	0.0159	4.7421*

También se han realizado experiencias para determinar la eficiencia de los distintos tipos de mutaciones definidos con anterioridad. La Tabla 5.4 muestra los resultados obtenidos. Los tipos de mutaciones que han sido probados son las mutaciones con límites fijos no uniformes (lfnu en la tabla), dependiente de la coordenada (lfdc) y simple (lfs); y mutaciones proporcionales con modificación de una (en la tabla pr1) y dos coordenadas (pr2).

Tabla 5.4 Valores de la función de adaptación (en segundos) para distintos tipos de mutación.

Ejem.	Tipo	Media	Desv. Típ.	Mínimo
1	pfnu	5.4551	0.0309	5.3930*
	pfs	5.4519	0.0312	5.4060
	pfdc	5.4660	0.0207	5.4060
	pr1	5.4741	0.0220	5.4060
	pr2	5.4782	0.0123	5.4735
2	pfnu	4.7422	0.0001	4.7421*
	pfs	4.7422	0.0001	4.7421*
	pfdc	4.7458	0.0151	4.7421*
	pr1	4.7618	0.0308	4.7422
	pr2	4.7531	0.0246	4.7421*
3	pfnu	3.4651	0.0000	3.4651*
	pfs	3.4651	0.0000	3.4651*
	pfdc	3.4651	0.0000	3.4651*
	pr1	3.5538	0.0787	3.4651*
	pr2	3.5815	0.0974	3.4651*

Como conclusiones, de nuevo se ha comprobado un comportamiento robusto del algoritmo frente a este parámetro. De cualquier forma, se ha obtenido un mejor comportamiento de las mutaciones con límites fijos frente a las mutaciones proporcionales. Dentro de las mutaciones con límites fijos, se obtienen mejores resultados con las mutaciones simples, es decir cuando sólo se modifica una coordenada en del punto de sincronización, y con las mutaciones no uniformes, haciendo que el paso de la mutación disminuya a medida que aumenta el número de generación.

5.3.3 Cruces

De nuevo las experiencias realizadas han mostrado un comportamiento bastante robusto ante los distintos tipos de cruces implementados y que fueron explicados con anterioridad. En la Tabla 5.5 se muestran los resultados obtenidos para los cruces simple, rectangular, lineal y lineal extendido.

Tabla 5.5 Valores de la función de adaptación para distintos tipos de cruce.

Ejem.	Tipo	Media	Desv.Típ.	Mínimo
1	simple	5.4604	0.0359	5.4060
	rectang.	5.4961	0.0246	5.4681
	lineal	5.5068	0.0415	5.4060
	lin. ext.	5.4893	0.0258	5.4735
2	simple	4.7458	0.0150	4.7421*
	rectang.	4.7637	0.0314	4.7423
	lineal	4.7722	0.0320	4.7422
	lin. ext.	4.7470	0.0155	4.7423
3	simple	3.4651	0.0000	3.4651*
	rectang.	3.5152	0.0805	3.4651*
	lineal	3.4677	0.0110	3.4651*
	lin. ext.	3.4730	0.0231	3.4651*

De cualquier forma, las experiencias demuestran un mejor comportamiento en todos los casos probados con el cruce simple, obteniéndose también en general buenos resultados con el cruce lineal extendido.

5.3.4 Operador de reducción

Un factor para el que se han observado importantes diferencias en los resultados ha sido el uso del operador específico de reducción. La Tabla 5.6 muestra los resultados comparativos cuando en las mismas circunstancias se utiliza y cuando no se utiliza dicho operador, al utilizar individuos compuestos por 8 puntos de sincronización.

Tabla 5.6 Valores de la función de adaptación (en segundos) en función de la utilización del operador de reducción, cuando la longitud cromosómica es 8.

Ejem.	Reduce.	Media	Desv. Típ.	Mínimo
1	si	4.7538	0.0270	4.7421
	no	4.8901	0.0805	4.7424
2	si	5.4636	0.0240	5.4060
	no	5.6615	0.3034	5.4605
3	si	3.4651	0.0000	3.4651
	no	3.7191	0.1103	3.4651

Además, esta diferencia se acrecienta cuando aumenta el número de puntos de sincronización que forman el individuo. En la Tabla 5.7 aparecen los resultados de una prueba similar a la anterior pero considerando individuos formados por 20 puntos de sincronización. Como se puede observar, los resultados son prácticamente iguales a los correspondientes a la tabla anterior cuando se utiliza el operador de reducción, pero se produce una considerable pérdida en la calidad de la solución cuando dicho operador no se utiliza.

Tabla 5.7 Valores de la función de adaptación (en segundos) en relación al uso del operador de reducción, cuando la longitud cromosómica es 20.

Ejem.	Reduce.	Media	Desv. Típ.	Mínimo
1	si	4.7694	0.0330	4.7421
	no	5.2117	0.1450	4.9450
2	si	5.4666	0.0562	5.4930
	no	5.8551	0.1415	5.5955
3	si	3.4651	0.0000	3.4651
	no	3.9677	0.1490	3.7353

5.3.5 Otros parámetros

En este apartado se estudia la incidencia de otros parámetros propios del algoritmo evolutivo, en concreto el tamaño del cromosoma y el porcentaje de individuos válidos en la población inicial.

5.3.5.1 Tamaño del cromosoma

La longitud del cromosoma, es decir, el número máximo de puntos de sincronización que forman el individuo, es un parámetro importante, ya que a priori no se conoce el número de puntos de sincronización que tendrá la solución óptima. Debido a esto, la efectividad del algoritmo pasará necesariamente por un comportamiento robusto ante este parámetro.

Como se comprobó en el apartado anterior, el buen comportamiento del algoritmo ante este parámetro está ligado al uso del operador de reducción. La Tabla 5.8 muestra los resultados obtenidos cuando se utiliza el operador de reducción al considerar un tamaño del cromosoma de 8 y 20 puntos de sincronización.

Tabla 5.8 Valores de la función de adaptación (en segundos) con relación a la longitud cromosómica.

Ejem.	Tamaño	Media	Desv. Típ.	Mínimo
1	8	4.7538	0.0270	4.7421
	20	4.7694	0.0330	4.7421
2	8	5.4636	0.0240	5.4060
	20	5.4666	0.0562	5.3930
3	8	3.4651	0.0000	3.4651
	20	3.4651	0.0000	3.4651

5.3.5.2 Porcentaje de soluciones válidas en la población inicial

Este parámetro puede afectar especialmente a aquellos diagramas de coordinación con regiones de colisiones que restrinjan de modo significativo las soluciones válidas. Entre los ejemplos propuestos, el ejemplo 1 cumple esta condición.

En general se han obtenido resultados muy similares salvo en los casos en que el número de soluciones iniciales válidas era muy bajo. La Tabla 5.9 muestra los resultados obtenidos variando el porcentaje de soluciones iniciales válidas respecto al total.

Como se observa, los peores resultados se obtienen cuando no se exige que ninguna solución inicial sea válida. Además en este caso se da la circunstancia de que en el 10% de las simulaciones realizadas no ha sido posible obtener ninguna solución valida. Este hecho no se ha tenido en cuenta en los valores que aparecen representados en la tabla.

Otro hecho importante a destacar es el tiempo de cálculo necesario para conseguir de forma totalmente aleatoria una solución válida. En las pruebas realizadas para el ejemplo 1, con poblaciones de 25 individuos, se necesitan aproximadamente 8 segundos de C.P.U. para encontrar un 25% de soluciones válidas, unos 16 segundos para un 50% y unos 32 segundos para que todas las soluciones sean válidas. Vistos estos resultados se ha optado por utilizar un porcentaje de individuos válidos del 25%.

Tabla 5.9 Valores de la función de adaptación (en segundos) en función del porcentaje de individuos válidos en la población inicial.

Ejem.	%	Media	Desv. Típ.	Mínimo
1	0	5.5637	0.2913	5.4607
	25	5.4674	0.0239	5.3939
	50	5.4666	0.0284	5.4060
	100	5.4604	0.0275	5.4060

También, en casos especialmente desfavorables, se puede presentar el caso de ser demasiado costoso el encontrar estas soluciones iniciales, sobre todo cuando la longitud cromosómica es alta. Para estos casos se ha utilizado una búsqueda de soluciones iniciales no aleatoria, sino basada en algún camino coordinado conocido, obtenido por ejemplo de la forma que se expone en el capítulo anterior.

5.3.6 Tiempo de C.P.U.

En este apartado se indican los tiempos de C.P.U. necesarios para ejecutar el algoritmo. El computador utilizado ha sido una estación de trabajo I.B.M. RISC-6000 320H con sistema operativo AIX 3.2.

El tiempo de C.P.U. depende principalmente de los siguientes factores:

- Porcentaje de individuos válidos en la población inicial.
- Longitud cromosómica.
- Número de individuos por generación.
- Número de generaciones.

El primero de los puntos se trató en el apartado anterior, siendo totalmente dependiente del problema considerado. Por su parte, el número de generaciones es dependiente del criterio de terminación seleccionado, por lo que se ha estudiado el tiempo de C.P.U. por generación dependiendo de la longitud cromosómica.

Los resultados obtenidos muestran que el tiempo de C.P.U. por generación es prácticamente independiente del problema (evidentemente para el mismo sistema multi-robot). Así, para un tamaño de cromosomas de 8 puntos de sincronización, se ha obtenido una media de 0,15 segundos por generación, mientras que si la población aumenta a 20 puntos de sincronización el tiempo medio de C.P.U. es de 0,21 segundos.

Teniendo en cuenta los resultados de la Tabla 5.2 donde se muestra que en esas condiciones, en 100 generaciones se alcanza un valor con una precisión por debajo de la décima de segundo, se puede dar como tiempo típico de realización del algoritmo incluyendo la generación de poblaciones iniciales unos 20 segundos para cromosomas compuestos por 8 puntos de sincronización, y 30 segundos si son de longitud 20.

Capítulo 6

Aplicaciones

En este capítulo se muestran los resultados obtenidos en aplicaciones prácticas de los algoritmos propuestos en esta tesis sobre sistemas formados por robots reales. En concreto, se han utilizado robots PUMA 500 de STAUBLI UNIMATION LTD. y robots SCORBOT V, comercializados por la compañía ESHED ROBOTEC Ltd. Los programas generados por los algoritmos que permiten realizar el movimiento coordinado están realizados para los robots PUMA en el lenguaje de programación VAL II, mientras que los programas de los robots SCORBOT se generan en el lenguaje de programación ACL (Advanced Control Language). Para la aplicación de los algoritmos es necesario la obtención de un modelo dinámico para los robots.

6.1 Modelo dinámico del robot PUMA 500

El robot PUMA 500 es un manipulador industrial con seis grados de libertad, todos ellos de revolución. Tres de los ejes permiten mover el brazo, mientras que los tres restantes permiten el movimiento de la muñeca.

El PUMA 500 puede ser programado tanto en el espacio cartesiano como en el de las articulaciones, utilizando para ello el lenguaje de programación VAL II. Para las experiencias realizadas en esta tesis se ha considerado que las trayectorias se especifican mediente un vector de puntos en el espacio de las articulaciones. Además se considera que se opera en *modo continuo,* es decir, el robot no detiene su movimiento en cada punto, sino que realiza una transición suave entre cada dos tramos definidos por dichos puntos. Como se ha descrito a lo largo de esta tesis, los algoritmos propuestos utilizan un modelo de las trayectorias generadas por el controlador, para lo cual se va a utilizar el modelo descrito en el apéndice C. En este modelo, la velocidad de cada una de las articulaciones se supone constante, salvo durante los períodos transitorios de aceleración o deceleración entre tramos,

en que se considera que la posición de cada articulación frente al tiempo viene determinada por un polinomio de cuarto grado.

Como se describe en el apéndice, la trayectoria queda especificada en función de dos parámetros que es necesario identificar:

- La velocidad constante del tramo.
- La máxima aceleración alcanzada durante el período transitorio, que según el modelo propuesto, se considera que se alcanza en el punto medio de dicho transitorio.

Estos parámetros han sido obtenidos de forma experimental por Tondu *et al.* [119]. Los resultados que se obtienen para un valor del parámetro *monitor speed*=100 son los que aparecen en la Tabla 6.1.

Tabla 6.1 Valores de la velocidad constante del tramo y aceleración máxima en las transiciones para *monitor speed*=100

	Art. 1	Art. 2	Art. 3	Art. 4	Art. 5	Art. 6
Vel. (°/s)	81	53	118	220	118	420
A. máx.(°/s^2)	900	600	4000	5000	4000	8500

6.2 Modelo dinámico del robot SCORBOT V

El robot educativo SCORBOT V es un brazo articulado con cinco grados de libertad: giro de la base, del hombro, del codo, elevación y giro (Ver Figura 6.1). Todas las articulaciones están accionadas de forma indirecta por motores de corriente continua, mediante correas. Cada uno de los motores dispone de un *encoder* incremental para su posicionamiento. La Tabla 6.2 proporciona los límites de las distintas articulaciones.

El controlador del SCORBOT V dispone de ocho entradas digitales y ocho salidas digitales, que se utilizarán para la sincronización de los robots.

La programación del robot se ha realizado en ACL, que proporciona un lenguaje multitarea de programación y un entorno de programación al que se accede desde un PC

a través del software ATS (Advanced Terminal Software). ACL permite trabajar en coordenadas articulares, aunque no trabaja directamente con ángulos, sino con *unidades de codificador*. Estas unidades, proporcionales a los valores angulares, se leen directamente de los encoders.

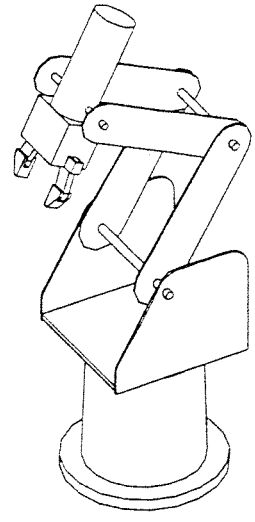

Figura 6.1 Robot SCORBOT V

Tabla 6.2 Valores límites de las articulaciones del SCORBOT en grados y en unidades de codificador, respecto a la posición a la Figura 6.1 (HOME).

Articulación	Max. (grados)	Min. (grados)	Max.(u.codif.)	Min. (u.codif.)
1	170	-132	7246	-5600
2	9	-150	-282	5110
3	122	113	4156	-3862
4	218	-31	3654	-528
5	Ilim.	Ilim.	Ilim.	Ilim.

A pesar de algunas limitaciones que se comentarán a lo largo del capítulo, debido al hecho de estar diseñado con fines educativos, se trata de un manipulador que presenta la mayor parte de las prestaciones presentes en los robots industriales.

Debido a la poca flexibilidad en la programación que presenta el *modo continuo*, no se utilizará en las aplicaciones. Por tanto, el camino vendrá dado por un conjunto de tramos rectilíneos en el espacio de las articulaciones y el movimiento se realizará de forma que el robot se detendrá entre cada dos tramos. Es importante remarcar que estas características del movimiento no vienen impuestas por los algoritmos propuestos en esta tesis, sino que se han adoptado de acuerdo a las capacidades del robot utilizado.

El modelo utilizado para el generador de trayectorias, al igual que para el robot PUMA es el desarrollado an el apéndice C, que, como se ha comentado anteriormente, depende de dos parámetros: la velocidad constante en cada tramo y la aceleración máxima durante las transiciones. En este caso, al no disponerse de dichos parámetros, para su obtención se ha procedido a estudiar de forma experimental las trayectorias realmente generadas en los SCORBOT, mediante la lectura de los encoders.

El parámetro más fácil de obtener es la velocidad, que es dependiente del valor asignado mediante el comando SPEED del lenguaje de programación, al que se le asignan valores entre 1 (velocidad mínima) y 100 (velocidad máxima). Sin embargo, para un mismo valor SPEED se han comprobado pequeñas oscilaciones en función de la posición en que se encuentren las articulaciones. Para la obtención de un único valor, se ha considerado un valor medio de los obtenidos para diferentes configuraciones. La ? muestra los resultados obtenidos en grados por segundo, para distintos valores de la variable SPEED.

Tabla 6.3 Velocidad durante el período de velocidad constante, en grados/s para distintos valores de la variable SPEED.

Articulación	SPEED 20	SPEED 50	SPEED 80
1	9.39	27.20	39.93
2	11.74	35.25	46.98
3	14.68	35.25	51.38
4	29.88	68.73	104.60

Para el cálculo del valor máximo de la aceleración durante el período de transición se han obtenido, de forma experimental, las curvas que representan la posición y la velocidad de las articulaciones frente al tiempo durante los períodos de aceleración o deceleración. La Figura 6.3 muestra las curvas de la posición de las articulaciones, mientras que la Figura 6.4 lo hace con las velocidades, en ambos casos para distintos valores de la variable SPEED.

En las curvas que representan la velocidad se observa que la aceleración media con la que se realiza el proceso de aceleración es dependiente de la variable SPEED, de forma que la aceleración media es mayor a medida que crece el valor de dicha variable. Por tanto, a efectos del modelo, el segundo parámetro, la aceleración máxima, será también dependiente del valor de SPEED.

La obtención de la aceleración máxima se ha realizado de forma empírica, ajustando el polinomio de cuarto grado a las curvas experimentales obtenidas, de forma que la discrepancia en la posición de cada articulación sea lo menor posible.

La Tabla 6.4 muestra los resultados obtenidos para el parámetro aceleración máxima para SPEED=50 (utilizado en la mayor parte de los ejemplos propuestos). Asimismo, la Figura 6.5 muestra de forma gráfica la discrepancia entre el modelo y los valores reales en relación a la posición de las articulaciones, mientras que en la Figura 6.6 se muestra la discrepancia en las velocidades.

Tabla 6.4 Aceleraciones máximas (en grados/s^2) usadas en el modelo para SPEED=50

Articulación	A_{max} (grados/s^2)
1	144.69
2	240.27
3	330.37
4	542.67

6.3 Sincronización entre los robots

Las comunicaciones entre los robots se realizan de forma muy simple, mediante la utilización de las entradas y salidas digitales existentes en los controladores de los robots. Como se ha comentado en capítulos anteriores, la comunicación entre los robots es necesaria en los puntos de sincronización, de forma que al llegar un robot al lugar establecido deberá detener su movimiento y comunicar al otro robot que se encuentra en dicha posición. Por tanto, esta sincronización solo requiere una señal digital por cada uno de los robots, que estará en un estado si el robot se encuentra en el coordenadas del punto de sincronización y en el estado opuesto en caso contrario. La señal asociada a cada robot debe hacerse llegar a los demás. Por tanto, en la sincronización de dos robots el cableado necesario se limita a conectar una salida digital de cada uno de los robots con una entrada digital del otro robot.

Además de la comunicación física es necesario establecer un protocolo de comunicaciones en los puntos de sincronización. A título de ejemplo en lenguaje ACL, las instrucciones necesarias se muestran en los programas de la Figura 6.2, que se ejecutan en cada uno de los robots en cada punto de sincronización. En ellos, se supone que en ambos casos las salidas utilizadas se corresponden con las identificadas con el número 2, mientras que las entradas son la número 7.

```
SINCRONIZACIÓN ROBOT 1          SINCRONIZACIÓN ROBOT 2

     set out[2]=1                    wait in[7]=1
     wait in[7]=1                    set out[2]=1
     set out[2]=0                    wait in[7]=0
     wait in[7]=0                    set out[2]=0
```

Figura 6.2 Rutinas para la sincronización de los robots en lenguaje ACL.

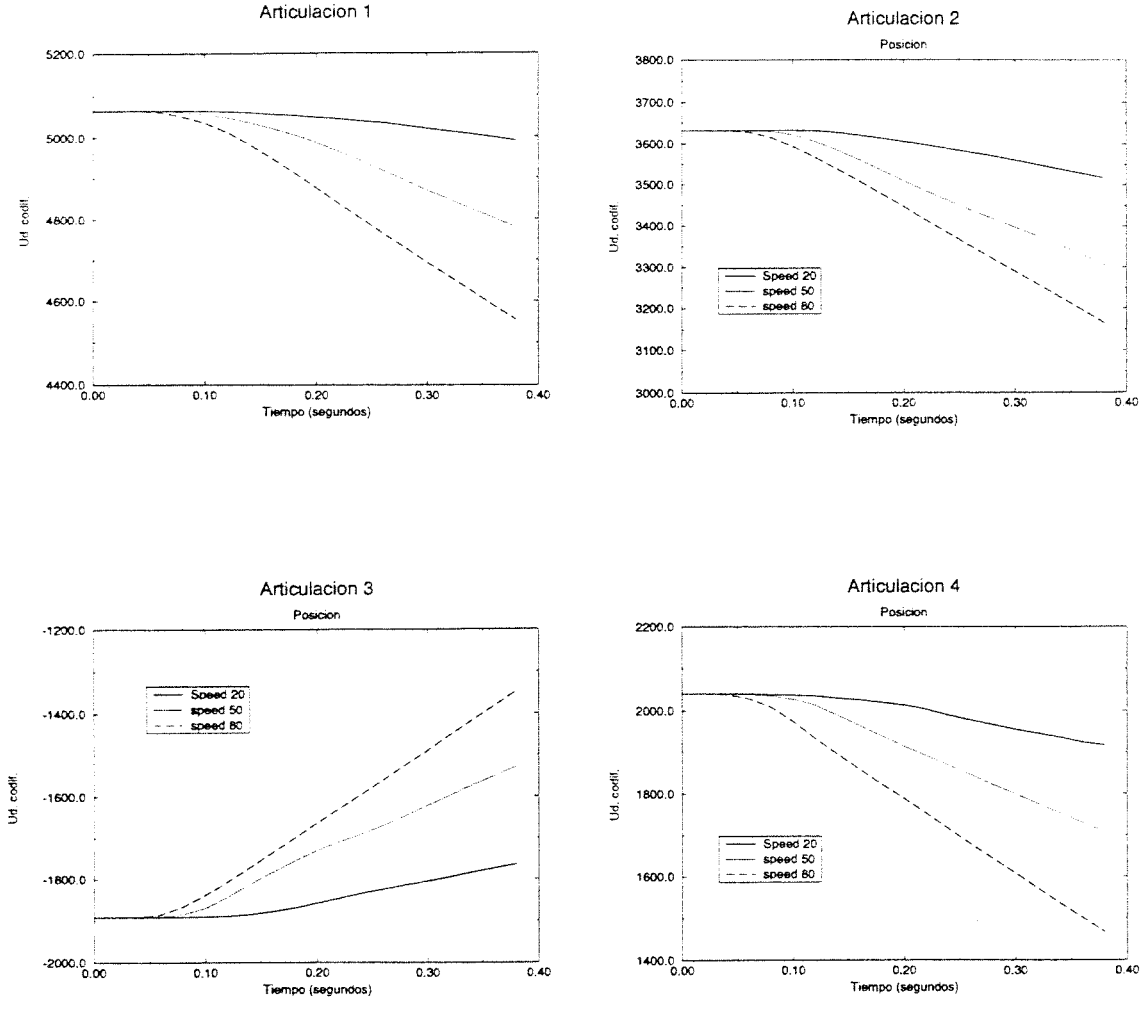

Figura 6.3 Posición de las articulaciones durante el transitorio para distintos valores de SPEED

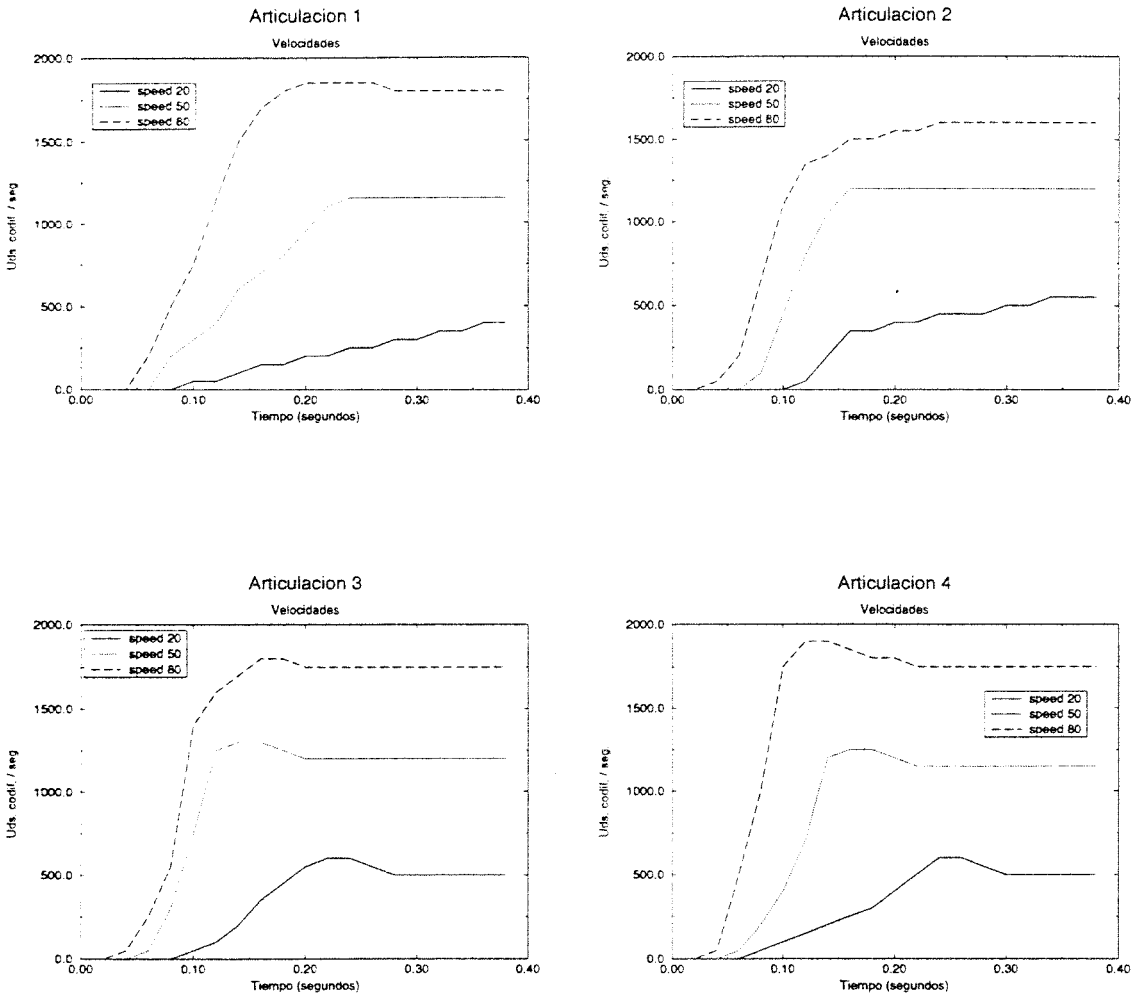

Figura 6.4 Velocidad de las articulaciones durante el transitorio para distintos valores de SPEED

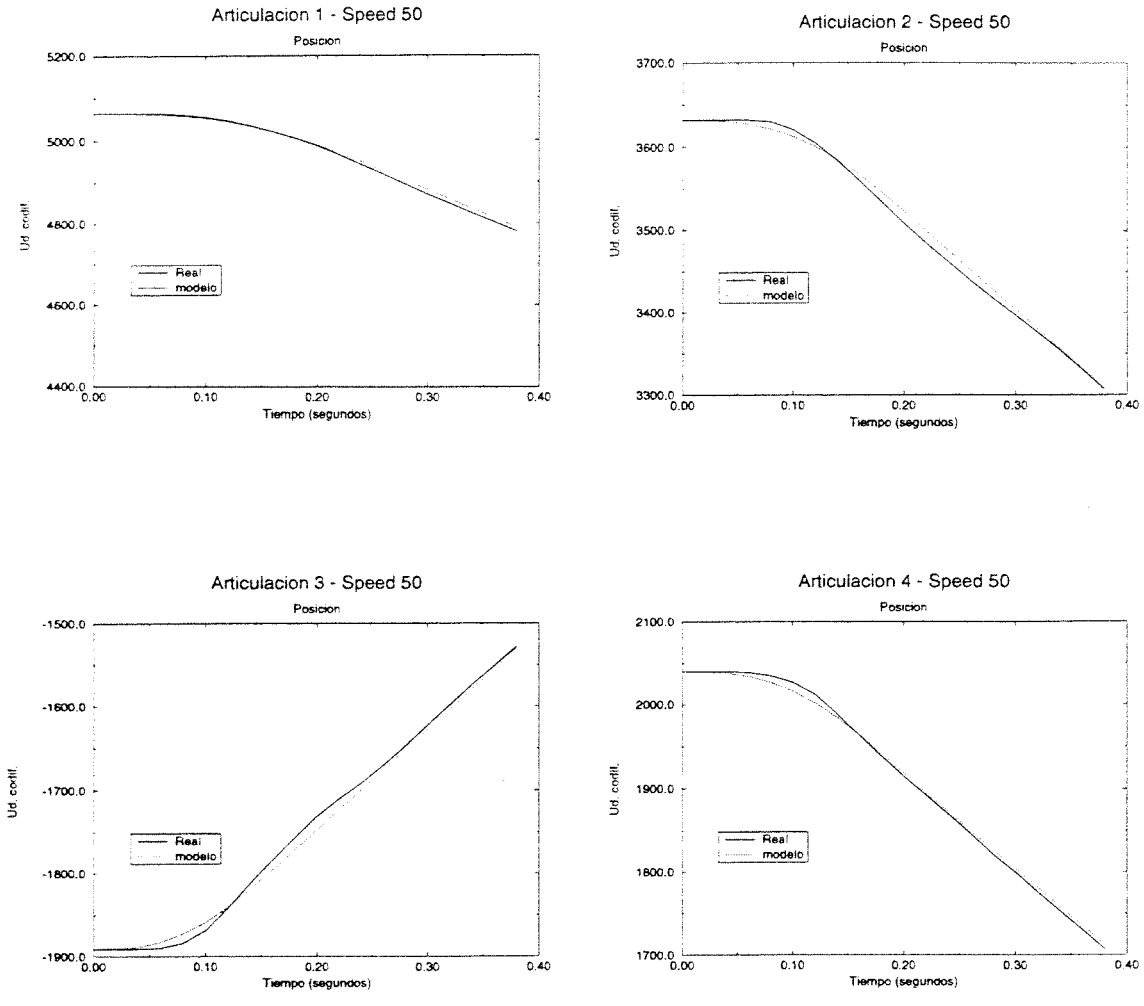

Figura 6.5 Posición real frente a posición modelada en el transitorio para SPEED=50

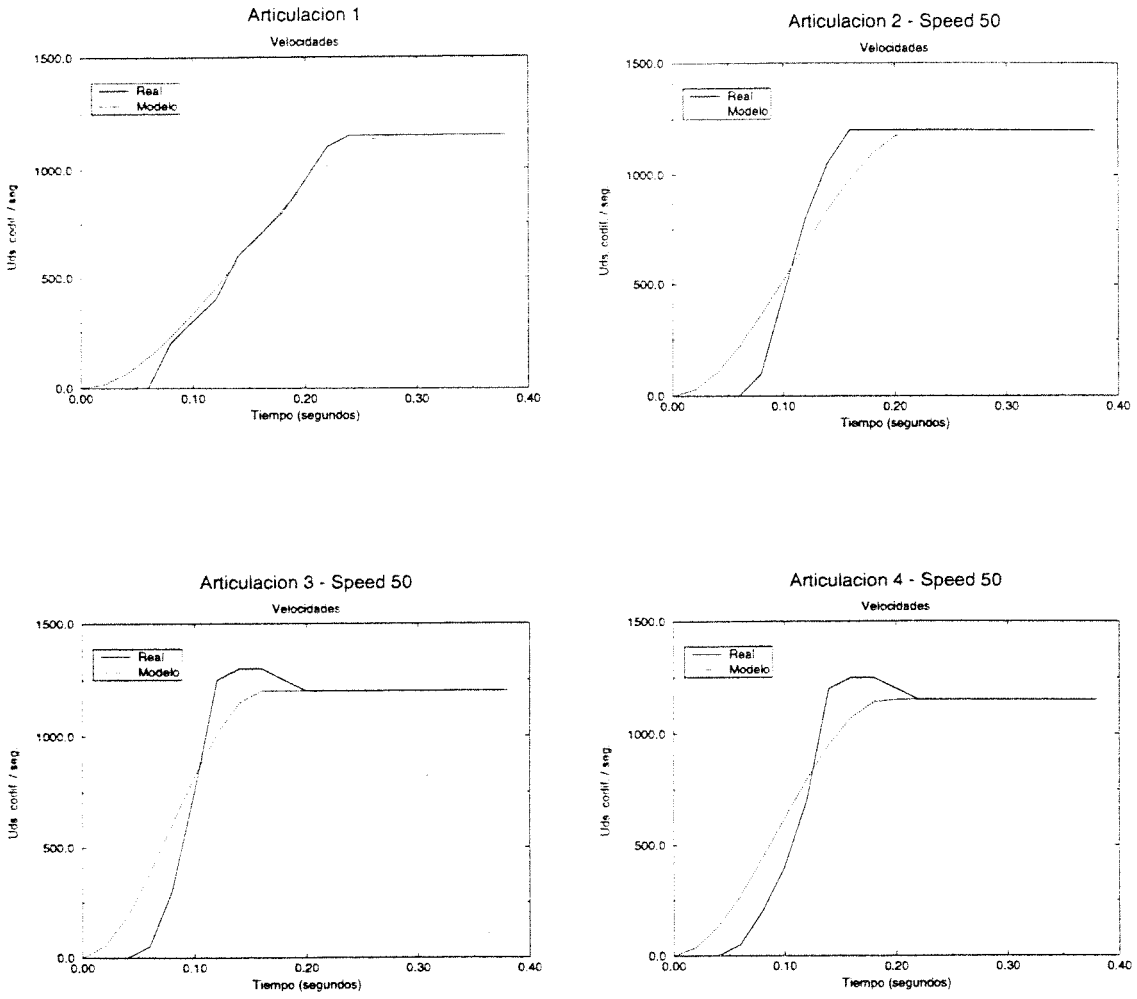

Figura 6.6 Velocidad real frente a velocidad modelada en el transitorio para SPEED=50

6.4 Ejemplo 1

En el primer ejemplo se estudia el comportamiento de los algoritmos en un sistema compuesto por dos robots tipo PUMA que se mueven en un entorno común en el que existe un obstáculo en forma de T.

La Figura 6.7-1 muestra la configuración inicial del robot 1, mientras que la Figura 6.7-5 muestra la configuración final deseada. Este problema de planificación fue resuelto considerando un entorno donde sólo existen el robot 1 y el obstáculo, utilizando para ello el elgoritmo descrito en el apéndice B, dando lugar a las posiciones intermedias que aparecen en dicha figura.

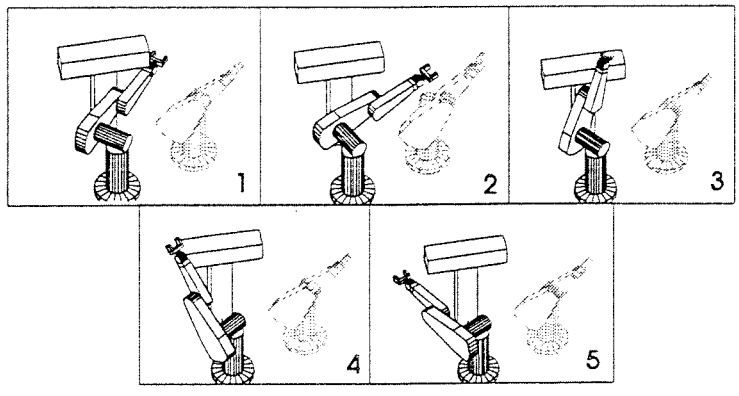

Figura 6.7 Puntos que definen el movimiento del robot 1 en el ejemplo 1.

Del mismo modo, en la Figura 6.8-1 y en la Figura 6.8-4 aparecen las configuraciones iniciales y finales para el segundo robot. Este problema fue resuelto del mismo modo que el anterior, dando lugar a un movimiento definido por las posiciones intermedias 2 y 3 de la misma figura. Esta planificación se realizó de una forma más simple al no interferir el obstáculo en el movimiento.

Como siguiente paso de la planificación desacoplada, se construyó el diagrama de coordinación, que aparece en la Figura 6.9-a. Como se puede comprobar, dicho diagrama no tiene solución, por lo que se le aplicó una de las modificaciones propuestas en esta tesis, en concreto la modificación tipo 2, que como se indica en el capítulo 3, consiste en apartar momentaneamente de su camino uno de los robots. El algoritmo correspondiente, determinó que el robot más adecuado para alterar su camino era el segundo, y la posición en que se

abandonaría el camino original coincide con la configuración inicial del robot. Este algoritmo encontró un nuevo tramo, siendo la posición final del nuevo tramo la que aparece en la 150-2. El nuevo tramo consistía en un giro de 10° de la primera articulación.

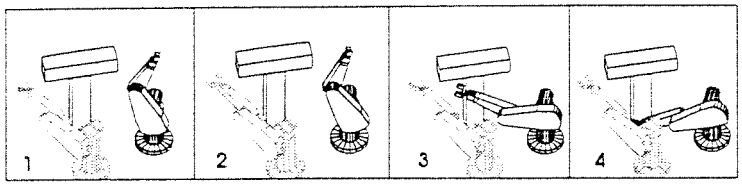

Figura 6.8 Puntos que definen el movimiento del robot 2 en el ejemplo 1.

Al añadir un nuevo tramo a los caminos iniciales se obtuvo el diagrama de coordinación que aparece a la derecha en la Figura 6.9, que como se puede comprobar tiene solución.

El último paso consistió en la determinación de la secuencia óptima de puntos de sincronización, para lo cual fue utilizado el procedimiento basado en el algoritmo A* que se describe en el capítulo 4. En este caso, se necesitan dos puntos de sincronización, que aparecen representados en la Figura 6.10.

En cuanto a los tiempos de C.P.U. empleados para cada una de estas fases utilizando una IBM RISC-6000, son los que aparecen en la ?.

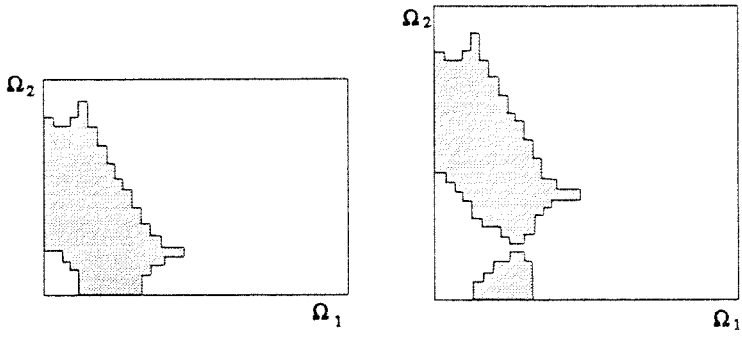

Figura 6.9 Diagrama de coodinación antes y después de la modificación necesaria en el ejemplo 1.

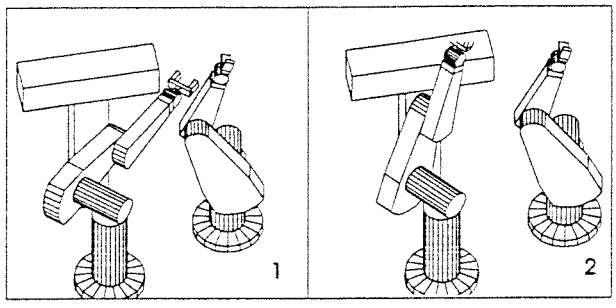

Figura 6.10 Puntos de sincronización necesarios en el ejemplo 1.

Tabla 6.5 Tiempos de C.P.U. en las distintas fases de la planificación del movimiento en el ehjemplo 1.

Fase	C.P.U.((seg.)
Camino Robot 1	62
Camino Robot 2	10
Clausura-SW	15
Selección	8
Búsqueda de camino	12
Obtención puntos sincronización	2

6.5 Ejemplo 2

Con este segundo ejemplo se pretende comparar los dos algoritmos propuestos para la obtención de puntos de sincronización: el basado en búsqueda en grafos mediante un algoritmo A^* y el basado en algoritmos evolutivos.Para ello, se estudia el comportamiento de dos robots SCORBOT ante un movimiento consistente en un giro de la base de cada uno de los robots. Para estudiar el comportamiento de los algoritmos con diagramas de

coordinación mayores, también se han considerado varios movimientos de ida y vuelta sobre el caso anterior.

En primer lugar se considera el caso más simple. En la Figura 6.11-1 y la Figura 6.11-3 se muestran respectivamente las configuraciones inicial y final del sistema. En cada uno de los robots, el camino considerado consiste en un único tramo en que la base gira un total de 180 grados (desde la posición absoluta -90 a la 90), mientras el resto de las articulaciones permanecen fijas. Como se puede comprobar en las figuras, existe una zona de posibles colisiones cuando ambos robots se encuentran en la zona intermedia de su camino. La Figura 6.12 muestra la clausura SW del diagrama de coordinación para dicho ejemplo, con un tamaño de 46×46 celdas. Sobre este diagrama de coordinación se han aplicado los dos algoritmos basados en el método de los rectángulos, con el fin de realizar un estudio comparativo. Para la aplicación del algoritmo de evolución se ha utilizado un mecanismo de selección proporcional, una población de 25 individuos con 8 puntos de sincronización y un porcentaje de individuos válidos en la población inicial del 40%. Los operadores genéticos utilizados se corresponden con los que mejor comportamiento proporcionan según el estudio comparativo del capítulo anterior, es decir *cruce simple* y *mutación simple con límites fijos*. Los algoritmos se ejecutaron en una estación de trabajo RISC 6000 320 H.

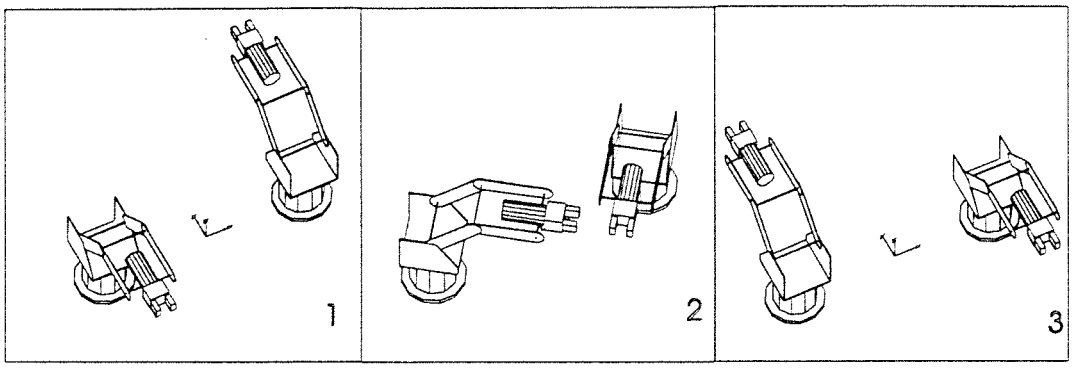

Figura 6.11 Configuración inicial, en el punto de sincronización y final en el primer caso del ejemplo 2.

El método de búsqueda basado en el algoritmo A* encontró una solución óptima en 0,69 s., consistiendo dicha solución en un único punto de sincronización localizado según se muestra en la Figura 6.11-2. El algoritmo evolutivo también encontró la misma solución, llegando a ésta en un caso medio en 4.98 s. (recuérdese que al tener los algoritmos evolutivos una componente aleatoria, distintas ejecuciones llegan a distintos resultados). Al contrario de lo que ocurre con el método A*, en el que no se dispone de ninguna solución hasta que no se completa el algoritmo, en el método evolutivo en cada momento se dispone de soluciones válidas. En la Figura 6.13 se muestra la mejor solución obtenida por el

algoritmo evolutivo en función del tiempo. El tiempo que transcurre hasta que el algoritmo evolutivo encuentra su primera solución se invierte sobre todo en la obtención de la población inicial. La línea discontinua muestra el valor óptimo, y el círculo indica el instante en que se encontró la solución con el método A^*. En este caso el algoritmo A^* encuentra la solución en menos tiempo que el necesario para la obtención de la población inicial.

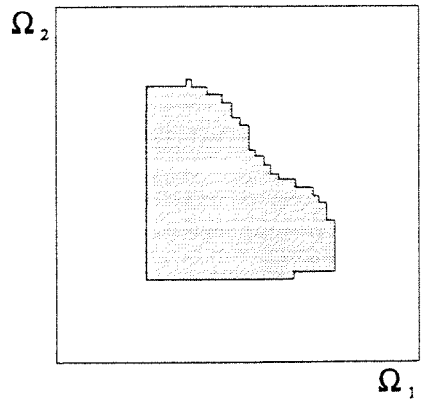

Figura 6.12 Diagrama de coordinación en el primer caso del ejemplo 2.

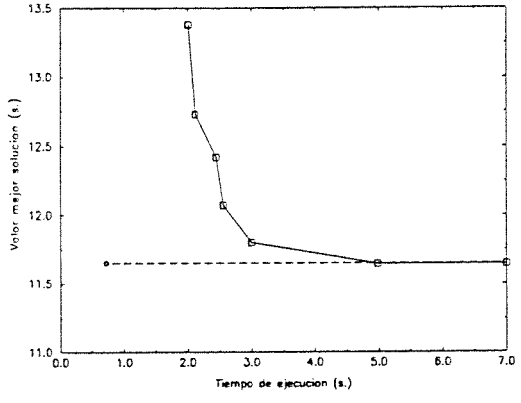

Figura 6.13 Evolución del mejor valor hallado en función del tiempo de ejecución del algoritmo para el primer caso del ejemplo 2.

Con un segundo caso se pretende estudiar el comportamiento de los dos algoritmos para diagramas de coordinación mayores. En este caso, el camino del primer robot consiste en dos tramos, un primero idéntico al del caso anterior al que se le añade un nuevo tramo simétrico correspondiente al camino de vuelta. El camino del segundo robot se mantiene inalterado. Para este caso se obtiene el diagrama de coordinación de 90×46 celdas de la Figura 6.14.

Para este segundo caso, el algoritmo A* necesitó un tiempo considerablemente mayor (50,38 s.) para encontrar una solución óptima, consistente en este caso en 3 puntos de sincronización. Este aumento del tiempo de cálculo se debe, además del aumento del tamaño del diagrama de coordinación, a la aparición de nuevas regiones de colisión, que reduce el número de sucesores redundantes e innecesarios. La misma solución también fue encontrada por el algoritmo evolutivo, en este caso en la generación 67, después de 13.42 s. La Figura 6.15 muestra la evolución en el tiempo de la mejor solución encontrada durante la ejecución de este algoritmo.

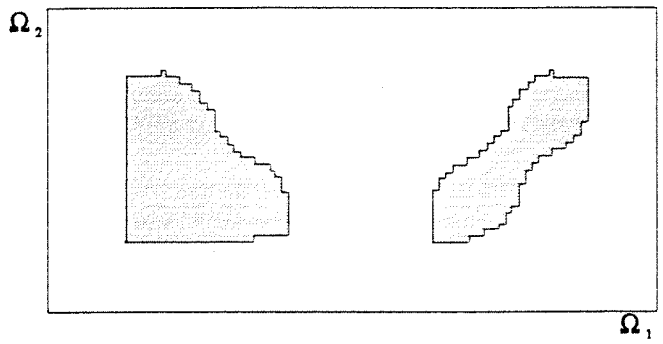

Figura 6.14 Diagrama de coordinación para el segundo caso del ejemplo 2.

Aún se consideró un tercer caso, con un diagrama de coordinación mayor. Ahora, ambos robots realizan un movimiento de ida y vuelta, dando lugar a un diagrama de coordinación con 90×90 celdas y 4 regiones de colisión. La Figura 6.16 muestra este nuevo diagrama. En este último caso, el algoritmo A* necesitó 4m. 27 s. para encontrar una solución óptima con 4 puntos de sincronización, siendo el valor de esta solución, es decir, el tiempo necesario para la ejecución del movimiento coordinado, de 19.76 s. El algoritmo de evolución, aunque no consigue encontrar la solución óptima, sí encuentra una solución muy próxima con un valor de 19,81 s. también con cuatro puntos de sincronización. Esta solución se encuentra en la generación 130 en 39,03 s.

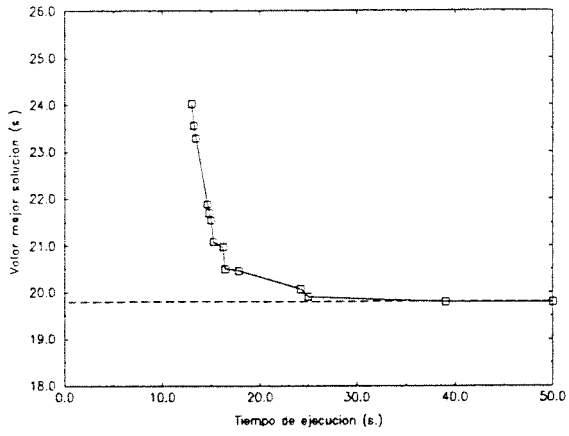

Figura 6.15 Mejor solución encontrada en función del tiempo de ejecución del algoritmo para el caso 2 del ejemplo 2.

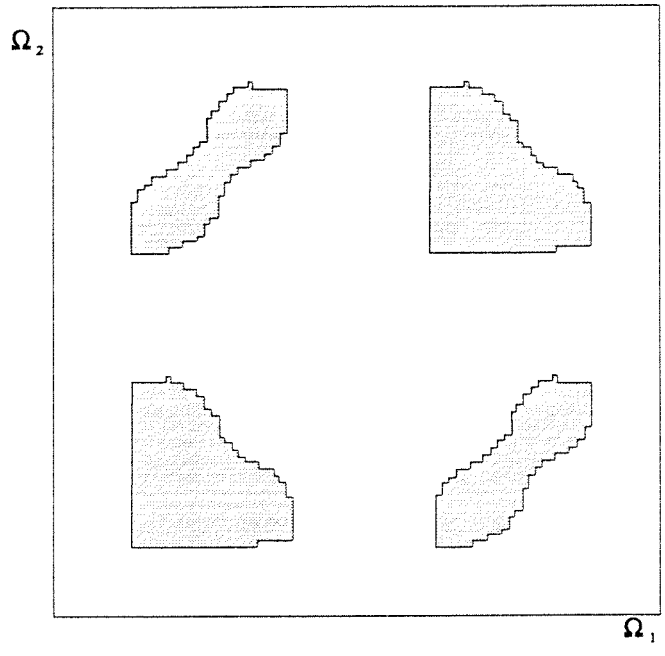

Figura 6.16 Diagrama de coordinación correspondiente al tercer caso del ejemplo 2.

A la vista de estas pruebas, y otras muchas realizadas durante el desarrollo de esta tesis, se puede llegar a las siguientes conclusiones en relación a estos dos algoritmos:

- El método basado en el algoritmo A* encuentra la solución óptima si esta existe, mientras el método evolutivo, ni puede asegurar encontrarla, ni aún encontrándola puede reconocerla. Esto último dificulta la elección de un criterio de terminación, lo que obliga a utilizar el algoritmo A* en el caso que se necesite la solución óptima.

- El método evolutivo va disponiendo de soluciones cada vez mejores a lo largo de la ejecución del algoritmo, de forma que en pocas generaciones se puede disponer de soluciones aceptables. Por el contrario el método A* encuentra una única solución al terminar el algoritmo. Por tanto, en caso de tener limitaciones sobre el tiempo de búsqueda, aún a costa de no encontrar una solución óptima, es interesante el uso del algoritmo evolutivo.

- En cuanto al tiempo de cálculo, el algoritmo A* obtiene buenos resultados en diagramas de coordinación pequeños (del orden del correspondiente al primer caso). Para diagramas mayores, el tiempo de cálculo crece rápidamente. Este crecimiento en relación al tamaño es mucho menor en el caso del algoritmo evolutivo.

- También tiene mucha importancia la disposición de las regiones de colisión. Para el algoritmo A* la situación más desfavorable se da cuando existen numerosas zonas de colisión de pequeño tamaño, lo que da lugar a un amplio espacio de búsqueda y numerosos caminos posibles. Por el contrario, el algoritmo evolutivo encuentra más dificultades cuando las posibles soluciones están muy restringidas por las zonas de colisión, debido a la dificultad de encontrar soluciones válidas.

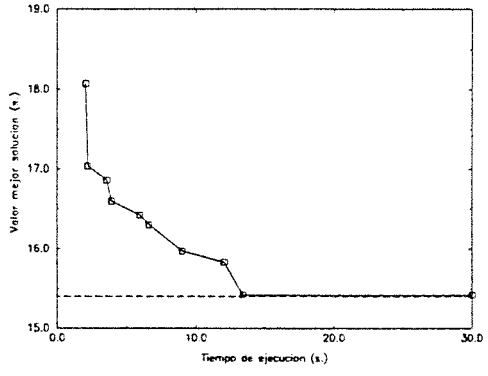

Figura 6.17 Mejor valor encontrado para el caso 3 del ejemplo 2.

Estos casos han sido ejecutados sobre un sistema real, consiguiendo los resultados esperados en cuanto a evitación de colisiones. También se ha comprobado un buen comportamiento de los modelos utilizados. La Tabla 6.6 proporciona para cada uno de los tres casos la diferencia entre el comportamiento real del sistema en cuanto a tiempo de ejecución del movimiento y el correspondiente al modelo propuesto.

Tabla 6.6 Tiempo real y tiempo obtenido con el modelo, necesario para la ejecución del movimiento coordinado en los distintos casos

	T. Real (s.)	T.Mod. (s.)
Caso 1	11.50	11.65
Caso 2	15.20	15.42
Caso 3	19.71	19.77

6.6 Ejemplo 3

En este tercer ejemplo se comprueba el resultado de las modificaciones propuestas ante un problema que da lugar a un diagrama de coordinación sin solución. La Figura 6.18 muestra las configuraciones inicial y final del sistema, donde uno de los robots SCORBOT realiza un movimiento de la primera articulación de 180° (robot 1), mientras el otro (robot 2) realiza un movimiento del elemento terminal en una posición tal que impide el paso del primero. Esta posición del segundo robot, bloqueando el paso del primero, hace que el diagrama de coordinación resultante, tal como se refleja en la Figura 6.19, no tenga solución.

A este diagrama de coordinación se la aplicaron las dos modificaciones propuestas en esta tesis, obteniéndose en ambos casos solución. Al aplicar la modificación tipo 1, el robot 1 fue elegido como robot primario, seleccionándose como posición más favorable para evitar el segundo robot aquella en la que el elemento terminal de este último se mantenía en un plano horizontal. En la Figura 6.20 se muestran algunas posiciones intermedias considerando el nuevo camino modificado. Las configuraciones en que aparece

el robot 1 en las figuras 1 y 3 se corresponden con las del inicio y el final del tramo a modificar. En la Figura 6.21 se muestra el diagrama de coordinación resultante.

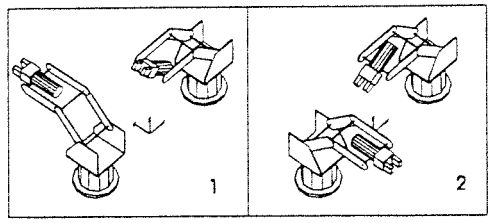

Figura 6.18 Posición inicial (1) y final (2) de los robots en el ejemplo 3.

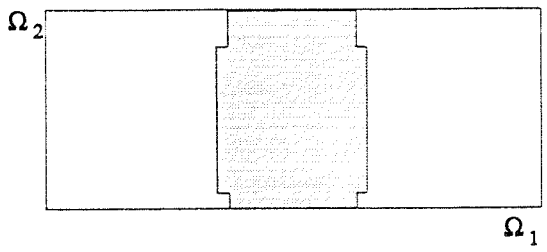

Figura 6.19 Diagrama de coordinación correspondiente a los caminos originales en el ejemplo 3.

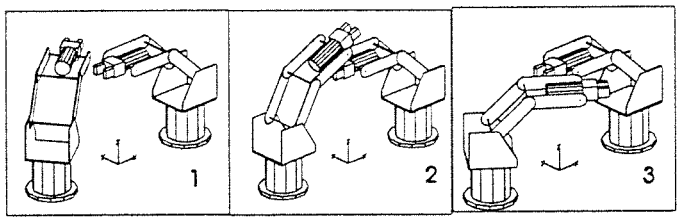

Figura 6.20 Puntos intermedios en el movimiento planificado después de realizar una modificación de tipo 1.

Los tiempos de C.P.U. necesarios para las distintas fases de la planificación son los que aparecen en la Tabla 6.7. Conviene destacar que en el módulo de selección, en el proceso de búsqueda de un camino coordinado, es necesario evaluar el estado de algunas celdas que no lo fueron previamente por estar incluidas dentro de la región correspondiente a la clausura SW. El método utilizado para la obtención de puntos de sincronización es el basado en búsqueda en grafos.

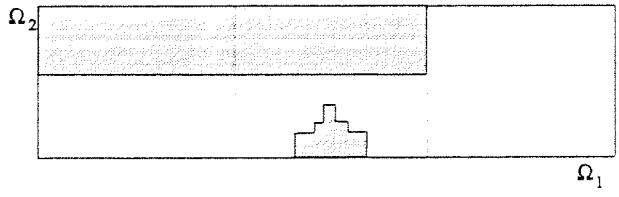

Figura 6.21 Diagrama de coordinación resultante tras la realización de la modificación tipo 1.

Tabla 6.7 Tiempos de C.P.U. invertidos en la planificación del movimiento al aplicar la modificación tipo 1.

Fase	C.P.U.(seg.)
Clausura SW	7
Selección	11
Búsqueda de camino	31
Obtención puntos de sincronización	1

También se ha resuelto el problema utilizando la modificación tipo 2, en la que el robot primario es el 2. La Figura 6.22 muestra la configuración para dicho robot encontrada para el punto final del nuevo tramo (α). En la Figura 6.23 se muestra el diagrama de coordinación resultante tras aplicar la modificación.También en este caso se utilizó el método basado en búsqueda en grafos para la obtención de los puntos de sincronización. En este caso, la solución óptima consta de dos puntos de sincronización, que se corresponden con las configuraciones que aparecen en la Figura 6.22. En la Tabla 6.8 se indican los tiempos de C.P.U. en cada una de las fases.

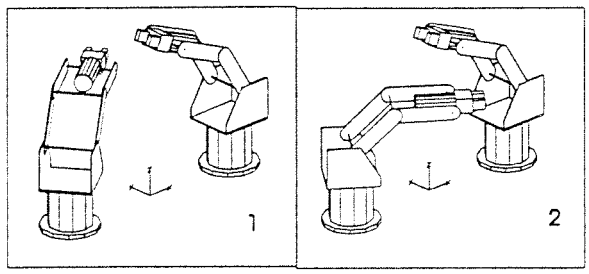

Figura 6.22 Puntos de sincronización resultantes en la planificación del ejemplo 2 tras una modificación tipo 2.

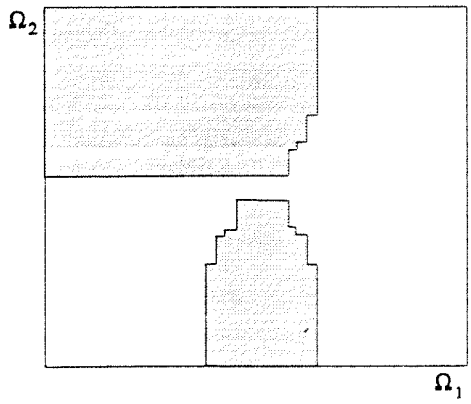

Figura 6.23 Diagrama de coordinación tras la aplicación de la modificación tipo 2.

Tabla 6.8 Tiempos de C.P.U. invertidos en la planificación del movimiento al aplicar la modificación tipo 2.

Fase	C.P.U.(seg.)
Clausura SW	7
Selección	11
Búsqueda de camino	23
Obtención puntos de sincronización	0.75

6.7 Ejemplo 4

En este último ejemplo se ha comprobado el funcionamiento del sistema para la coordinación de tres robots. Las configuraciones inicial y final aparecen representadas en la Figura 6.24-1 y la Figura 6.24-5. En esta última figura aparecen reflejados los números que servirán parar identificar a cada uno de los robots.

En primer lugar se procedió a la obtención del diagrama de coordinación que en este caso se corresponde con un espacio tridimensional. Como se indicó en el capítulo 3, la obtención de este diagrama tridimensional se reduce a la obtención de tres diagramas bidimensionales, en cada uno de los cuales se representan las colisiones entre dos de los robots. Los tres diagramas obtenidos aparecen en la Figura 6.25.

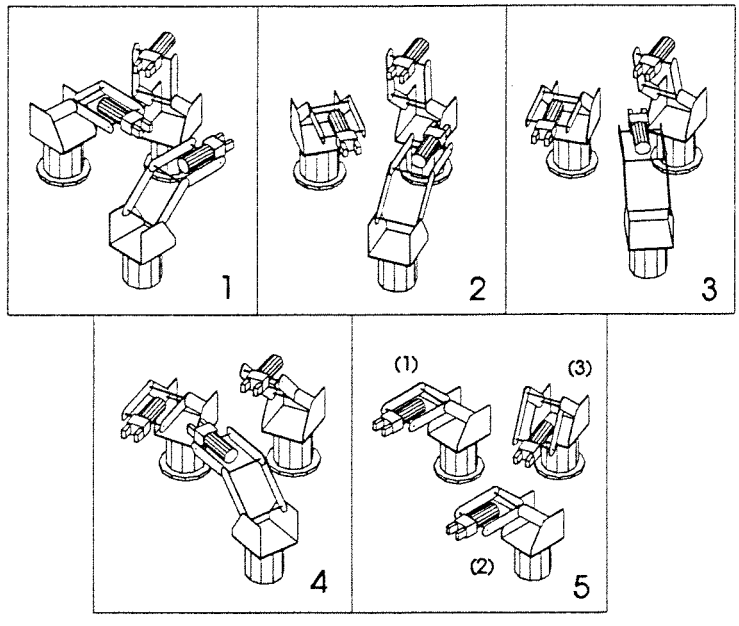

Figura 6.24 Configuraciones inicial, intermedias y final correspondientes al ejemplo 3.

Los tiempos de C.P.U. necesarios para la obtención de los diagramas de coordinación, utilizando una RISC-6000, fueron los siguientes:

- Diagrama de coordinación robot 1-robot 2: 45 s.
- Diagrama de coordinación robot 1-robot 3: 29 s.
- Diagrama de coordinación robot 2-robot 3: 51 s.
- Total: 1m. 57 s.

Para la obtención de la secuencia de puntos de sincronización y debido al tamaño del espacio de búsqueda se ha utilizado el método de los rectángulos, resuelto mediante algoritmos evolutivos. En dicho algoritmo se ha utilizado selección proporcional, y una población de 25 individuos con 8 puntos de sincronización. Además, como operadores genéticos se han utilizado el cruce simple y la mutación simple con límites fijos.

La mejor solución obtenida proporciona un tiempo de ejecución del movimiento de 8.37 s., necesitando tres puntos de sincronización. Las configuraciones correspondientes a dichos puntos son las que aparecen en Figura 6.24-2,3 y 4. El tiempo de C.P.U necesario para simular 100 generaciones ha sido 35 s.

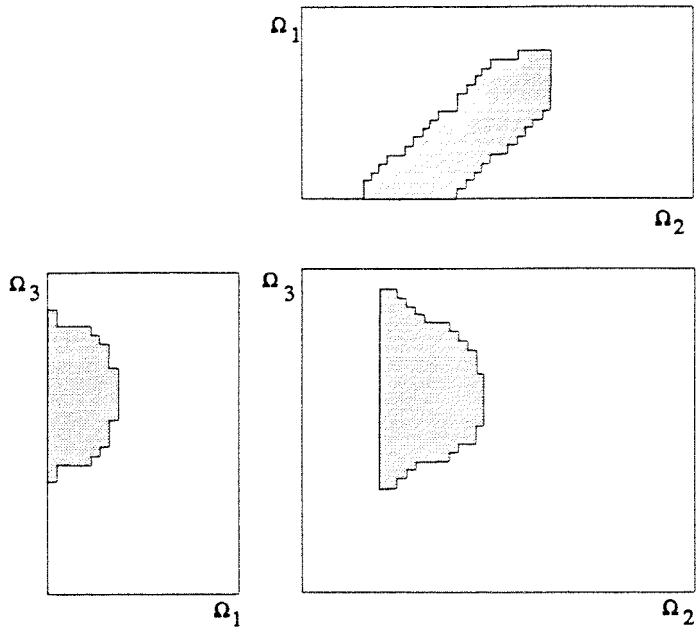

Figura 6.25 Diagramas de coordinación del ejemplo 3.

Capítulo 7

Conclusiones

En este capítulo se exponen las conclusiones obtenidas en esta tesis, así como algunas de las líneas futuras de investigación.

Se ha desarrollado un sistema para la planificación del movimiento de varios robots manipuladores que operan en un entorno común de trabajo. El problema planteado parte de una configuración inicial y una final para cada uno de los robots, y se pretende generar un programa para cada robot que evite las colisiones de cada uno de los robots con los obstáculos fijos del entorno y con cualquiera de los otros robots.

Con este propósito se han utilizado técnicas de planificación desacoplada que han obligado a la resolución de varios problemas:

- Obtención de un camino para cada uno de los robots, para lo cual se ha utilizado un procedimiento heurístico de búsqueda en el espacio de las configuraciones basado en un algoritmo del tipo A^*.

- Construcción del diagrama de coordinación, para lo cual se ha propuesto un algoritmo en el que no es necesario evaluar el estado de todas y cada una de las celdas (se evalúan las celdas libres de colisiones y las correspondientes al contorno de las zonas de colisiones).

- Estudio de distintas técnicas para la detección de colisiones cuando los robots se encuentran en configuraciones determinadas. Asimismo ha sido necesario ampliar estas técnicas para determinar el estado de colisión de las celdas (se consideran rangos de colisiones).

Para los casos en que el diagrama de coordinación no tiene solución, se han propuesto dos métodos para la modificación parcial de los caminos iniciales. Estos

métodos no garantizan encontrar un nuevo diagrama de coordinación con solución; sin embargo, en la mayor parte de los casos prácticos han demostrado ser métodos eficientes, que realizan la modificación del camino con un coste de cálculo pequeño. Además, una parte importante del primitivo diagrama de coordinación continúa siendo válido.

Para la obtención de los programas de los robots a partir del diagrama de coordinación se han propuesto básicamente dos métodos, ambos basados en la colocación de puntos de sincronización en el diagrama de coordinación. El *método de los rectángulos* garantiza que no existirán colisiones entre los robots, aunque éstos realicen el movimiento de una forma muy distinta a la planificada. Esta seguridad va a suponer la necesidad de utilizar un mayor número de puntos de sincronización y un mayor tiempo de ejecución del movimiento coordinado.

El *método de la banda de la trayectoria* proporciona soluciones mejores en cuanto a tiempo de ejecución del movimiento, pero es dependiente del modelo con el que se realiza la planificación, y por tanto no garantiza la inexistencia de colisiones si las condiciones del movimiento se separan de las modeladas. Además presenta el inconveniente de necesitar unos algoritmos más complejos.

La implementación de estos métodos se ha planteado como un problema de optimización, en el que se pretende optimizar el tiempo de ejecución del movimiento, por lo cual ha sido necesario un modelo de las trayectorias generadas por los controladores de los robots. Dos algoritmos han sido utilizados para resolver el problema: el primero basado en procedimientos de búsqueda en grafos mediante algoritmos heurísticos del tipo A^*, mientras que el segundo está basado en algoritmos evolutivos.

Las diferentes filosofías de ambos métodos hacen que tengan un comportamiento muy distinto, cada uno con ventajas e inconvenientes. Así, el primer método garantiza encontrar la solución óptima si esta existe, circunstancia que es imposible garantizar con un método evolutivo. Sin embargo, este último método proporciona, a medida que se ejecuta el algoritmo, soluciones válidas cada vez mejores, de forma que desde las primeras etapas del algoritmo se dispone de soluciones "razonablemente" buenas. Por el contrario, en el método de búsqueda A^*, solo se obtiene una única solución, una vez que el algoritmo ha finalizado.

Como futuras líneas de investigación se proponen:

- Utilización de los métodos propuestos como parte de un sistema de planificación de tareas.

- Adaptación de los métodos para movimientos repetitivos de los robots. Así, en el caso de dos robots, el diagrama de coordinación se convertiría en una superficie

cilíndrica si sólo uno de los robots presenta movimiento repetitivo, o bien en la superficie de un toro si el movimiento repetitivo se da en ambos robots.

- Elaboración de nuevos algoritmos para mejorar la eficiencia en la detección de colisiones. En esta línea se está trabajando en un nuevo sistema CAD especialmente dedicado a la representación de robots y sus entornos.

- Nuevos métodos de modificación de caminos para diagramas de coordinación sin solución, especialmente para el caso de más de dos robots.

- Ampliación del problema de la obtención de la secuencia de puntos de sincronización con la inclusión de nuevas variables que intervengan en la optimización, como las velocidades y aceleraciones.

- Estudio del comportamiento de nuevas funciones de evaluación, tanto admisibles como semiadmisibles, para el procedimiento de búsqueda de puntos de sincronización basado en el algoritmo A^*, así como de nuevas técnicas y operadores genéticos para los algoritmos evolutivos.

Apéndice A

Algoritmos de Búsqueda en Grafos Implícitos

A.1 Introducción

Este apéndice describe diferentes métodos básicos de resolución de problemas de búsqueda en grafos basados en técnicas de Inteligencia Artificial, con especial atención al algoritmo A^*, uno de los más conocidos métodos heurísticos de búsqueda [3], [84], [93], [98], [103], [113].

Se define [103] un proceso de búsqueda P como la cuádrupla:

$$P=\{G,N_{ini},T,F\}$$

donde

G es el conjunto de estados del problema (espacio de estados)
$N_{ini}\in D$ es el estado inicial
T es un conjunto de transformaciones que hacen que el problema pase de un estado a otro.
$F\subseteq G$ es el conjunto de estados finales

El *Grafo del Espacio de Estados* es una representación formal de un problema. Los estados del problema se representan mediante *nodos*, siendo N_{ini} el nodo correspondiente al estado inicial, y las transformaciones de T como un conjunto de *arcos* o *enlaces*, de forma que existirá un arco desde N_i *a* N_j si existe $t\in T$ tal que al aplicar t cuando el problema se encuentra en el estado N_i éste pasa al estado N_j. En estas condiciones, se dice que N_j es *sucesor* o *hijo* de N_i, y que N_i es *antecesor* o *padre* de N_j.

Un *camino* se define como una secuencia de nodos $N_1,N_2,...,N_k$, donde cualquier N_i es sucesor de N_{i-1}. Una solución al problema será un camino desde N_{ini} hasta cualquier nodo

perteneciente a F. En lo sucesivo se considerará también un coste asociado a cada arco que se denominará $c(N_i,N_j)$, y se definirá el coste asociado a un camino como la suma de los costes de los arcos atravesados (esta suposición es cierta para los problemas que se presentan en esta tesis).

A.2 Métodos básicos de búsqueda

Una forma simple de realizar el proceso de búsqueda es partir del nodo inicial y recursivamente ir expandiendo nodos, es decir, obtener sus sucesores hasta alcanzar un estado final. Muchos métodos parten de esta idea, entre los que se puede destacar:

- *Método de escalada (Hill-Climbing).* Esta estrategia consiste en ir expandiendo nodos partiendo del inicial. En cada paso se selecciona el mejor de estos sucesores como siguiente nodo a ser expandido. Esta estrategia se denomina *irrevocable* ya que no permite volver a expandir nodos previamente rechazados. La mayor ventaja de este método es su simplicidad mientras que el mayor inconveniente es el hecho de no garantizar el encontrar una solución.

- *Búsqueda en profundidad (Deep-First).* Frente a las estrategias irrevocables se presenten las estrategias *tentativas* o *retroactivas*, donde las alternativas no seleccionadas no son definitivamente rechazadas, sino que son almacenadas para ser utilizadas posteriormente si fuera necesario. Esto va a obligar a generar un árbol, denominado *árbol de búsqueda*, cuya raíz es el nodo inicial de forma que los nodos tipo *hoja* se corresponden con los elementos del grafo no expandidos y los interiores con los ya expandidos. En esta estrategia de búsqueda, después de cada expansión, uno de los hijos generados es elegido como siguiente nodo a ser expandido. En caso de que no fuera posible expandir ninguno de los hijos, se escogería el nodo tipo hoja que se encuentre a más profundidad en el árbol.

- *Búsqueda en anchura (Breadth-First).* En esta estrategia tentativa se da una mayor prioridad a la hora de ser elegidos para la expansión a aquellos nodos del árbol que se encuentren más próximos a la raíz, es decir, antes de expandir un nodo, se deben haber expandido todos aquellos que se encuentren a una profundidad menor.

La principal ventaja de estos dos últimos métodos es que son completos, esto es, aseguran encontrar una solución se ésta existe.

A.3 Métodos heurísticos de búsqueda en grafos

La gran desventaja de los métodos descritos en el apartado anterior radica en el hecho de expandir nodos de una forma puramente arbitraria, sin tener en cuenta la naturaleza del problema a resolver. Esto conlleva búsquedas extensivas, que para problemas complejos puede dar lugar a una *explosión combinatoria*. Disponer de conocimiento específico de un problema puede llevar a reducir la búsqueda. Este tipo de información acerca de un problema se denomina *conocimiento heurístico*.

Esta información heurística se puede aplicar en los problema de búsqueda en grafos en alguna de estas formas:

- Decidir cual será el siguiente nodo a expandir, en lugar de utilizar un método de selección arbitrario.

- Durante la expansión de un nodo, seleccionar los sucesores a expandir, en lugar de expandir todos los posibles sucesores.

- Descartar nodos del árbol como candidatos a ser expandidos.

La primera de ellas es la utilizada con mayor profusión en la literatura. La forma más usual de representar numéricamente la información heurística de un nodo es mediante el uso de *funciones heurísticas de evaluación* o *funciones objetivo f(N)*, donde, en general, los nodos con un valor de la función más pequeño son los que se consideran más prometedores. El valor de la función dependerá del nodo N, del objetivo, de la información generada en la búsqueda hasta ese punto, y en general de cualquier otro tipo de conocimiento adicional sobre el problema.

Una de las estrategias más conocidas que utilizan información heurística se denomina *Búsqueda por el Mejor Nodo* o *Búsqueda Ordenada (Best First)*. Esta técnica utiliza la información heurística para decidir, entre todos los nodos tipo hoja del árbol, cual será el siguiente a expandir. Esta estrategia tiene numerosas variantes, entre las que se incluye el algoritmo A^*, que se diferencian normalmente entre sí por la función de evaluación utilizada. Nótese que las estrategias retroactivas anteriores son casos particulares de la búsqueda ordenada: en la búsqueda en profundidad la función de evaluación se puede considerar como la profundidad del nodo en el árbol de búsqueda cambiada de signo, mientras que en la búsqueda en anchura es la propia profundidad del nodo.

A.4 Algoritmo A*

El algoritmo A* [47],[84] sigue una estrategia de Búsqueda Ordenada. Su característica fundamental consiste en que garantiza encontrar un camino de coste mínimo, siempre que este camino exista, si la función de evaluación cumple unas determinadas condiciones que se describirán posteriormente.

El método consiste en explorar el espacio de estados partiendo de N_{ini} mediante la expansión de nodos hasta llegar a un nodo perteneciente a F. Para decidir cuál será el siguiente nodo a expandir se utiliza una función de evaluación $f(N)$ definida del siguiente modo:

$$f(N)=g(N)+h(N) \tag{A.1}$$

donde:

$g(N)$ es el coste del camino entre el nodo N_{ini} y el nodo N.

$h(N)$ es una estimación heurística del coste mínimo $h^*(N)$ del camino entre N y un nodo perteneciente a F.

El algoritmo A* irá formando el árbol de búsqueda. Para preservar el camino, será necesario utilizar un puntero que señale desde un nodo hacia su antecesor. De esta forma, cuando se llega a un nodo objetivo, la forma de reconstruir el camino será recorrer estos punteros desde el nodo objetivo hasta N_{ini}.

A continuación se describe el algoritmo [62], donde se utilizarán los siguientes conjuntos:

- ABIERTOS: Conjunto de nodos que se han generado, pero que no han sido expandidos (se corresponden con los nodos hoja del árbol de búsqueda).
- CERRADOS: Conjunto de nodos ya expandidos (nodos interiores del árbol).

Sobre el conjunto ABIERTO están definidas las siguientes funciones:

- PRIMERO(ABIERTOS): Devuelve el nodo con un valor inferior de la función objetivo f, eliminándolo del conjunto.
- INSERTA(N,ABIERTOS): Inserta el nodo N en el conjunto ABIERTOS.

- BORRA(N,ABIERTOS): Elimina en nodo N de ABIERTOS
- PERTENECE(N,ABIERTOS): Evalúa si N pertenece a ABIERTOS.

Algoritmo A*

```
EMPEZAR
    INSERTA(N_ini,ABIERTOS);
    MIENTRAS ABIERTOS no está vacío
        N ← PRIMERO(ABIERTOS);
        SI N ∈ G salir del bucle mientras;
        Expandir N;
        PARA cada N' sucesor de N
            SI N' ∉ (ABIERTOS ∪ CERRADOS)
                Evaluar f(N');
                INSERTA(N',ABIERTOS);
            SINO
                SI g(N')>g(N)+c(N,N')
                    Redirigir el puntero de N' hacia N;
                    SI N' ∈ ABIERTOS
                        BORRA(N',ABIERTOS);
                        INSERTA(N',ABIERTOS);
                    FIN SI
                    SI N'∈ CERRADOS
                        Actualizar los sucesores de N'
                    FIN SI
                FIN SI
            FIN SI
        FIN PARA
    FIN MIENTRAS
    SI ABIERTO no está vacío
        devolver el camino
    SINO
        camino no encontrado
    FINSI
FIN
```

Figura A.1 Pseudocódigo del algoritmo A*

Conviene analizar más en profundidad las acciones a realizar cuando al expandir un nodo N aparece un sucesor N' que ha sido previamente generado, lo que implica que, disponiendo previamente de un camino hasta el nodo N', aparece un nuevo camino para llegar al mismo nodo. Si este nuevo camino es mejor que el anterior será necesario modificar el árbol de búsqueda para que recoja esta circunstancia. En este caso el antiguo puntero de N' a su antecesor se modifica para que señale a N. Si el nodo N' no había sido todavía expandido, es decir, se encuentra en ABIERTOS, bastará con incluir los nuevos datos relacionados con la función de evaluación del nodo.

Sin embargo, si el nodo N' había sido previamente expandido, es decir se encuentra en CERRADOS, las acciones son más complicadas. Una modificación del valor de la función de evaluación para N' llevará ligada una modificación de las funciones de evaluación de todos sus sucesores. Una posible alternativa consiste en no realizar la modificación inmediatamente, sino eliminar los sucesores de N' y volver a introducir dicho nodo en el conjunto ABIERTOS, para que vuelva a ser expandido.

A.5 Propiedades del algoritmo A*

Algunas de las propiedades más importantes que tiene el algoritmo A* son las siguientes:

Es completo: Un algoritmo es completo si encuentra solución cuando ésta existe. El algoritmo A* es completo tanto para grafos finitos, propiedad común a todos los algoritmos basados en Búsqueda Ordenada, como para grafos infinitos [84].

Admisibilidad: Se dice que un algoritmo es admisible si garantiza encontrar la solución óptima cuando ésta existe. El algoritmo A* es admisible si la función heurística h cumple la siguiente condición [84]:

$$0 \leq h(N) \leq h^*(N) \quad \forall N$$

En estas circunstancias se dice que h es una función heurística *admisible*. Como caso particular, la función $h(N)=0$, que sería una función heurística "no informada", es una función admisible, equivalente a una búsqueda en anchura.

Si C^* es el camino de menor coste desde el nodo inicial hasta un nodo objetivo, es decir $h^*(N_{ini})$, se puede demostrar [89] que si la función heurística es admisible cumplirá que cualquier nodo en ABIERTOS con $f(N)<C^*$ será expandido por A*. Como consecuencia de este resultado, si se conoce un camino desde el nodo inicial a un nodo objetivo con

coste C', donde lógicamente $C' \leq C^*$, cualquier nodo que se obtenga al realizar una expansión tal que $f(N)>C'$ no será necesario incluirlo en ABIERTOS, ya que se puede garantizar que nunca será elegido para ser expandido. Este resultado puede llevar a un considerable ahorro en la memoria necesaria para almacenar este conjunto.

Dominancia: Dados dos algoritmos A_1 y A_2, se dice que A_1 *domina* a A_2 si cada nodo expandido por A_1 también es expandido por A_2. Del mismo modo, A_1 domina estrictamente a A_2 si A_1 domina a A_2 pero A_2 no hace lo propio con A_1. La dominancia es un concepto que va ligado a la eficiencia: un algoritmo que domine a otro necesitará menos nodos para encontrar una solución.

Dadas dos funciones heurísticas, se dice que h_1 está *más informada* que h_2 si se cumple:

$$h_1>h_2 \quad \forall\, N$$

En estas condiciones, dado el algoritmo A^*_1 con una función heurística h_1 y un algoritmo A^*_2 con una función heurística h_2, de forma que h_1 está más informada que h_2, se puede demostrar [89],[84] que A^*_1 domina a A^*_2. Como ambas son admisibles por definición, se puede afirmar que el primer algoritmo encontrará el óptimo expandiendo menos nodos que el segundo.

Restricción de monotonía: Una función heurística h se dice que satisface la *restricción de monotonía* si cumple:

$$h(N)\leq c(N,N')+h(N') \quad \forall\, N,N' \,/\, N' \in \mathit{sucesor}(N)$$

por otro lado una función heurística es *consistente* si cumple:

$$h(N)\leq c(N,N')+h(N') \quad \forall\, N,N'$$

aparentemente la consistencia es una restricción mayor que la monotonía, sin embargo se puede demostrar [89] que ambas son equivalentes.

A partir de estas definiciones, se puede demostrar que cualquier función monótona es admisible. En efecto, aplicando el resultado anterior cuando γ es un nodo objetivo:

$$h(N)\leq c(N,\gamma)+h(\gamma) \quad \gamma \in F$$

pero $h(\gamma)=0$ y $c(N,\gamma)=h^*(\gamma)$, por lo que cumple la condición de admisibilidad.

La restricción de monotonía permite obtener algunas conclusiones interesantes:

- El algoritmo A* con una función heurística monótona encuentra caminos óptimos para todos los nodos expandidos

$$g(N)=g^*(N) \quad \forall N \in \text{CERRADOS}$$

Este resultado asegura que en esas condiciones, al expandir un nodo nunca aparecerá uno previamente generado que ya hubiera sido expandido por lo que no es necesario considerar esa circunstancia en el algoritmo.

- La monotonicidad implica que los valores de la función de evaluación de la secuencia de nodos expandidos por A* es no decreciente.

A.6 Complejidad de A*

En el estudio de la complejidad de un algoritmo se siguen principalmente en la literatura dos tipos de análisis. El primero de ellos está basado en la suposición de que el algoritmo se comporta de la peor manera posible (*análisis del peor caso*), mientras el segundo intenta estudiar el comportamiento *medio* de dicho algoritmo (*análisis probabilístico*).

Dentro del análisis del *peor caso* [28], los primeros estudios intentan determinar el número de operaciones realizadas por A* en función del número total de estados M. Se parte de la hipótesis de que el número de sucesores de cada nodo está superiormente acotado, y por tanto la expansión del nodo supone un número constante de operaciones.

Bajo las anteriores hipótesis, se demuestra que de una forma general A* puede llegar a expandir $O(2^M)$ nodos. Si la heurística es *admisible*, se puede demostrar [81] que bajo ciertas condiciones la complejidad se reduce a $O(M^2)$. Si además la función heurística es monótona, se puede afirmar que la complejidad máxima es $O(M)$.

Otros estudios de complejidad intentan estudiar ésta en función del número de arcos del camino óptimo a un estado final. Si L es este número, Ibaraki [53] demostró que para heurísticas casi-perfectas la complejidad de A* es $O(L)$. Se han realizado de igual modo estudios para heurísticas admisibles, en árboles uniformes (K sucesores por nodo) en función de la bondad de la heurística (diferencia entre h y h^*). Suponiendo r una constante positiva, se llega a las siguientes conclusiones [37]: Si $(1-r)h^* \le h \le h^*$, se puede demostrar que la complejidad es $O(K^{rL})$. Por otro lado, si $h^* - \sqrt{h^*} \le h \le h^*$ entonces la complejidad será $O(\sqrt{L}\ K^{\sqrt{L}})$. Finalmente, la complejidad sólo será lineal si se cumple $h^* - r \le h \le h^*$, concretamente $O(M\ K^r)$.

Sin embargo, estos estudios basados en el *peor caso* suelen ser muy conservativos, pareciendo más lógico realizar un estudio probabilístico que determine el comportamiento medio del algoritmo A*. En [89] se realiza un profundo estudio de este punto de vista de la complejidad, donde se demuestra que salvo para heurística muy próximas a la información perfecta, la complejidad del algoritmo es exponencial con la longitud del camino solución.

A.7 Heurísticas no admisibles

La admisibilidad, como se comentó anteriormente, es la propiedad que permite asegurar que el óptimo será encontrado. Sin embargo, el efecto que puede producir es un tiempo largo de búsqueda, muchas veces entre soluciones con diferencias entre sus costes no significativas. En muchas situaciones se prefiere encontrar una solución de forma rápida a encontrar el óptimo. Para estos casos puede ser válido utilizar heurísticas no admisibles. En otras circunstancias, se pretende llegar a un equilibrio entre la bondad de la solución encontrada y el esfuerzo computacional requerido para encontrarla, es decir, una solución que se encuentre de forma rápida y con un coste suficientemente cerca del óptimo.

En este apartado se verán una serie de modificaciones al algoritmo A*, donde se utilizarán heurísticas no admisibles y que por tanto no garantizan encontrar el óptimo, y heurísticas que garantizan que la diferencia entre el coste de la solución que encuentran y el mínimo global está acotada. Estas heurísticas se denominan ϵ-admisibles.

A.7.1 Asignación de pesos a g y h

Pohl [94] propone una variación de la función de evaluación, asignándole pesos a g y h mediante la función:

$$f_w(N)=(1-w)\cdot g(N)+w\cdot h(N) \qquad 0\leq w\leq 1 \tag{A.2}$$

Esta función pretende potenciar el efecto de cada uno de los sumandos de la ecuación. El aumento de la importancia relativa de la función g hace que la búsqueda se vaya acercando a una búsqueda en anchura, mientras que potenciar la función h implica acercarse a una búsqueda ordenada, dando menos importancia al camino recorrido.

Si h es una función heurística admisible, se puede asegurar que esta heurística ponderada también lo será en el intervalo $0 \leq w \leq ½$. En el resto del intervalo dependerá de lo lejos que se encuentre h de h^*.

A.7.2 Ponderación Dinámica

Como ampliación a la anterior, Pohl [93] propone una heurística ϵ-admisible, donde los pesos van evolucionando dinámicamente a medida que el algoritmo avanza. La idea consiste en realizar al principio una búsqueda más "agresiva" para posteriormente, cuando se esté cerca de algún nodo objetivo, hacer una búsqueda más exhaustiva. La función que se propone es:

$$f(N)=g(N)+h(n)+\epsilon\left[1-\frac{d(N)}{D}\right]h(N) \qquad \text{(A.3)}$$

donde $d(N)$ es la profundidad en el árbol de búsqueda del nodo N y D es una cota superior estimada de la profundidad del nodo objetivo.

Es fácil de demostrar que, si h es admisible, este algoritmo es ϵ-admisible. Para ello basta con demostrar que es capaz de encontrar una solución con un coste menor que $(1+\epsilon)C^*$, donde C^* es el coste óptimo.

Este algoritmo es muy fácil de implementar, ya que sólo cambia ligeramente la función objetivo con respecto al algoritmo A*. Sin embargo, para poder ser utilizado, se debe conocer a priori una buena estimación de la profundidad del nodo objetivo en el árbol de búsqueda.

A.7.3 Algoritmo A_ϵ^*

Este algoritmo [88],[89] pretende evitar la búsqueda exhaustiva entre nodos que tengan un valor de la función de evaluación similar, intentando ponderar para estos casos el esfuerzo de búsqueda necesario para llegar al óptimo, de forma que entre los nodos con una función heurística parecida se expande aquel que se estime que requiere un esfuerzo de búsqueda menor.

Para implementar este algoritmo se utiliza, además de ABIERTOS, otro conjunto de nodos que se denomina FOCAL, donde se almacenan todos los nodos cuyo coste de la función objetivo respecto al nodo con menor coste no es mayor que $1+\epsilon$

$$FOCAL=\{N : f(N)\leq(1+\epsilon)\min_{N\in ABIERTOS} f(N)\} \quad (A.4)$$

El algoritmo A_ϵ^* selecciona el siguiente nodo a expandir del conjunto FOCAL en lugar de ABIERTOS. Se define una nueva función heurística distinta, que se denominará $h_F(N)$, que deberá estimar el esfuerzo computacional requerido para encontrar una solución a partir de N. Este función heurística tiene un significado totalmente distinto al de la función h; así, mientras esta última debe estimar la calidad de la solución a partir de N, h_F estimará la dificultad de encontrar la solución.

En ciertas ocasiones es frecuente utilizar $h_F=h$. Aún en este caso, nótese la diferencia respecto al algoritmo A^* tradicional, mientras éste siempre expande atendiendo a la función f, el algoritmo A_ϵ^* expande el nodo con mejor valor para la función h de entre aquellos que tengan los mejores valores de f.

Los autores demuestran que el algoritmo A_ϵ^* es ϵ-admisible.

A.8 Búsqueda Bidireccional

En aquellos problemas donde el conjunto de transformaciones T que permiten pasar de un estado a otro se puede aplicar tanto hacia adelante como hacia atrás, puede ser interesante la realización de una búsqueda bidireccional. Esta consiste en realizar simultaneamente una búsqueda desde el nodo inicial hacia los nodos finales y desde el conjunto de nodos finales hacia el inicial. El proceso de búsqueda terminará cuando estos árboles de búsqueda se encuentren.

La motivación de este método radica en que frecuentemente el número de nodos crece exponencialmente con la profundidad del árbol, por lo que el número de nodos expandido es menor si se consideran dos árboles de profundidad $P/2$ en lugar de considerar un sólo árbol de profundidad P. Debido a este hecho, este método presenta muy buen comportamiento cuando la búsqueda se realiza en anchura.

Esta situación se complica cuando se trata con búsquedas heurísticas. Pohl [92] propuso unas funciones de evaluación similares a las del algoritmo A^*, distintas para la búsqueda hacia adelante y hacia atrás. Para la búsqueda hacia adelante:

$$f_s(N)=g_s(N)+h_s(N) \quad (A.5)$$

donde $g_s(N)$ es el coste del camino más corto encontrado desde el nodo origen hasta el nodo N y $h_s(N)$ es la estimación heurística del coste del camino desde N hasta el nodo final. Del mismo modo para la búsqueda hacia atrás se define:

$$f_t(N)=g_t(N)+h_t(N) \tag{A.6}$$

donde $g_t(N)$ es el coste del camino más corto encontrado desde el nodo objetivo hasta el nodo N y $h_s(N)$ es la estimación heurística del coste del camino desde N hasta el nodo inicial.

La búsqueda planteada de esta manera puede obtener, en algunos casos, peores resultados que la búsqueda unidireccional. Al realizar una búsqueda heurística se pretende que los árboles de búsqueda sean más estrechos, con lo que puede ocurrir que ambos árboles de búsqueda no se encuentren en un punto intermedio sino que se sobrepasen sin unirse, como se puede observar en la Figura A.27. De este modo, el número de nodos expandidos es del orden del doble de los expandidos en una búsqueda unidireccional.

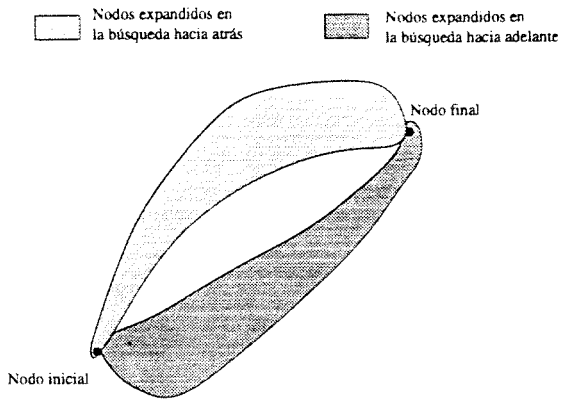

Figura A.2 Caso desfavorable en la búsqueda bidireccional

Para este problema se han propuesto varias suluciones. Entre ellas Champeaux [14] propone una nueva definición para las funciones heurísticas. h_s la define como una estimación heurística de la distancia al nodo objetivo pasando por algún nodo de ABIERTOS de la búsqueda hacia atrás. También h_t se define de un modo similar. El

principal inconveniente del método es el excesivo esfuerzo de cálculo que supone la obtención de la función heurística.

Con esta solución se encuentran mejores resultados en cuanto al número de nodos expandidos, sin embargo, presenta el inconveniente del mayor tiempo de cálculo de las funciones heurísticas h.

Apéndice B

Planificación de caminos para un robot mediante búsqueda heurística

B.1 Introducción

En este apéndice se describe el método de planificación heurística implementado para la obtención de los caminos libres de obstáculos fijos para cada uno de los robots que componen el sistema. El método fue descrito por Kondo [60] y utiliza un algoritmo basado en la descomposición del espacio de las configuraciones en celdas. La búsqueda se realiza mediante un algoritmo A^*, considerando simultáneamente varios árboles de búsqueda con funciones de evaluación distintas, de modo que la expansión de nodos depende de lo prometedora que sea cada una de las estrategias. También se utiliza búsqueda bidireccional. Este método permite la planificación del camino para cualquier tipo de móvil, aunque en esta tesis se utiliza para el caso de manipuladores.

El algoritmo ha sido implementado, realizándose diversas pruebas y estudios sobre la eficiencia del método [29].

B.2 Espacio de búsqueda

El problema que se plantea consiste en encontrar un camino libre de obstáculos para el robot que permita llevar al robot desde una configuración inicial, definida por sus articulaciones, hasta una configuración final. El espacio de búsqueda que se utiliza es el espacio de las configuraciones. Lógicamente, la dimensión de este espacio vendrá dada por el número de articulaciones del robot.

La idea fundamental es discretizar este espacio de búsqueda en celdas del mismo tamaño. De esta forma el espacio de las configuraciones estará formado por celdas regulares que podrán ser de dos tipos: Celdas Libres de Colisión, si para cualquier punto de la celda no exista colisión entre el manipulador y los obstáculos fijos, y Celdas Obstáculo, cuando existe algún punto de la celda que se corresponde con una configuración del manipulador, que colisione con algún obstáculo fijo. De esta forma el problema se reduce a encontrar una secuencia de celdas consecutivas que conecten la celda donde se encuentre el estado inicial (celda inicial) con aquella donde se encuentre el estado de llegada (celda final).

Este método puede encuadrarse dentro de los métodos de planificación locales, por lo que no se necesita calcular previamente el espacio de las configuraciones completo, sino que se va obteniendo solamente el estado de las celdas (libre u obstáculo) que el algoritmo necesita. De esta forma aparece un tercer tipo de celdas, aquellas cuyo estado es desconocido por no haber sido calculado. Debido al esfuerzo de cálculo que supone un algoritmo de chequeo de colisiones, se pretende que el número de celdas para el que se necesita calcular colisiones sea lo más pequeño posible.

B.3 Función de evaluación

Como se ha comentado anteriormente, el método de búsqueda está basado en el algoritmo A*. Los parámetros que utiliza para el algoritmo son los siguientes:

Los estado vienen dados por las celdas, es decir por las posiciones de las articulaciones del robot. Los operadores que se consideran son los movimientos en las direcciones de los ejes coordenados de longitud una celda. Para calcular los sucesores se toma una articulación, y dejando fijas el resto se mueve dicha articulación una cantidad fija, dependiente del tamaño de la celda, en cada una de las direcciones. Por tanto, una celda tendrá un número de sucesores igual al doble del número de grados de libertad del robot.

Como función de evaluación se utiliza:

$$f(C)=g(C)+h(C) \tag{B.1}$$

donde, como se ve en el apéndice A, C es una celda y $g(C)$ es el coste expresado en número de celdas del camino entre la celda inicial y la celda C. La estimación heurística del coste $h(C)$ la define el autor como:

$$h(C)=A\sqrt{\sum_{i=1}^{NGDL}\left(a(i)\times(c(i)-c_f(i))^2\right)} \tag{B.2}$$

donde *NGDL* es el número de grados de libertad del manipulador, $c(i)$ y $c_f(i)$ son los valores de la posición del i-ésimo grado de libertad (articulación) en la celda C y la celda final, respectivamente, $a(i)$ son unos coeficientes que ponderan los distintos grados de libertad y A es un coeficiente de ponderación general.

La idea de los coeficientes $a(i)$ es establecer distintas estrategias de búsqueda, de forma que si un grado de libertad tiene un coeficiente mayor que los correspondientes a otros, la búsqueda se realizará preferentemente mediante movimientos en esa dirección.

En relación a la función h, el autor no considera ninguna limitación a los distintos parámetros que pueden aparecer en dicha función heurística. Sin embargo, dicha función puede ser no admisible para valores de los coeficientes suficientemente altos, con los inconvenientes que ello conlleva, entre otros el no garantizar que se encuentre el óptimo. Supóngase que A es 1. Esto no supone ninguna restricción, ya que el efecto de este parámetro puede ser conseguido aumentando adecuadamente los coeficientes $a(i)$. Para garantizar que la función sea admisible es fácil comprobar que se tiene que cumplir que $a(i)\leq 1$, $\forall i$.

B.4 Búsqueda con múltiples estrategias

Dependiendo del espacio de las configuraciones y de las posiciones inicial y final del problema, la elección de distintos valores de los coeficientes de la función heurística puede llevar a importantes variaciones en la eficiencia del método. Sin embargo no se conoce ningún método que haga posible conocer a priori qué valores de los coeficientes son los más adecuados para un determinado caso. La alternativa que se propone consiste en ejecutar simultaneamente varias estrategias distintas, evaluando la eficiencia de cada una de ellas, de modo que mientras mayor sea la eficiencia estimada de una estrategia mayor será el número de nodos que se expanden de ella.

Para implementar el algoritmo se eligen una serie de estrategias para ser evaluadas. El algoritmo se divide en varias etapas, de forma que en cada etapa se expande un determinado número de celdas de cada una de las estrategias. El número de celdas expandidas de cada estrategia dependerá de la medida que se disponga de la eficiencia de

la estrategia. El procedimiento terminará cuando alguna de las estrategias llegue al estado final.

Sea S el número de estrategias selecionadas y $E_t(j)$ el número de celdas expandidas de la estrategia t en la etapa j. Inicialmente, no se conoce la eficiencia de ninguna de las estrategias, por lo que en la primera etapa se expandirá el mismo número de celdas para todas las estrategias.

$$E_t(1)=E_{init} \qquad 1\leq t\leq S \tag{B.3}$$

donde E_{init} es una constante predefinida. Para evaluar la eficiencia de una estrategia t en la etapa j se utilizará una función heurística que se denominará $P_t(j)$. Para obtener este valor se calcula la eficiencia de cada una de las celdas expandidas. Esta función se denomina $p_t(C)$ y está definida como:

$$p_t(C)=\frac{q_t(C)}{r_t(C)} \tag{B.4}$$

donde C es la celda expandida y $q_t(C)$ es una función, siempre positiva, que representa el estado de la estrategia, de forma que deberá tener valores mayores a medida que el proceso de búsqueda se vaya acercando al estado final. $r_t(C)$ es una función también positiva con la que se quiere evaluar el esfuerzo de cálculo de la estrategia t hasta el momento de la expansión de la celda C. De esta forma, las mayores estimaciones de eficiencia se darán para celdas que se encuentran en un estado avanzado en la búsqueda, en una estrategia con un esfuerzo de cálculo pequeño.

Para definir la eficiencia de una estrategia se utilizan los valores de las eficiencias de las Q últimas celdas expandidas para esa trayectoria, donde Q es una constante arbitraria. La expresión propuesta por el autor para el cálculo de esta eficiencia es:

$$P_t(j)=\frac{\sum_{\text{últimas } Q \text{ celdas}} p_t(C)}{Q} \tag{B.5}$$

A partir de este valor se obtiene el número de celdas a expandir en cada una de las estrategias durante la etapa siguiente mediante la expresión:

$$E_t(j)=E_{init}\frac{P_t(j-1)}{\max\limits_{1\le r\le S}(P_r(j-1))} \tag{B.6}$$

De esta forma, las estrategias ineficientes tendrán un valor bajo de $P_t(j\text{-}1)$, con lo que se expandirán pocos nodos de ellas.

Las funciones necesarias para la estimación de la eficiencia de una celda son implementadas por el autor de la siguiente forma:

$$q_t(C)=D_t(C)^{NGDL} \tag{B.7}$$

donde D_t se define como la longitud del camino desde la celda inicial hasta la celda C, y $NGDL$ es la dimensión del espacio de las configuraciones. Por otro lado $r_t(C)$ se define como el número total de celdas que han sido expandidas por la estrategia t. El exponente $NGDL$ en la definición de la función q_t lo utiliza el autor para compensar la diferencia de dimensión entre estas dos cantidades. En efecto, D_t puede ser considerado de dimensión unitaria, mientras que r_t representa un volumen $NGDL$-dimensional.

En este proceso es normal que las distintas estrategias puedan expandir la misma celda; sin embargo, nótese que, en este caso, el aspecto más costoso de esta expansión, la detección de colisiones, se realiza una sóla vez. Evidentemente la eficiencia del método es menor que la correspondiente a la ejecución únicamente de la mejor estrategia. Sin embargo, ya que éste es desconocido a priori, normalmente no se seleccionará, escogiendo una arbitraria, que puede llevar a la expansión de un número considerable de nodos, y consiguientemente a numerosas llamadas a la rutina de detección de colisiones.

Sin embargo, cuando realmente se ha comprobado en esta tesis la eficiencia de este método es cuando se incorpora búsqueda bidireccional, ya que en la mayoría de los casos tiene más importancia la elección adecuada de la dirección de búsqueda que la elección entre varias estrategias que buscan en la misma dirección.

B.5 Búsqueda bidireccional

Un aspecto muy importante en la eficiencia de la búsqueda es la dirección en la que se realiza, es decir, si se realiza la búsqueda desde el estado inicial al final o a la inversa. En la Figura B.1 se muestra cómo ante una determinada situación es preferible una dirección de búsqueda frente a la opuesta.

Para la realización de esta búsqueda bidireccional se consideran dos tipos de estrategias, hacia adelante y hacia atrás, y un número determinado de estrategias de cada uno de estos tipos. Sea S_f el número de estrategias hacia adelante y S_b el número de estrategias hacia atrás. La idea consiste en realizar en cada etapa una estimación a nivel global de la eficiencia de las estrategias hacia adelante frente a las estrategias hacia atrás, y seleccionar el tipo de estrategia más eficiente. Una vez realizado esto, de cada estrategia del tipo seleccionado se expanden un número de nodos de acuerdo con su eficiencia.

Inicialmente se expanden ambos tipos de estrategias y el mismo número de nodos de cada estrategia ya que no existen datos sobre la eficiencia. A partir de la segunda etapa se selecciona en primer lugar uno de los dos tipos de estrategias. Para ello es necesario estimar la eficiencia en conjunto de las estrategias hacia delante por un lado, y hacia atrás por otro lado. Para evaluar esta eficiencia se proponen las siguientes expresiones:

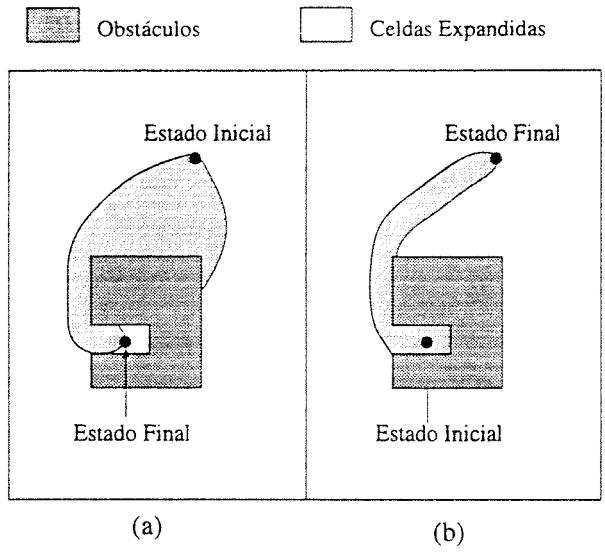

Figura B.1 Celdas expandidas al realizar la búsqueda hacia adelante (a) y hacia atrás (b)

$$R_f=\frac{G_f}{H_f} \qquad R_b=\frac{G_b}{H_b} \tag{B.8}$$

donde R_f y R_b son estimaciones de la eficiencia del conjunto de estrategias hacia adelente y hacia atrás, respectivamente. G_f y G_b son el número total de celdas expandidas por cada uno de los tipos de búsqueda y H_f y H_g son el número total de celdas para las que se realizó comprobación de colisiones. En cada etapa sólo se expanden celdas de las estrategias correspondientes al tipo de búsqueda más eficiente, es decir al que tenga un valor más pequeño de la función R (R_f o R_b).

Una vez seleccionado el tipo de búsqueda, el número de celdas expandidas de cada uno de los nodos se determina de un modo similar al utilizado en la búsqueda unidireccional. Sean $E_{f_t}(j)$ el número de celdas a expandir en la estrategia hacia delante t durante la etapa j. Este valor está determinado por:

$$E_{f_t}(1)=E_{init}$$

$$E_{f_t}(j)=E_{init}\frac{P_{f_t}(j-1)}{\max\limits_{1\le r\le S_f}\{P_{f_r}(j-1)\}} \tag{B.9}$$

Del mismo modo se puede determinar el número de celdas a expandir para búsqueda hacia atrás:

$$E_{b_t}(1)=E_{init}$$

$$E_{b_t}(j)=E_{init}\frac{P_{b_t}(j-1)}{\max\limits_{1\le r\le S_b}\{P_{b_r}(j-1)\}} \tag{B.10}$$

En el caso de búsqueda bidireccional, el proceso de búsqueda terminará cuando se conecte alguna estrategia hacia adelante con alguna hacia atrás, es decir, cuando al expandir

un nodo de una estrategia, aparezca un nodo que se encuentre en el conjunto ABIERTOS de alguna de las estrategias en sentido contrario.

Como función heurística en la búsqueda bidireccional el autor propone utilizar la distancia desde el nodo actual al centro de gravedad de los últimos nodos expandidos en la direción contraria. De este modo los nodos expandidos de ambos tipos de búsqueda tienden a unirse.

B.6 Representación del espacio libre

El trabajar con espacios de búsqueda de dimensión alta hace necesario utilizar métodos eficientes para el almacenamiento de la información asociada a cada celda. Para esto, el autor representa el espacio de las configuraciones de forma jerárquica mediante su división recursiva en subespacios. El método es similar a la representación de espacios bidimensionales mediante *árboles cuaternarios* (*quadtree*).

El espacio de las configuraciones de dimensión N es dividido en 2^N subespacios, dividiendo cada eje en dos secciones. Cada uno de estos subespacios es de nuevo dividido en el mismo número de subespacios. El proceso se repite recursivamente hasta que se alcancen celdas con la resolución requerida. El espacio libre queda representado por tanto mediante un árbol, de forma que cada nodo tiene 2^N sucesores.

Con el fin de utilizar la memoria de forma eficiente, este árbol se va construyendo a medida que avanza el proceso de búsqueda. Es decir, sólo se realiza la asignación de memoria a aquellas celdas que aparecen durante el proceso de búsqueda.

Apéndice C

Modelo de las trayectorias de un manipulador

C.1 Introducción

En capítulos anteriores se mostró la necesidad de utilizar un modelo de las trayectorias descritas por un manipulador cuando se utiliza el generador de trayectorias implementado en los controladores de robots industriales. Este modelo es utilizado para la obtención del tiempo total necesario para la realización del movimiento coordinado de varios manipuladores, con la inclusión de *puntos de sincronización* para evitar colisiones entre ellos. El modelo utilizado debe ser, por una parte, lo suficientemente exacto para obtener resultados representativos, y por otra parte lo suficientemente simple para que el tiempo de ejecución de los algoritmos se mantenga en cotas aceptables, debido a que al ser un proceso de optimización, este modelo debe ser utilizado intensivamente.

El modelo deberá obtener la trayectoria para un camino formado por tramos rectilíneos en el *espacio de las articulaciones*. Por tanto, un camino puede se descrito por una secuencia de puntos, donde cada punto es un vector cuyas componentes son los valores de cada una de las articulaciones del robot.

$$C=\{P_i\} \quad \text{con} \quad P_i=(\theta_i^1,...,\theta_i^{NART}) \quad \text{y} \quad 1\leq i\leq N \tag{C.1}$$

donde C es el camino, N el número de puntos que forman dicho camino, y $NART$ el número de articulaciones del robot. En general, para este tipo de movimientos, los robots industriales incluyen dos modos distintos. En primer lugar, el *movimiento punto a punto* donde el robot se detiene en cada uno de los puntos intermedios, con lo que el camino seguido está formado por los tramos rectilíneos programados. El segundo tipo de movimiento es el *modo continuo*, donde el robot no se detiene en los puntos intermedios,

sino que realiza una transición suave entre tramos, de forma que, en estos puntos, la trayectoria real se separa de la deseada. El modelo deberá posibilitar la operación en cada uno de estos modos. En cualquier caso, el estudio se centrará en el modo continuo, ya que el movimiento punto a punto puede considerarse como un caso particular del anterior.

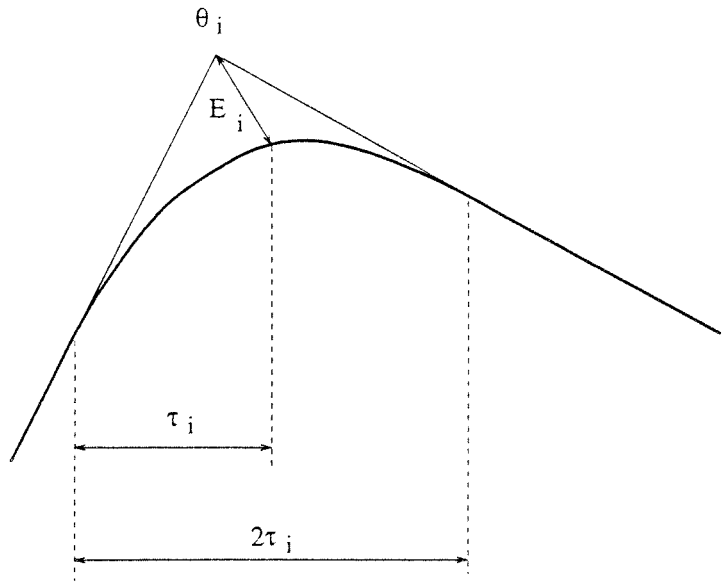

Figura C.1 Transición entre dos tramos.

El modelo utilizado está basado en los trabajos teóricos de Paul [87], Luh [74],[75] y Tondu [119]. En el modelo del movimiento de las articulaciones se realizarán las siguientes suposiciones:

- El movimiento en cada tramo se realiza con velocidad constante, con transiciones suaves entre tramos, considerando un perfil de aceleraciones continuo. De esta forma el camino resultante no pasará por los puntos intermedios. Para reducir esta desviación, se considerará que la aceleración máxima será alcanzada en cada una de estas transiciones. (Figura C.1).

- Tal como se ve en la Figura C.1, se supone que la transición se realiza de forma simétrica, de forma que la aceleración máxima se produce en el punto medio de la transición.

- En cada tramo, el movimiento de cada una de las articulaciones está sincronizado con el resto de las articulaciones, es decir, la velocidad de cada articulación está

ajustada de forma que todas las articulaciones terminen su movimiento simultáneamente.

C.2 Modelo de una articulación

En este apartado se describe el modelo utilizado para la trayectoria de cada una de las articulaciones del robot. En próximos apartados se indicará como se ha realizado la coordinación entre cada una de las articulaciones en cada tramo.

Como se indicó anteriormente, la trayectoria de una articulación se describe mediante un movimiento a velocidad constante, salvo en las proximidades de los puntos intermedios, donde existirá una zona de transición suave de velocidades. (Figura C.2). La nomenclatura usada será la siguiente [119]:

θ_i^j Valor de la articulación j en el punto intermedio i

T_i^j Tiempo necesario para realizar el movimiento entre θ_{i-1}^j y θ_i^j

τ_i^j Mitad del tiempo necesario para realizar la transición en el punto θ_i^j

$\dot{\theta}_i^j$ Velocidad constante en el segmento anterior al punto i

$\left|\ddot{\theta}_i^j\right|$ Aceleración máxima en la transición i

En las etapas de velocidad constante, lógicamente el valor de la articulación en función del tiempo $\theta^j(t)$ será lineal. Más complejo es el período de transición. Para ello vamos a suponer que $\theta^j(t)$ es polinomial. Teniendo en cuenta las restricciones impuestas de continuidad en posición, velocidad y aceleración, se hace necesario utilizar un polinomio de quinto grado. Sin embargo, debido a que se considerará simetría en las transiciones, será suficiente con un polinomio de cuarto grado. Con estas condiciones, las ecuaciones para cada una da estas zonas serán:

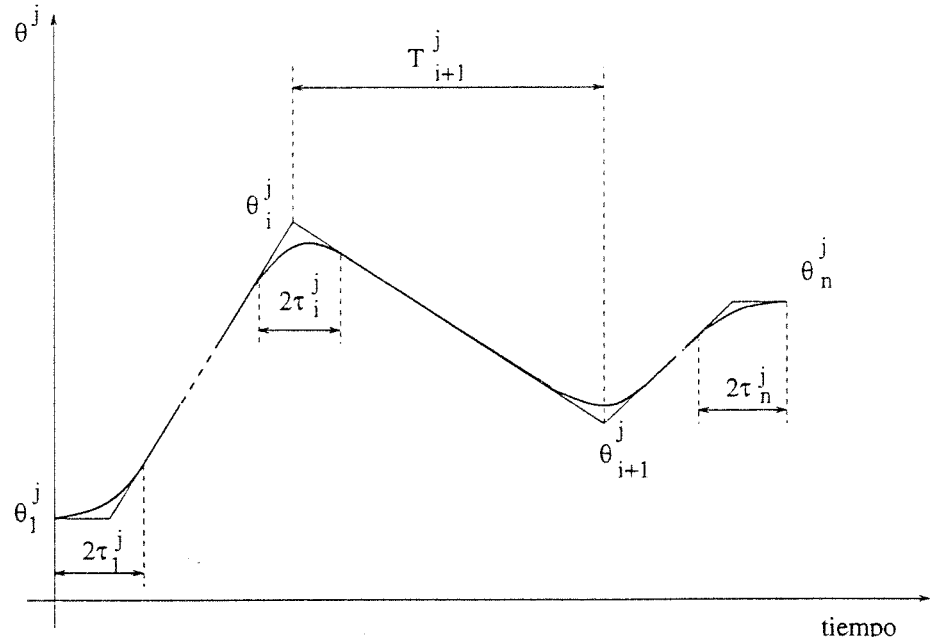

Figura C.2 Modelo de la trayectoria de una articulación.

Transición:

$$\theta^j(t)=-\frac{1}{4}\left(\frac{\dot{\theta}^j_{i+1}-\dot{\theta}^j_i}{(\tau^j_i)^3}\right)\left(\frac{(t-t^j_i)^4}{4}-\tau^j_i(t-t^j_i)^3\right)+\dot{\theta}^j_i(t-t^j_i-\tau^j_i)+\theta^j_i \qquad \text{(C.2)}$$

$$\text{para} \quad t^j_i \le t \le t^j_i+2\tau^j_i \quad y \quad 1 \le i \le N$$

Velocidad constante:

$$\theta^j(t)=\dot{\theta}^j_{i+1}(t-t^j_i-\tau^j_i)+\theta^j_i \qquad \text{(C.3)}$$

$$\text{para} \quad t^j_i+2\tau^j_i \le t \le t^j_i+\tau^j_i+T^j_i-\tau^j_{i+1} \quad \text{y} \quad 1 \le i \le N-1$$

Considerando que el manipulador parte de la posición de reposo y termina en esta misma condición, de deberá cumplir que $\dot{\theta}_1^j=\dot{\theta}_{N+1}^j=0$. En las ecuaciones anteriores, t_i^j se define como el tiempo invertido en realizar el movimiento entre la posición inicial y la posición en que se inicia la transición en i, es decir para el primer punto $t_1^j=0$ y para los demás:

$$t_i^j=\left(\sum_{k=2}^{i} T_k^j\right)+\tau_1^j-\tau_i^j \quad \text{con} \quad 2\le i\le N \tag{C.4}$$

Finalmente, T_i^j se define como el tiempo necesario en recorrer el tramo entre los puntos θ_{i-1}^j y θ_i^j suponiendo velocidad constante en todo el tramo, es decir:

$$T_i^j=\frac{\left|\theta_i^j-\theta_{i-1}^j\right|}{\left|\dot{\theta}_i^j\right|} \quad \text{con} \quad 2\le i\le N \tag{C.5}$$

Teniendo en cuenta que se ha supuesto que la aceleración máxima $\left|\ddot{\theta}_i^j\right|$ se alcanza en el punto medio de la transición, se puede obtener el tiempo necesario para realizar la transición:

$$\tau_i^j=\frac{3}{4}\frac{\left|\dot{\theta}_{i+1}^j-\dot{\theta}_i^j\right|}{\left|\ddot{\theta}_i^j\right|} \quad \text{con} \quad 1\le i\le N \tag{C.6}$$

Un aspecto importante de esta aproximación es el error cometido al no seguir en las transiciones el camino predefinido. Este error se puede evaluar mediante la siguiente expresión [75]:

$$\varepsilon_i=\left|\theta^j(t=t_i+\tau_i^j)-\theta_i^j\right| \tag{C.7}$$

sustituyendo en (C.2), y teniendo en cuenta (C.6) se deduce:

$$\varepsilon_i = \frac{9}{64} \frac{\left|\dot{\theta}_{i+1}^j - \dot{\theta}_i^j\right|^2}{\left|\ddot{\theta}_i^j\right|} \quad \text{con} \quad 1 \le i \le N \quad \text{y} \quad 1 \le j \le NART \tag{C.8}$$

C.3 Sincronización de las articulaciones de un robot

Después de obtener las ecuaciones asociadas al movimiento de cada una de las articulaciones, es necesario considerar que en cada uno de los tramos que definen el camino, el movimiento de éstas debe estar sincronizado, es decir el movimiento deberá comenzar y terminar simultáneamente. Para conseguir esto, las velocidades y aceleraciones obtenidas para cada articulación deben ser modificadas, en función de la que necesite un mayor tiempo para realizar su movimiento.

Para ello habrá que conseguir que los tiempos invertidos en cada tramo por cada articulación sean iguales al correspondiente a la articulación que necesite invertir un tiempo mayor. Por tanto, el tiempo en que se ejecuta cada tramo será:

$$T_i = \max_{1 \le j \le NART} (T_i^j) \quad \text{con} \quad 2 \le i \le N \tag{C.9}$$

Esto obligará a ajustar las velocidades de las distintas articulaciones a un nuevo valor que se denominará $(\dot{\theta}_i^j)_s$:

$$\left|(\dot{\theta}_i^j)_s\right| = \left|\dot{\theta}_i^j\right| (T_i^j / T_i) \quad \text{con} \quad 2 \le i \le N \quad \text{y} \quad 1 \le j \le NART \tag{C.10}$$

Según (C.6), debido a estos cambios en las velocidades, será necesario modificar los valores de τ_i^j, dando lugar a:

$$(\tau_i^j)' = \frac{3}{4} \frac{\left|(\dot{\theta}_{i+1}^j)_s - (\dot{\theta}_i^j)_s\right|}{\left|\ddot{\theta}_i^j\right|} \quad \text{con} \quad 1 \le i \le N \quad \text{y} \quad 1 \le j \le NART \tag{C.11}$$

Finalmente, habrá que ajustar el tiempo de las transiciones en los puntos iniciales y finales, para que sean iguales para todas las articulaciones, es decir:

$$\tau_i = \max_{1 \le j \le NART} ((\tau_i^j)') \quad \text{con} \quad i=1 \quad \text{e} \quad i=N \tag{C.12}$$

con lo que finalmente será necesario ajustar las aceleraciones máximas en estos dos puntos de la siguiente manera:

$$\left|(\ddot{\theta}_i^j)_s\right| = \left|\ddot{\theta}_i^j\right| ((\tau_i^j)'/\tau_i) \quad \text{con} \quad i=1,\ i=N \quad \text{y} \quad 1 \le j \le NART \tag{C.13}$$

Con este modelo, la trayectoria queda especificada con las ecuaciones anteriores en función de dos parámetros: la velocidad deseada en cada tramo y la aceleración máxima en cada articulación.

C.4 Restricciones del modelo

En el modelo de trayectoria propuesto se supone que dos transiciones consecutivas no se solapan, es decir:

$$T_i^j \ge \tau_{i-1}^j + \tau_i^j \quad \text{con} \quad 2 \le i \le N \quad \text{y} \quad 1 \le i \le NART \tag{C.14}$$

Sin embargo, en la práctica es poco frecuente que se solapen debido principalmente al valor elevado de las aceleraciones, que hace que las transiciones sean cortas. En cualquier caso, esta restricción puede ser introducida, considerando que el robot se detendrá al principio del tramo conflictivo.

Más importancia tiene considerar el tiempo invertido por el generador de trayectorias para los cálculos, lo cual hace necesario considerar un valor mínimo para T_i. Sea T_C esta constante mínima. Normalmente, cuando en un tramo se viola esta restricción, el controlador reduce la velocidad del manipulador, de forma que el tiempo invertido en

el tramo será T_C (así ocurre para el robot PUMA-560 con el lenguaje VAL). Esta restricción en fácil de añadir en el modelo, sustituyendo los valores de T_i de (C.9) por T_C cuando sean menores que éste.

Bibliografía

[1] T. Bäck, G. Rudolph y H.P. Schwefel. *Evolutionary Programming and Evolution Strategies: Similarities.* Fogel y Atmar, editores. Proceedings of the 2° Annual Conference on Evolutionary Programming, San Diego CA, 1993.

[2] T. Bäck, F. Hoffmeister y H.P. Schwefel. *A Survey of Evolution Strategies.* Proceedings of the 4th International Conference on Genetic Algorithms and their Applications, páginas 2-9. San Mateo, CA,1991

[3] A. Barr y E. Feigenbaum. *The Handbook of Artificial Intelligence.* Pitman. 1981.

[4] J. Barraquand, *Automatic motion planning for complex articulated bodies*, Research Report n° 14, Digital Paris Research Laboratory. Junio 1991.

[5] J. Barraquand, B. Langlois y J.C. Latombe, *Numerical potential field techniques for robot path planning*, IEEE Transations on Systems, man and Cybernetics, Vol. 22, n° 2, páginas 224-241. Marzo-Abril 1992.

[6] R.A. Basta, R. Mehrotra y M.R. Varanasi, *Detecting and avoiding collisions between two robots arms in a common workspace,* Robot Control Theory and Applications, páginas 185-192, IEE Press.

[7] P. Bessière, J.M. Ahuactzin, E.G. Talbi y E. Mazer, *The "Ariadne's clew" algorithm: global planning with local methods*, IEEE/RSJ Conference on Intelligence Robot Systems. 1993.

[8] Z. Bien y J. Lee, *A minimum-time trajectory planning method for two robots.* IEEE Transations on Robotics and Automation, Vol. 8, n° 3, páginas 414-418. Junio 1992.

[9] J.E. Bobrow, *Optimal robot path planning using the minimum-time criterion.* IEEE Journal of Robotics and Automation, Vol. 4, n° 4, páginas 443-450. Agosto 1988.

[10] S. Bonner y R.B. Kelley, *A novel representation for planning 3-D collision-free paths.* IEEE Transations on Systems, Man, and Cybernetics, vol. 20, n°6, páginas 1337-1351. Noviembre-Diciembre 1990.

[11] R.A. Brooks, *Solving the find-path problem by good representation of the free space.* IEEE Transations on Systems, Man and Cybernetics. Vol. SMC-13, páginas 190-197. Marzo-Abril 1983.

[12] S.J. Buckley, *Fast motion planning for multiple moving robots*, Proceedings IEEE Int. Conf. on Robotics and Automation, páginas 322-326. Scottdale AZ. 1989.

[13] A.S. Çela y Y. Hamam, *Optimal Motiion Planning of a Multiple-Robot System Based on Descomposotion Coordination.* IEEE Transations on Robotics and Automation, Vol. 8, n° 5, páginas 585-596. Octubre 1992.

[14] D. de Champeaux y L. Sint. *An improved bi-directional heuristic search algorithm.* Journal of the ACM 24(2) pp. 177-191. 1977.

[15] C. Chang, M.J. Chung y R.H. Lee, *Collision avoidance of two general robots manipulators by minimum delay time.* IEEE Transations on System, Man and Cybernetics. Vol. 24, n° 3, páginas 517-522. Marzo 1994.

[16] Y.H. Chang, T.T. Lee y C.H. Liu, *On-line approximate cartesian path trajectory planning for robotic manipulators.* IEEE Transations on Systems, Man and Cybernetics. Vol. 22, n° 3, páginas 542-547. Mayo-Junio 1992.

[17] Y.P. Chien y Q. Xue, *Path planning for two planar robots moving in unknown enviroment.* IEEE Transations on Systems, Man and Cybernetics, Vol. 22, n° 2, páginas 307-317. Marzo-Abril 1992.

[18] Y.P. Chien, Q. Xue y Y. Chen, *Configuration space model of tightly coordinated two robot manipulators operating in 3-dimensional workspace*, IEEE Transations on Systems, Man and Cybernetics, Vol. 25, n° 4, páginas 695-704. Abril 1995.

[19] G. Conte y R. Zulli, *Hierarchical path planning in a multi-robot enviroment with a simple navigation function.* IEEE Transations on Systems, Man and Cybernetics, Vol 25, n° 4. Abril 1995.

[20] Y. Davidor, *Robot programming with a genetic algorithm,* Proceedings of the 1990 IEEE International Conference on Computers Systems and Software Engineering, páginas 186-191. 1990.

[21] L. Davis. *The Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York,1991.

[22] K.A. De Jong, *An analysis of the behaviour of a class of genetic adaptive systems.* Doctoral dissertation, University of Michigan, 1975.

[23] N. Duffy, D. Allan y J.T. Herd, *Real-time collision avoidance system for multiple robots operating in shared work-space.* IEE Proceedings, Vol. 136, Pt. E, nº 6, páginas 478-484. Noviembre 1989.

[24] A. Durán Toro. *Proyecto REX: Extensión de Robótica para las GL.* Proyecto Fin de Carrera para la obtención del título de Licenciado en Informática. Dpto. Ingeniería de Sistemas y Automática. Universidad de Sevilla. 1994.

[25] Erdmann y Lozano-Pérez, *On multiple moving objects,* AI MEMO nº 883, Artificial Intelligence Laboratory, MIT, 1986.

[26] Eshed Robotec, *Manual de usuario del SCORBOT - ER V.* 1994

[27] Eshed Robotec, *ACL Lenguaje de Control Avanzado y ATS Software de terminal Avanzado. Guía de Referencia.* 3ª Edición. Noviembre 1994.

[28] H. Farreny y M. Ghalab. *Éléments d'intelligence artificielle.* Traitè des Nouvelles Technologies. Série Intelligence artificielle. Hermès. Paris 1990.

[29] E. Fernández, *Planifivcación del movimiento para manipuladores mediante búsqueda multiestratégica.* Proyecto Fin de Carrera para la obtención del título de Licenciado en Informática. Dpto. Ingeniería de Sistemas y Automática. Universidad de Sevilla. 1994.

[30] L.J. Fogel, A.J. Owens y M.J. Walsh. *Artificial Intelligence trhough Simulated Evolution.* John Wiley, New York, 1966.

[31] S. Fortune, G. Wilfong y C. Yap, *Coordinated motion of two robot arms,* Proceedings IEEE Int. Conf. on Robotics and Automation, páginas 1216-1223. San Francisco. 1986.

[32] E. Freund y H. Hoyer, *Pathfinding in multirobot systems: solution and applications.* Proceedings IEEE Int. Conf. on Robotics and Automation., páginas 103-111. San Francisco. 1986.

[33] E. Freund y H. Hoyer, *Real-time pathfinding in multirobot systems including obstacle avoidance*. The International Journal of Robotic Research, Vol. 7, n° 1, páginas 42-70. Febrero 1988.

[34] K.S. Fu, R.C. González y C.S.G. Lee, *Robótica: Control, detección, visión e inteligencia*. Ed. McGraw-Hill. 1988.

[35] M. Galicki, *Optimal planning of a collision-free trajectory of redundant manipulators*. The International Journal of Robotics Research. Vol. 11, n° 6, páginas 549-559. Diciembre 1992.

[36] M.A. Ganter y J.J. Uicker Jr. *Dynamic collision detection using swept solids*. Journal of Mechanism, Transmissions and Automation in Design. Vol. 8, n°4, páginas 549-555. Diciembre 1986.

[37] J. Gashnig, *Performance measurement and analysis of certain search algorithms*. Ph. D. Thesis, Dept. of C.S. Carnegie-Mellon. Univ. Pittsburgh, 1979.

[38] E.G. Gilbert y D.W. Johnson, *A fast procedure for computing the distance between complex objects in three-dimensional space*. IEEE Journal of Robotics and Automation, Vol. 4, n° 2, páginas 193-203. Abril 1988.

[39] E.G. Gilbert y C. Foo, *Computing the distance between general convex objects in three-dimensional space*. IEEE Transations on Robotics and Automation, Vol. 6, n° 1, páginas 53-61. Febrero 1990

[40] G. Gini, R. Massa y N. Negretti, *Towards Efficient path-planning for articulated robots*. Journal of Robotics Systems, Vol. 12, n° 2, páginas 93-104. 1995

[41] D.E. Goldberg. *Genetic Algorithm in Search, Optimization and Machine Learning*. Adisson-Wesley, New York, 1989.

[42] J. Gómez Ortega, *Navegación en robots móviles basada en técnicas de control predictivo neuronal*. Tesis Doctoral. Universidad de Sevilla. 1994.

[43] F. Gordillo. *Contribuciones al problema del control óptimo*. Tesis Doctoral. Universidad de Sevilla. 1994.

[44] S. Gootschilch, C. Ramos y D. Lyons, *Assembly and task planning: A taxonomy*. IEEE Robotics and Automation Magazine. Vol. 1, n° 3, páginas 4-12. Septiembre 1994.

[45] K.K. Gupka y Z. Guo, *Motion Planning with many degrees of freedom: sequential search with backtraking,* IEEE International Conference on Robotics and Automation. 1992.

[46] K.K. Gupka y X. Zhu, *Practical global motion planning for many degrees of freedom: A novel approach within sequential framework.* Journal of Robotics Systems. Vol 12, nº 2, páginas 105-117. 1995.

[47] P. Hart, N.J. Nilsson and B. Raphael. *A formal basis for the heuristic determination of minimum cost paths.* IEEE Transations on SSC. SSC-4. pp. 100-107. 1968.

[48] R.F. Hartl. *A global convergence proof for a class of genetic algorithm.* Technische Universität Wien. 1990.

[49] T. Hasegawa y H. Terasaki, *Collision Avoidance: Divide-and-conquer approach by space characterization and intermediate goals.* IEEE Transations on Systems, Man and Cybernetics, Vol. 18, nº 3, páginas 337-347. Mayo-Junio 1988.

[50] F. Herrera, M. Lozano y J.L. Verdegay. *Algoritmos Genéticos: Fundamentos, Extensiones y Aplicaciones.* Technical Report #DECSAI 94105. ETS de Ingeniería Informática. Universidad de Granada. Abril 1994.

[51] F. Hoffmeister y T. Bäck. *Genetic Algorithm and Evolution Strategies: Similarities and Differences.* Technical Report Nº SYS-1/92. System Analysys Research Group. University of Dortmund. Alemania. Febrero 1992.

[52] J.H. Holland. *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, 1975. (The MIT Press, Londres, 1992).

[53] T. Ibaraki, *Theorical comparison of search strategies in branch-and-bound alggorithms.* International Journal of Computer and Information Sciences. Vol. 5, nº 4, páginas 315-344. 1976.

[54] I.B.M. *CATIA Version 3. Reference Manual.* 1991.

[55] K. Kant y W. Zucker, *Toward efficient trajectory planning: The path-velocity decomposition.* International Journal on Robotics Research, vol. 5, nº 1, páginas 72-89. Otoño 1986.

[56] K. Kant y W. Zucker, *Planning collision-free trajectories in time-varying enviroments: a two level hierarchy.* Proceedings of IEEE International Conference on Robotics and Automation. páginas 1644-1649. 1988

[57] O. Khatib, *Real-time obstacle avoidance for manipulators and mobila robots,* The International Journal of Robotic Research, Vol. 5, n° 1, páginas 90-98. Primavera 1986.

[58] K. Kondo, *A simple motion planning algorithm using heuristic free space enumeration,* Proceedings IEEE International Workshop on Intelligence Robots Systems (IROS '88). Tokio 1988.

[59] K. Kondo y F. Kimura, *Collision avoidance using a free space enumeration method based on grid expansion.* Advanced Robotics, Vol. 3, n°3, páginas 159-175. 1989.

[60] K. Kondo. *Motion Planning with six degrees of freedom by multistrategic bidirectional heuristic free-space enumeration.* IEEE Trans. on Robotic and Automation. Vol 7, no 3, Junio 1991.

[61] J.R. Koza. *Concept formation and decision tree induction using the genetic programming paradigm.* En Branco Součec y The IRIS Group, editores. Dynamic, Genetic and Chaotic Programming, páginas 203-321. John Wiley and Son, New York, 1992.

[62] J.C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers.

[63] B.H. Lee y C.S.G. Lee, *Collision-free motion planning of two robots,* IEEE Transations on Systems, Man and Cybernetics, Vol. SMC-17, n° 1, páginas 21-32. Enero-Febrero 1987

[64] C.T. Lee y P.C.Y. Sheu, *A divide-and-conquer approach with heuristics of motion planning for a cartesian manipulator.* IEEE Transations on Systems, Man and Cybernetics, Vol. SMC-22, n° 5, páginas 929-944. Septiembre 1992.

[65] J. Lee, *A dynamic programming approach to near minimum-time trajectory planning for two robots.* IEEE Transations on Robotics and Automation. Vol. 11, n° 1, páginas 160-164. Febrero 1995.

[66] K.D. Lee, B.H. Lee y M.S. Ko, *A comparisson of interconnection methods for multi-robot systems.* Contro Engeneering Practice, Vol. 3, n° 3, páginas 337-346. 1995.

[67] C.F. Lin y W.H. Tsai, *Optimal assginment of robot task with precedence for multirobot coordination by disjunctive graphs and state-space search,* Journal of Robotics Systems, Vol 12, n° 4, páginas 219-236. 1995

[68] C.S. Lin y P.R. Chang, *Joint trajectories of mechanical manipulators for cartesian path approximation.* IEEE Transations on Systems, Man and Cybernetics. Vol. SMC-13, nº 6, Noviembre-Diciembre 1983.

[69] C.S. Lin, P.R. Chang y J.Y.S. Luh, *Formulation and optimization of cubic polynomial joint trajectories for industrial robots.* IEEE Transations on Automatic Control, Vol. AC-28, nº 12, Diciembre 1983.

[70] Y.H. Liu, S. Kuroda, T. Naniwa, H. Niborio y S. Arimoto, *A practical algorithm for planning collision-free coordinated motion of multiple mobile robots,* Proceedings IEEE Int. Conf. on Robotics ans Automation, páginas 1427-1432. Scottsdale, AZ. Mayo 1989.

[71] T. Lozano Pérez, *Automatic Planning of Manipulator Transfer Movements.* IEEE Transations on Systems, Man and Cybernetics, Vol. SMC-11, nº 10, páginas 681-698. Octubre 1981.

[72] T. Lozano-Pérez, *Spacial Planning: a configuration space approach.* IEEE Transations on Computers. Vol. C-32, nº 2, páginas 108-120. 1983

[73] T. Lozano-Pérez, *A simple motion-planning algorithm for general robot manipulators.* IEEE Journal of Robotics and Automation, Vol. RA-3, nº 3. Junio 1987.

[74] J.Y.S. Luh y C.S. Lin, *Optimum Path Planning for mechanical manipulators,* ASME Transations, 1981

[75] J.Y.S. Luh y M.W. Walker, *Minimum-time along the path for a mechanical arm.* Proceedins of the 1977 IEEE Conference on Decision and Control, páginas 755-759.

[76] J.Y.S. Luh y C.E. Campbell Jr., *Minimum distance collision-free path planning for industrial robots with a prismatic join,* IEEE Transations on Automatic Control, Vol. AC-29, nº 8. Agosto 1984.

[77] J.Y.S. Luh y C.C. Lin, *Approximate joint trajectories for control of industrial robots along cartesian paths.* IEEE Transations on Systems, Man and Cybernetics, Vol. SMC-14, nº 3, páginas 444-450. Mayo-Junio 1984.

[78] V.J. Lumelwsky, *Dymamic path planning for a planar articulated robot arm moving amidst unknown obstacle,* Automatica, Vol. 23, nº 5, páginas 551-570. 1987

[79] V.J. Lumelwsky, *Effect of kinematics on motion planning for planar robot arms moving amidst unknown obstacles.* IEEE Journal of Robotics and Automation, Vol. RA-3, nº 3, páginas 207-223. Junio 1987.

[80] A.A. Maciejewski y J.J. Fox, *Path Planning and the topology of configuration space*, IEEE Transations on Robotics and Automation, Vol. 9, nº 4, páginas 444-456. Agosto 1993.

[81] A. Martinelli. *On the complexity of admissible search algorithms.* Artificial Intelligence, 8, páginas 1-13. 1977.

[82] T. Nagata, K. Honda y Y. Teramoto, *Multirobot plan generation in a continuous domain: planning by use of plan graph and avoiding collisions among robots.* IEEE Journal of Robotics and Automation, Vol. 4, nº 1, páginas 2-13. Febrero 1988.

[83] N.J. Nilsson, *A mobile automaton: an application of artificial intelligence techniques.* Proceedings of the 1st International Joint Conference on Artoficial Intelligence, páginas 509-520. Washington D.C. 1969.

[84] N.J. Nilsson. *Principios de Inteligencia Artificial.* Díaz de Santos. 1987.

[85] P.A. O'Donnell y T. Lozano-Pérez, *Deadlock-free and collision-free coordination of two robot manipulators.* Proceedings IEEE International Conference on Robotics and Automation, páginas 484-489. 1989

[86] J. O'Rouke y N. Badler, *Decomposition of three-dimensional objects into spheres.* IEEE Transations on Pattern Analysis and Machine Intelligence, vol 1. 1979.

[87] R. Paul. *Robot Manipulators: Mathematics, Programming and Control*, MIT Press, Cambridge, MA, 1982.

[88] J. Pearl y J.H. Kim. *Studies in semi-admissible heuristics.* IEEE Trans. on Pattern Analysis and Machine Intelligence, PAMI-4(31) pp 250-62. 1982.

[89] J. Pearl. *Heuristics: Intelligent search strategies por computer problem solving.* Addison Wesley, 1984.

[90] H. Pedram, *A resource sharing approach to multiple robot motion planning.* Ph. D. Dissertation. Washington State University. Mayo 1992.

[91] F. Pfeiffer y R. Johanni, *A concept for manipulator trajectory planning*, IEEE Journal of Robotics and Automation, Vol. RA-3, nº 2, páginas 115-123. Abrol 1987.

[92] I. Pohl. *Bidirectional Search.* Machine Intelligence. 6. Ed. B. Meltzer y D. Michie, pp. 127-40. New York: American Elsevier. 1971

[93] I. Pohl. *The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computational issues in heuristic problem solving.* Proc. IJCAI 3. Stanford, pp-20-23. 1973.

[94] I. Pohl. *First results on the effect of error in heuristic search.* Machine Intelligence 5, ed. B. Meltzer y D. Michie, pp. 219-36. New York: American Elsevier. 1970

[95] F.P. Preparata y M.I. Shamos, *Computational geometry: An introducction.* Springer-Verlag. Nueva York. 1985.

[96] N.J. Radcliffe, *Forma Analysis and Random Respectful Recombination.* Proceedings of the 4th International Conference on Genetic Algorithms. páginas 222-229, San Mateo. 1991.

[97] I. Rechemberg. *Evolutionsstrategie: Optimierung Teshnicher Systeme nach Prinzipien der Biologischen Evolution.* Frommann-Holtzboog, Stugart 1973.

[98] E. Rich. *Inteligencia Artificial.* Gustavo Gili. 1988.

[99] M.A. Ridao y E. F. Camacho, *A heuristic algorithm for coordinated motion planning of two manipulators.* Proceedings of the European Robotics and Intelligent Systems Conference (EURISCON'94). Vol. 2, páginas 1089-1096. Málaga (Spain). Agosto 1994.

[100] M.A. Ridao, J. Riquelme, E.F. Camacho y M. Toro, *Coordinated motion planning of manipulators by evolution strategies.* Proceedings of the Tenth International Conference on Applications of Artificial Intelligence in Engineering (AIENG'95), páginas 245-252. Udine (Italia). Julio 1995.

[101] P.I. Ro, B.R. Lee, M.A. Seelhemmer, *Two-arm kinematic posture optimization for fixtureless assembly*, Journal of Robotic Systems, Vol. 12, n° 1, páginas 55-65. 1995.

[102] J.W. Roach y M.N. Boaz, *Coordinating the motion of robot arms in a common workspace*, IEEE Journal of Robotics and Automation, Vol. RA-3, n° 5, páginas 437-444. Octubre 1987.

[103] R. Schalkoff. *Artificial Intelligence: An Engineering Approach.* McGraw Hill. 1990.

[104] H.P. Schewfel. *Numerical Optimization of Computers Models.* John Wiley & Sons, New York, 1981.

[105] J.T. Schwartz y M. Sharir, *On the piano movers' problem: III. Coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal barriers.* International Journal of Robotics Research. Vol. 2, nº 3, páginas 46-75. Otoño 1983.

[106] M. Sharir y S. Sifrony, *Coordinated motion planning for two independent robots.* Proceedings of the fourth ACM Symposium on Computational Geometry, páginas 319- 328. 1988.

[107] Z. Shiller y S. Dubowsky, *On computing the global time-optimal motions of robotics manipulators in the presence of obstacles.* IEEE Transations on Robotics and Automation, Vol. 7, nº 6, páginas 785-797. Diciembre 1991.

[108] K.G. Shin y N.D. McKay, *Minimum-time control of robotics manipulators with geometric path constraints*, IEEE Transations on Automatic Control, Vol. AC-30, nº 6, páginas 531-541. Junio 1985.

[109] K.G. Shin y N.D. McKay, *Robust trajectory planning for robotic manipulators under payload uncertainties.* IEEE Transations on Automatic Control, Vol. AC-32, nº 12, páginas 1044-1054. Diciembre 1987.

[110] K.G. Shin y Q. Zheng, *Minimum-time collision-free trajectory planning for dual-robot systems.* IEEE Transactions on Robotics and Automation, Vol. 8, nº 5, páginas 641-644. Octubre 1992.

[111] D. Simon y C. Isik, *Suboptimal robot joint interpolation within user-specified knot tolerances.* Journal of Robotics Systems. Vol. 10, nº 7, páginas 889-911. 1993

[112] J.J. Slotine y H.S. Yang, *Improving the efficiency of time-optimal path-following algorithms.* IEEE Transations on Robotics and Automation, Vol. 5, nº 1, páginas 118-124. Febrero 1989.

[113] B.S. Steward, C.F. Liaw y C.C. White. *A bibliography of heuristic search research through 1992.* IEEE Trans. on Systems, Man and Cybernetics. Vol 24, no 2, Febrero 1994.

[114] H. Szczerbicka, *Are Genetic Algorithms a Panacea for Optimization of Simulation Models?.* Proceedings of the Conference ond Modelling and Simulation 1994, páginas 38-46. 1994.

[115] E. Tabarah, B. Benhabib y R.G. Fenton, *Optimal motion coordination of two robots - A polynomial parameterization approach to trajectory resolution.* Journal of Robotics Systems. Vol. 11, nº 7, páginas 615-629. 1994

[116] R.H. Taylor, *Planning and execution of straight line manipulator trajectories.* I.B.M. Journal on Research and Development, vol. 23, nº 4, páginas 424--436. Julio 1979.

[117] J. Tornero, G.J. Hamlin y R.B. Kelley, *Efficient distance functions using spherical-objects and their application to the two-Puma platform system.* Technical Report CIRSSE-TR-90-64, Center for Intelligence Robotics Systems for Space Exploration, Rensselaer Polytechnic Instititute, Troy (NY). 1990.

[118] J. Tornero, *Trabajo de investigación: Detección de colisiones en sistemas robotizados.* Concurso-oposición para plaza de Catedrático de Universidad. Universidad Politécnica de Valencia. Julio 1993.

[119] B. Tondu y H. El-Zorkany, *Identification of a Trajectory Generator Model for the PUMA-560 Robot,* Journal of Robotic Systems, 11(2), páginas 77-90, 1994.

[120] P. Tournassound, *A strategy for obstacle avoidance and its application to multirobot systems.* Proceedings of the 1986 IEEE Intarnational Conference on Robotics and Automation, páginas 1224-1229. 1986.

[121] C.K. Tsai, *Multiple robot coordination and programming,* Proceedings of the 1991 IEEE Int. Conf. on Robotics and Automation, páginas 978-985. Sacramento, California. 1991.

[122] R. Volpe y P. Khosla, *Manipulator control with superquadric artificial potential functions: Theory and experiments.* IEEE Transations on Systems, Man and Cybernetics, Vol. 20, nº 6, páginas 1423-1436. Noviembre-Diciembre 1990.

[123] M.A. Ward, *Robot cooperation: solving the motion coordination problem for two scara manipulators.* Ph. D. Dissertation. North Carolina State University. 1989.

[124] D. Whitley. *A Genetic Algorithm Tutorial.* Technical Report CS-93-103. Department of Computer Science. Colorado State University. Noviembre 1993.

[125] T.H. Wu y K.Y. Young, *Path planning in the presence of obstacles based on task requeriments.* Journal of Robotic Systems. Vol. 18, nº 8, páginas 703-716. 1994.

[126] Q. Xue, A. Maciejewsky y P.C.Y. Sheu, *Determining the collision free joint space graph for two cooperating robot manipulators*. IEEE Transations on Systems, Man and Cybernetics, Vol. 23, n° 1, páginas 285-294. Enero-Febrero 1993.

[127] Q. Xue y Y.P. Chien, *Determining the path search graph and finding a collision-free path by the modified A* algorithm for a 5-link closed chain*. Applied Artificial Intelligence vol. 9, páginas 235-255. 1995.

[128] M.Z. Yannanakis, C.H. Papadimitiou and H.T. Kung, *Locking policies: safety and freedom from deadlock*. IEEE Symposium on the Foundation of Computer Science, páginas 286-297. 1979.

UNIVERSIDAD DE SEVILLA

Reunido el Tribunal integrado por los abajo firmantes en el día de la fecha, para juzgar la Tesis Doctoral de D. MIGUEL ANGEL RIDAO CARLINI titulada GENERACIÓN AUTOMÁTICA DE TRAYECTORIAS LIBRES DE COLISIONES PARA MÚLTIPLES ROBOTS MANIPULADORES

acordó otorgarle la calificación de APTO CUM LAUDE

Sevilla, 17 de DICIEMBRE 19 95

El Vocal, El Vocal, El Vocal,

El Presidente El Secretario, El Doctorado,