Revista Iberoamericana de Automática e Informática Industrial
CEA comité español de automática
RIAI http://www.revista-riai.org
ISSN: 1697-7912. Vol. 6, Núm. 4, Octubre 2009, pp. 51-60

Planificación de Trayectorias Libres de Colisión para Múltiples UAVs usando el Perfil de Velocidad

Juan José Rebollo * **Iván Maza** * **Aníbal Ollero** *,**

* *Grupo de Robótica, Visión y Control*
Universidad de Sevilla
Edif. Escuela Superior de Ingenieros
Camino de los Descubrimientos, s/n, 41092 Sevilla, España
e-mail: juanjorebollo@hotmail.com; imaza,aollero@cartuja.us.es

** *Centro Avanzado de Tecnologías Aeroespaciales, FADA-CATEC*
Aerópolis, Parque Tecnológico Aeroespacial de Andalucía
N-IV, Km. 529, C/ Wilbur y Orville Wright, 17
41309, La Rinconada - Sevilla, España
e-mail: aollero@catec.aero

Resumen: Este artículo presenta un método de resolución de colisiones entre múltiples UAVs que comparten el espacio aéreo con aeronaves no cooperativas. El método propuesto encuentra trayectorias libres de colisión modificando el perfil de velocidad de los diferentes vehículos involucrados en la colisión y teniendo en cuenta la existencia de obstáculos móviles o vehículos no cooperativos. El objetivo es encontrar la solución más cercana a las trayectorias que tenían planificadas los UAVs inicialmente. La resolución de colisiones se divide en dos pasos: inicialización usando una búsqueda en árbol, y cálculo de la solución final empleando búsqueda Tabú. En el artículo se presentan diferentes simulaciones que ponen de manifiesto la eficiencia del método propuesto para la resolución de colisiones en tiempo real.

Palabras Clave: Múltiples UAVs, resolución de colisiones, búsqueda Tabú, búsqueda en árbol, discretización del espacio.

1. INTRODUCCIÓN

El empleo de múltiples vehículos aéreos no tripulados (UAVs) ha sido propuesto como una alternativa eficiente en misiones de vigilancia, detección, seguimiento y monitorización con aplicaciones en seguridad, protección del medio ambiente, intervención en casos de catástrofes y otras. Dicho empleo, cuando se compara con el uso de un único UAV, permite incrementar la cobertura (Maza and Ollero, 2007), disminuir retrasos de detección de eventos, incrementar la fiabilidad al no depender la misión de una única aeronave, así como disminuir la incertidumbre al disponerse de informaciones y medidas tomadas desde distintos puntos de vista (Ollero and Maza, 2007). El empleo de múltiples UAVs requiere métodos apropiados de coordinación ya que los UAVs tendrán que compartir recursos (espacio aéreo, espacio radioeléctrico, etc.). En particular, cuando se emplean múltiples UAVs, pueden producirse colisiones tanto entre los UAVs que se utilizan en la misión como con otras aeronaves con las que se comparte el espacio aéreo. Este es el problema que se aborda en este artículo, llevando a cabo primero una detección de tales colisiones y posteriormente su resolución modificando las trayectorias.

El problema de la resolución de colisiones puede tratarse de dos formas distintas. La primera consiste en calcular trayectorias libres de colisión, antes de que los vehículos empiecen a moverse. Este método no tiene restricciones temporales en el cálculo de la solución. En la segunda, las colisiones se resuelven en tiempo real una vez que son detectadas. En este caso, el tiempo de cálculo juega un papel muy importante. Este segundo problema es el que se resuelve en este artículo. El escenario en el que se encuentran los UAVs es dinámico, por ello se producirán cambios en las misiones y no será posible fijar las trayectorias desde el inicio.

El problema de planificación de movimientos para múltiples robots ha recibido gran atención en los últimos años. En (Richards and How, 2002), se plantea un problema de programación lineal entera-mixta (MILP) que se resuelve mediante paquetes de programas bien conocidos. La resolución de este problema posee una gran complejidad debido al gran número de restricciones y además no considera la existencia de obstáculos móviles. El método MILP se ha aplicado también a la optimización de trayectorias en presencia de obstáculos y amenazas (Ruz *et al.*, 2006). En (Tsubouchi and Arimoto, 1994) se propone un método para construir geométricamente unas trayectorias libres de colisión en el espacio (x, y, t). Primero, los autores evalúan la posición y velocidad de los obstáculos móviles. Teniendo en cuenta que la velocidad de los obstáculos se supone constante, se calculan una serie de cilindros oblicuos en el espacio (x, y, t) que deben ser evitados. El problema consiste en encontrar una trayectoria que conecte la posición inicial con una línea vertical que representa el objetivo. Este método no resuelve el problema de planificación de movimientos multirobot, en el cual la trayectoria de más de un robot puede cambiar para resolver las colisiones. El método presentado en (Owen and Montano, 2005) tiene la misma característica. En este último caso, trabajando en el espacio de las velocidades, se

obtienen buenos resultados, considerando obstáculos móviles y fijos, pero no se plantea la modificación de más de una trayectoria de forma coordinada.

En (Ferrari *et al.*, 1997) se adopta un enfoque distinto del problema, obteniéndose caminos alternativos mediante pequeñas variaciones del movimiento del robot en el tiempo y en el espacio, empleándose las estrategias "Stop & Go" y "Shape Changing" para evitar las colisiones, una vez que se ha encontrado el camino. Se supone que los vehículos poseen una dinámica muy simple, y no se consideran obstáculos móviles. En (Pallottino *et al.*, 2007), se propone una estrategia para guiar múltiples vehículos entre un punto de inicio y un punto objetivo independientes, y asegurándose trayectorias libres de colisión. En estos métodos todos los vehículos siguen las mismas reglas de tráfico. Se mueven con velocidad constante, aunque se define una zona de seguridad y la velocidad en esta zona puede ser cero. Sin embargo, este método normalmente lleva a variaciones de las trayectorias para evitar las colisiones, que no serían necesarias si se evitaran simplemente modificando el perfil de velocidad de los vehículos.

En (Fujimori and Teramoto, 2000) se presenta un enfoque geométrico. El ángulo de dirección y la velocidad de los robots móviles se usan como variables de control para la navegación y la resolución de colisiones. Este método solo asegura resolución de colisiones para dos vehículos. En (Cruz *et al.*, 1998) se desarrolla un método basado en la planificación de velocidades con obstáculos móviles. Los obstáculos móviles se incluyen como restricciones del movimiento. Este método no considera la posibilidad de modificar la trayectoria de todos los vehículos involucrados en la colisión.

En (Kant and Zucker, 1986) se propone la descomposición del problema de resolución de colisiones en dos: planificación de las trayectorias (PPP) y planificación de velocidades (VPP). Una vez que se ha obtenido una trayectoria (PPP), se deberá encontrar un perfil de velocidades que evite las colisiones para tales trayectorias (VPP). Este segundo problema es el que se trata en nuestro trabajo.

En este artículo se resuelve un problema de planificación de movimientos en 3D para múltiples UAVs, compartiendo el espacio con vehículos aéreos no cooperativos (obstáculos móviles). Se obtendrá un nuevo perfil de velocidad para los diferentes UAVs involucrados en la colisión. Este artículo presenta un nuevo enfoque heurístico que va a permitir encontrar soluciones subóptimas en menos tiempo que métodos óptimos, gracias a la búsqueda de soluciones en un espacio discreto. Para aplicaciones en tiempo real, como la abordada en este artículo, es primordial encontrar una solución en un tiempo acotado y del orden de los segundos, aunque se tenga que sacrificar levemente la optimalidad de la solución. El objetivo es calcular una solución que cambie la trayectoria inicial lo mínimo posible, modificando solamente el perfil de velocidades de los diferentes vehículos. Los UAVs que deben modificar su trayectoria para evitar la colisión, puede que estén ejecutando una misión para la que sea crítico mantener cierta trayectoria, tal como por ejemplo la búsqueda o seguimiento de un objeto. Por ello se ha optado por modificar el perfil de velocidades lo menos posible, sin tener en cuenta otras estrategias que impliquen un mayor cambio en la trayectoria. La modificación de la altitud, aun siendo la estrategia más simple, en general no va a ser posible, debido a que el espacio aéreo se estructura en capas (Bichi and Pallottino, 2000). La mayoría de los vehículos tienen grandes limitaciones dinámicas, que no les permiten parar o modificar sus trayectorias lo suficientemente rápido. Como se pone de manifiesto en (Munoz *et al.*, 1994), la consideración de las restricciones cinemáticas y dinámicas tiene una importante influencia en la evitación de colisiones y planificación de trayectorias. En este artículo se considera también un modelo del UAV que permite tener en cuenta dichas restricciones en el movimiento de los diferentes vehículos involucrados. Modificando tan solo las velocidades no es posible resolver cualquier colisión, por ejemplo un choque frontal, por lo que se debería modificar el camino a seguir y posteriormente aplicar el algoritmo presentado en este trabajo para optimizar el perfil de velocidades. Este tipo de conflictos serán detectados por el algoritmo desarrollado.

El resto de este artículo está organizado como se describe a continuación. En la siguiente sección se plantea el problema que se va a resolver. En la Sección 3 se describen los algoritmos con los que se resuelve tal problema, en primer lugar la Búsqueda en Árbol y posteriormente la Búsqueda Tabú. A continuación, en la Sección 4, se presentan dos simulaciones, la primera de ellas con 3 UAVs y la segunda con 6 UAVs. Aunque en dichas simulaciones se emplean trayectorias compuestas por segmentos rectilíneos, los métodos presentados permiten resolver el problema para cualquier tipo de trayectoria. Finalmente, se presentan las conclusiones y las líneas de trabajo futuras.

2. PLANTEAMIENTO DEL PROBLEMA

El problema de resolución de colisiones puede ser difícil debido al gran número de soluciones posibles. Se puede simplificar el problema dividiéndose el espacio en celdas. Las celdas no van a permitir encontrar una solución óptima, pero permiten el uso de algoritmos de búsqueda muy rápidos que permitirán encontrar la solución deseada.

El primer paso para encontrar una solución al problema de resolución de colisiones es la detección de colisiones potenciales. El espacio cartesiano se divide en celdas cúbicas. Una trayectoria se puede describir como una secuencia de celdas a las que se les asocia un tiempo de entrada y un tiempo de salida. Por lo tanto, para asegurar trayectorias libres de colisión, mientras un vehículo ocupa una celda, no puede entrar otro en ella. Cada UAV conoce todas las listas de celdas por las cuales pasarán aquellos UAVs que están dentro de un cierto radio R en un cierto horizonte temporal ΔT. Los UAVs sólo conocerán las celdas de paso de los UAVs de su entorno, y solo transmitirán sus propias celdas a tales UAVs. El problema se resuelve localmente y la cantidad de información considerada depende del radio R. Esto facilita detectar si una colisión se va a producir, porque cada UAV simplemente tiene que detectar solapamiento temporal entre una celda de su trayectoria y una celda que pertenece a la trayectoria de otro UAV.

La rejilla tridimensional propuesta en este artículo disminuye la transmisión de información entre los vehículos, no siendo necesario transmitir las trayectorias completas. Esta estrategia también reduce el tiempo de detección de colisiones con respecto a los métodos que usan la trayectoria completa. El objetivo es encontrar las condiciones temporales para cada celda, que nos den unas trayectorias libres de colisión.

En el método propuesto es posible modificar las trayectorias de todos los vehículos involucrados en la colisión, para poder encontrar una solución mejor. En el caso de que cierto vehículo

no pueda modificar su trayectoria para alcanzar su objetivo adecuadamente, se consideraría como *obstáculo móvil*, y mantendría su trayectoria original tras resolver el problema.

En la resolución de colisiones se considerarán los vehículos que las han detectado en su trayectoria, y todos aquellos vehículos cuyas trayectorias intersectan con la de alguno de estos vehículos. El algoritmo propuesto considera tres tipos de vehículos implicados en la resolución del problema:

- *Implicados directos*: son todos aquellos vehículos involucrados en una determinada colisión potencial detectada.
- *Implicados indirectos*: son los vehículos cuyas trayectorias se cruzan con las de los *implicados directos* y cuyos perfiles de velocidades pueden ser modificados por el sistema de resolución de conflictos.
- *Obstáculos móviles:* vehículos involucrados en los conflictos, pero que no modificarán su perfil de velocidades (son vehículos *no cooperativos*).

Los vehículos *implicados indirectos* y *obstáculos móviles*, son aquellos vehículos que podrían estar implicados en los nuevos conflictos que pudieran surgir al resolver una determinada colisión entre los *implicados directos* modificando sus perfiles de velocidades. Por otro lado, a veces puede ser beneficioso considerar algunos vehículos cooperativos como *obstáculos móviles*, porque se disminuye el intercambio de información entre los diferentes UAVs y el tiempo de cómputo de los algoritmos. Esto se debe a que no se establecen comunicaciones con los obstáculos móviles al resolver la colisión. Esta estrategia también se ha tener en cuenta cuando la tarea que esté llevando a cabo cierto vehículo no permita variaciones en las velocidades.

2.1 Formulación

Sea j el identificador de una celda en el espacio. Consideremos una variable C_{ij} que valdrá 1 si el vehículo i pasa a través de la celda j, ó 0 en caso contrario. Sea t_{ij}^{in} el instante en el que el vehículo i entra en la celda j, y t_{ij}^{out} el instante en el que el vehículo i abandona la celda j. De esta manera, el intervalo de tiempo que el UAV i pasa en la celda j vendrá dado por $T_{ij} = [t_{ij}^{\text{in}}, t_{ij}^{\text{out}}]$, y podemos decir que habrá una colisión potencial en la celda j si se verifica:

$$\bigcap_{i:C_{ij}=1} T_{ij} \neq \emptyset \tag{1}$$

siendo $\emptyset$ el conjunto vacío. Dado un número total de vehículos N, habrá un conflicto entre m vehículos en la celda de identificador j si

$$\sum_{i=1}^{N} C_{ij} = m \tag{2}$$

Sea Δt_{ij} el tiempo que el vehículo i permanece en la celda j

$$\Delta t_{ij} = t_{ij}^{\text{out}} - t_{ij}^{\text{in}} \in [t_{ij}^{\min}, t_{ij}^{\max}] \tag{3}$$

donde $t_{ij}^{\min}$ y $t_{ij}^{\max}$ son los tiempos mínimo y máximo para recorrer la celda de identificador j de acuerdo con el modelo del UAV i en la trayectoria considerada.

El problema consiste en el cálculo para cada UAV del perfil de velocidades que da lugar a unos intervalos de paso T_{ij}, de forma que en cada celda con conflicto se verifique:

$$\bigcap_{i:C_{ij}=1} T_{ij} = \emptyset, \tag{4}$$

y además se minimice para cada UAV el siguiente índice:

$$J = \sum_{p=0}^{S_i-1} (\Delta t_p - \Delta t_p^{\text{ref}})^2 \tag{5}$$

donde S_i es el número de celdas en la trayectoria del vehículo i, Δt_p^{ref} es el tiempo que se tarda en la trayectoria de referencia en pasar a través de la celda de identificador p y Δt_p es el tiempo que se tarda en la trayectoria solución en pasar por dicha celda. El objetivo del coste (5) es encontrar una trayectoria solución cercana a la trayectoria de referencia, la cual es la trayectoria de los vehículos antes de que se detectara la colisión potencial. Minimizando este coste el cambio en las trayectorias es mínimo.

La ecuación (3) se establece teniendo el cuenta el modelo del UAV que se presentará a continuación. Así, el tiempo que cada vehículo puede permanecer en cierta celda dependerá de sus velocidades máxima y mínima, así como de la distancia que recorre en su interior.

2.2 Modelo del UAV

Los UAVs se mueven a lo largo de su trayectoria de acuerdo con el modelo (McLain and Beard, 2005):

$$\begin{aligned} \dot{x}_i &= v_i cos(\psi_i) \\ \dot{y}_i &= v_i sin(\psi_i) \\ \dot{\psi}_i &= \alpha_{\psi i}(\psi_i^c - \psi_i) \\ \dot{v}_i &= \alpha_{vi}(v_i^c - v_i) \\ \ddot{h}_i &= -\alpha_{\dot{h}i}h_i + \alpha_{hi}(h_i^c - h_i) \end{aligned} \tag{6}$$

donde $\alpha_{\psi i}$, α_{vi}, α_{hi} y $\alpha_{\dot{h}i}$ son constantes conocidas que dependen de la implementación del UAV. Las variables de control serán ψ_i^c y v_i^c.Las coordenadas x_i, y_i, h_i indican la posición en el espacio del UAV y ψ_i la orientación en el plano xy. Por simplicidad solo se considera una variable de orientación. La velocidad del UAV viene dada por v_i. Respeto a $\dot{\psi}_i$ y v_i, se va a considerar:

$$\begin{aligned} -c_i < \dot{\psi}_i < c_i \\ v_i^{\min} < v_i < v_i^{\max} \end{aligned} \tag{7}$$

donde c_i, $v_i^{\min}$ y $v_i^{\max}$ son constantes positivas que dependen de la dinámica del UAV.

Los tiempos máximos y mínimos que un UAV puede permanecer en cierta celda se calculan teniendo en cuenta este modelo. Para tal cálculo, se necesita la distancia que cada vehículo viaja en la celda correspondiente como se ha comentado anteriormente.

Los dos algoritmos presentados en este artículo tienen en cuenta el modelo dinámico de los UAVs. La Búsqueda en Árbol calcula el tiempo asociado a cada rama considerando el modelo

dinámico. Posteriormente, en la Búsqueda Tabú cada nuevo vecino será una solución que se atiene al modelo dinámico presentado en esta sección.

3. MÉTODO DE RESOLUCIÓN DE COLISIONES

El objetivo es encontrar cuanto tiempo tiene que permanecer cada vehículo en las diferentes celdas de su trayectoria para asegurar que no existen conflictos. Incluso para este problema, no es posible encontrar una solución óptima en tiempo polinomial, y es importante encontrar una solución rápidamente, porque de otro modo puede que no sea posible evitar la colisión. Por estas razones, en este artículo se desarrolla un método heurístico basado en la combinación de la Búsqueda en Árbol y la Búsqueda Tabú. Estos dos algoritmos heurísticos obtendrán un nuevo perfil de velocidad para los diferentes vehículos que evitan las colisiones. Los algoritmos se basan en la idea de que la forma más rápida de encontrar una solución, es tener en cuenta una serie de reglas lógicas. El Algoritmo de Búsqueda en Árbol encuentra una solución libre de colisiones, pero sin considerar el coste (5). Esta solución será la solución inicial que el algoritmo de Búsqueda Tabú necesita para obtener la solución que estamos buscando en este artículo. Todos los algoritmos tienen en cuenta el modelo de los UAVs y la distancia recorrida en cada celda.

3.1 Algoritmo de Inicialización: Búsqueda en Árbol

El algoritmo descrito en este apartado tiene como objetivo encontrar una primera solución libre de colisiones, aunque no será la que minimiza la función de coste presentada anteriormente. Esta solución servirá al algoritmo de Búsqueda Tabú como punto de partida. La Búsqueda en Árbol nos proporcionará una solución en la que los vehículos viajan a la velocidad máxima que les asegura una trayectoria libre de colisión. Esta elección se justifica en misiones tales como las de exploración en las que se pretende minimizar el tiempo de ejecución. La idea básica del algoritmo es que una vez definido el orden de paso de los UAVs que intervienen en cierto conflicto, existirá solución al conflicto si pasando el primero de los vehículos por la zona de conflicto a la velocidad máxima, el resto pueden pasar a velocidades que les permitan cruzar en el orden previamente establecido y sin que se provoque ninguna colisión entre ellos. Si alguno de los vehículo aun yendo a la velocidad mínima chocase con el anterior, entonces no habría solución para el orden de paso definido. Todos los cálculos que llevará a cabo el algoritmo de Inicialización, se realizan teniendo en cuenta el modelo del UAV presentado en (6).

Se define el *orden de paso*, como el orden en el que los vehículos cruzan a través de una potencial colisión. Un conflicto con m vehículos implicados, tiene $m!$ ordenes de paso distintos.

El algoritmo explora los diferentes *ordenes de paso* en cada conflicto hasta que se encuentra una solución. Dependiendo de la topología del conflicto, puede que no se encuentre solución, y en tal caso se tendría que modificar el camino a seguir con algunos de los métodos existentes (Wollkind, 2004; Massink and Francesco, 2001) y posteriormente las velocidades. Esto ocurrirá cuando se produzca un choque frontal, trasero o la dinámica del vehículo no permita la modificación de las velocidades necesaria para evitar la colisión. En el algoritmo presentado, en primer lugar se busca una solución explorando el *orden de paso* más lógico, para el cual el vehículo que tiene que recorrer menos distancia para llegar al conflicto pasaría primero. Para cada orden, es posible comprobar si existe solución en un tiempo computacional reducido (del orden de centésimas de segundo para las simulaciones que presentaremos, aunque dependerá de la complejidad del problema que se esté tratando). Si no hay solución para cierto *orden de paso*, el algoritmo permuta el orden de aquellos vehículos que tienen que recorrer más distancia para llegar al conflicto. Finalmente se llevará a cabo una nueva búsqueda para tal *orden de paso*.

Si el problema tiene un total de M conflictos con m_k vehículos implicados en el conflicto k-ésimo, habrá $m_0!m_1!...m_{M-1}!$ órdenes que comprobar. Cuando ya se ha explorado cierto orden de paso y no hay solución, el algoritmo permuta el orden de uno de los conflictos. Primero se permuta el conflicto k-ésimo cuyo coste R_k definido como

$$R_k = \mu_k - \sigma_k \tag{8}$$

es mayor, donde μ_k y σ_k son la media y la desviación típica de la distancia que cada vehículo implicado tiene que recorrer para llegar al conflicto k-ésimo. El algoritmo permuta primero el orden de los vehículos involucrados en los conflictos que están más lejos del inicio de la trayectoria. Esto se debe a que cuando un conflicto está cerca del inicio de la trayectoria, los vehículos que tienen que recorrer menos distancia para llegar a él, tienen muchas probabilidades de pasar primero en la solución del problema. Sin embargo, en un conflicto que está más lejos del inicio de la trayectoria, otro conflicto más cercano al inicio podría afectar al criterio de búsqueda definido arriba (el vehículo que recorre menos distancia para llegar a un conflicto pasa primero). Las diferencias en la distancia que los vehículos recorren para llegar a cierto conflicto hacen al *orden de paso* inicial más apropiado. El término σ_k en (8) tiene en cuenta esta idea, disminuyendo el valor de R_k del conflicto. El número de ordenes de paso existentes es elevado, pero se ha de tener en cuenta que muchos se descartarán muy rápidamente por la Búsqueda en Árbol.

A cada una de las trayectorias de los UAVs se le asociará un árbol. Cuando se va desde un nodo al siguiente se pasa por una celda. Entonces, si se va desde el nodo raíz al nodo más lejano, se recorrerá la trayectoria completa del UAV asociado a ese árbol. Se encontrará una solución cuando todos los árboles se construyan totalmente. La distancia entre dos nodos consecutivos está directamente relacionada con el tiempo que el UAV está en la celda asociada. El Algoritmo 1 muestra como se construyen los árboles y como se encuentra la solución del problema.

Básicamente, los árboles crecen a medida que se calcula el tiempo que cada vehículo pasa en las celdas de su trayectoria yendo a la velocidad máxima, hasta que se alcance una celda con conflicto. Cuando se detecta el conflicto, una rama asociada a él se crea o no dependiendo de si es el turno de tal árbol. Ese turno corresponde con el *orden de paso* que el árbol está comprobando en la iteración correspondiente. Por lo tanto, es el turno de un determinado vehículo si las ramas de los otros árboles asociadas a la misma celda con conflicto, y la asociada al UAV que tiene que pasar antes a través del conflicto, ya se han creado. Si es el turno, se comprueba si hay colisión en el conflicto comprobándose si hay solapamiento temporal entre los tiempos asociados a otras ramas asociadas

Algoritmo 1 Búsqueda en Árbol.

mientras no se haya encontrado una solución o no se hayan explorado todos los *ordenes de paso* **hacer**
  mientras no se haya llegado al final de cada árbol, y haya solución **hacer**
    para cada árbol, si no se ha llegado al final de este **hacer**
      Se avanza a velocidad V_{max} hasta el siguiente conflicto, creando las ramas pertinentes
      si no se ha llegado al final del árbol **entonces**
        si es el turno del árbol tratado en el conflicto **entonces**
          Se pasa por el conflicto, creándose la rama asociada a este
          si hay colisión **entonces**
            Se ha de ir hacia atrás en el árbol y crear nuevas ramificaciones, que resuelvan la colisión. Esto puede implicar que se retroceda también en los otros árboles para asegurar trayectorias libres de colisión.
            Si se llega al inicio del árbol actual al retroceder, es que no hay solución para el *orden de paso* considerado actualmente.
          fin si
        fin si
      fin si
    fin para
  fin mientras
fin mientras

al mismo conflicto. Si una colisión aparece, el algoritmo tiene que cambiar la velocidad en las celdas previas del vehículo que está generando el árbol y ha provocado el solapamiento temporal, y ahora no se podrá viajar a la velocidad máxima en las celdas previas al conflicto.

En un árbol aparece una bifurcación cada vez que se tiene que crear una nueva rama porque se detectó una colisión. La Figura 1 muestra cómo el algoritmo resuelve las colisiones que aparecen. El algoritmo va hacia atrás y crea nuevas ramas. La rama a, no es válida porque se necesitan ramas de mayor longitud para evitar la colisión, y debido al modelo del vehículo, no puede ser más larga, ya que se está yendo a la velocidad mínima. La velocidad mínima conlleva un mayor tiempo de estancia y por ello mayor longitud de la rama. Las ramas b y c son válidas, y permiten llegar más tarde a la celda (8,10,10). La Figura 2 muestra que había solapamiento temporal entre los vehículos 1 y 2 en la celda (8,10,10), lo cual indica que hay colisión. Después de crear las nuevas ramas la colisión desaparece, tal como se puede apreciar en la Fig. 2.

Los cambios llevados a cabo por el algoritmo de Búsqueda en Árbol para evitar el solapamiento temporal, podrían afectar a otros árboles y provocar nuevas colisiones. La Figura 1 muestra que el árbol asociado al UAV3 tiene que generar nuevas ramas (rama d), porque el UAV1 pasa antes por la celda (5,8,8) y el UAV3 colisionaría con el UAV1 si el árbol asociado al UAV3 no reconstruyera sus ramas previas al nodo m. Se tiene que recalcular el tiempo que el UAV3 permanece en las celdas previas a la (5,8,0), porque la condición que se tiene que satisfacer en el conflicto ha cambiado.

Si el algoritmo de búsqueda en árbol no encuentra una solución para cierto *orden de paso*, todos los árboles se volverán a

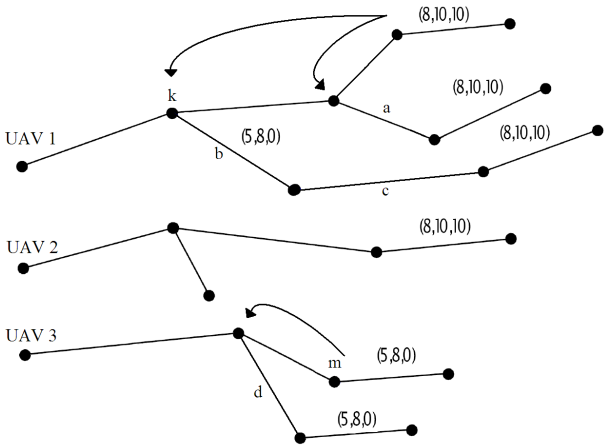

Figura 1. El algoritmo vuelve atrás y crea nuevas ramas (b, c y d) que permiten evitar la colisión. Los 3 números que aparecen entre paréntesis son el identificador de la celda tridimensional.

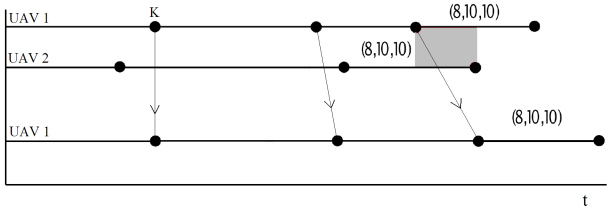

Figura 2. En este diagrama temporal se puede observar el solapamiento temporal inicial entre los UAVs 1 y 2 en la celda de identificador (8,10,10). Dicho solape corresponde a una colisión potencial y se ha indicado en el diagrama mediante el área rectangular gris entre las líneas temporales de los UAVs 1 y 2.

construir desde el inicio considerándose el siguiente *orden de paso* determinado por el criterio de búsqueda que se define en (8).

Los UAVs *implicados directos* e *indirectos* construyen el árbol de la misma forma. Sin embargo, el árbol asociado a un obstáculo móvil está terminado desde el inicio, y no se podrán modificar porque sus trayectorias y velocidades son fijas.

El algoritmo de Búsqueda en Árbol permite encontrar la solución al problema en un tiempo reducido, del orden de milisegundos, comparándolo con otros métodos que resuelven el problema sin considerar un espacio dividido en celdas cúbicas (Richards and How, 2002), los cuales tardan unos pocos minutos ya que algunos modifican toda la trayectoria.

3.2 Búsqueda Tabú

El algoritmo de Búsqueda en Árbol encuentra una solución al problema pero no considera la función de coste (5). El algoritmo de Búsqueda Tabú (Glover and Laguna, 1997) modifica la solución que la Búsqueda en Árbol encontró, minimizando el coste. La Búsqueda Tabú mejora el resultado de un método de búsqueda local, usando estructuras de memoria para evitar mínimos locales. Los elementos que es necesario definir para el correcto funcionamiento de la Búsqueda Tabú son los siguientes:

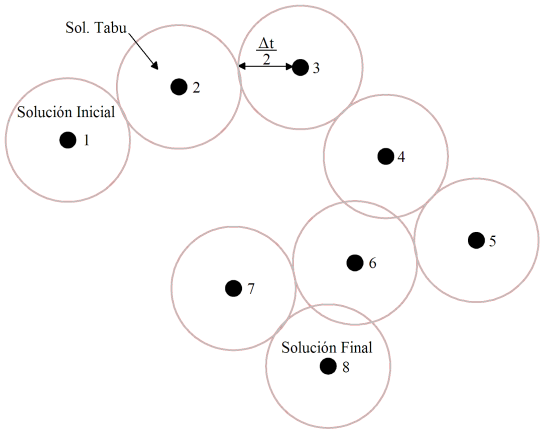

Figura 3. Representación del espacio de las soluciones a las cuales se podrá llegar desde la solución inicial obtenida mediante la Búsqueda en Árbol. Los puntos negros representan soluciones seleccionadas por la Búsqueda Tabú.

- *Solución inicial* x: Será la solución obtenida en la Búsqueda en Árbol. Esta solución está formada por una serie de variables temporales que determinan el tiempo que los vehículos pasan en cierta celda. Cada variable temporal es el tiempo que cierto vehículo está en un grupo de celdas. Gracias a la agrupación de celdas, el número de variables se reduce, lo cual nos permite encontrar la solución deseada más rápido.
- *Vecindario* $N(x)$: En cada iteración el algoritmo de Búsqueda Tabú explora el vecindario $N(x)$, y elige la mejor solución.
- *Lista tabú*: La Búsqueda Tabú necesita una memoria para recordar las últimas soluciones visitadas y evitar mínimos locales. En (Hertz *et al.*, 1995), se demuestra que un valor del tamaño de la lista para el que se obtienen buenos resultados es 7. En la Fig. 3) se puede ver cómo la Búsqueda Tabú evoluciona desde la solución inicial hasta la final, actualizando en cada iteración la lista tabú incluyendo las soluciones internas al radio indicado, y que por lo tanto no podrán ser solución hasta que pase cierto número de iteraciones y se borren de la lista tabú.
- *Criterio de aspiración*: Una solución que pertenece al vecindario es válida si no es una solución tabú. Pero algunas soluciones se han de considerar incluso si son soluciones tabú, por ejemplo soluciones que son mejores que la mejor solución encontrada hasta el momento (Gengreau, 2002).
- *Criterio de finalización*: Es una condición de finalización de la Búsqueda Tabú. Se considera un número máximo de iteraciones sin mejorar la solución óptima encontrada hasta el momento.

El Algoritmo 2 muestra el pseudocódigo de la Búsqueda Tabú.

En nuestro problema, el vecindario consistirá en todas las soluciones obtenidas incrementando en Δt, una de las variables de tiempo de x. Esta nueva solución no tiene colisiones y satisface las restricciones impuestas por la dinámica del modelo. Las soluciones tabú serán todas aquellas cuya distancia a las soluciones de la lista es menor que $\Delta t/2$ (ver Fig. 3). En dicha figura se representa el espacio de las soluciones a las cuales se podrá llegar desde la solución inicial obtenida mediante la

Algoritmo 2 Pseudocódigo Búsqueda Tabú.

Elegir $x \in X$ para empezar el proceso de búsqueda y hacer $x_{op} = x$
mientras no se cumpla el criterio de finalización **hacer**
  Se busca $x' \in N(x)$ que minimice $f(x)$ y que no está en la lista tabú, o si lo está, cumpla los criterios de aspiración
  si $f(x') < f(x_{op})$ **entonces**
    $x_{op} = x'$
  fin si
  Se incluye x' en la lista tabú
fin mientras

Búsqueda en Árbol. Los puntos negros representan soluciones seleccionadas por la Búsqueda Tabú. Esas soluciones se corresponden con la iteración que se indica en cada punto (serían las x'). El círculo que rodea cada punto negro contiene las soluciones que se van a añadir a la lista tabú en la iteración tratada (por lo que no serán válidas durante cierto numero de iteraciones que viene determinado por el número de elementos de la lista tabú), y en nuestro caso son las que están a una distancia inferior a $\Delta t/2$. Estos círculos son tangentes ya que se optó por considerar como vecindario en cada iteración las soluciones a una distancia Δt de la última solución encontrada.

La Búsqueda Tabú modifica el tiempo que los *implicados directos* e *indirectos* pasan en cada celda. La Figura 3 muestra cómo la solución se mueve desde la solución inicial que encontró el algoritmo de Búsqueda en Árbol, hacia otra solución en la cual el valor del coste es menor, y por lo tanto será una solución mejor para el problema que se pretende resolver. En la iteración final (la octava), se satisface el criterio de finalización y por eso ya no se continua con la búsqueda.

4. RESULTADOS DE SIMULACIÓN

En esta sección se presentan dos simulaciones. Se verá como para los dos problemas que se van a proponer con 3 y 6 UAVs, se han encontrado soluciones de gran calidad en tiempos muy reducidos.

4.1 Simulación con 3 UAVs

En esta sección se presenta una simulación con 3 UAVs implicados. El UAV1 y el UAV2 están barriendo una región, y el UAV3 es un vehículo no cooperativo teleoperado, el cual se considera como obstáculo móvil. La Figura 4 muestra la proyección $x - y$ de las trayectorias completas de los tres UAVs. Las colisiones se detectan y resuelven en el interior del área cuadrada rayada en la Fig. 4. En el espacio xyz de la Fig. 5 se muestra en tres dimensiones y ampliada dicha área de resolución de la Fig. 4.

El UAV1 y el UAV2 se consideran vehículos *implicados directos* y el UAV3 *obstáculo móvil*. La Figura 6 es un diagrama temporal de las trayectorias originales, en el que cada punto indica una transición hacia la siguiente celda de la trayectoria. En tal figura se puede ver que hay solapamiento temporal en la celda (13,13,10), y por lo tanto colisión. Hay dos conflictos, uno en la celda (13,13,10) entre el UAV1 y el UAV2 y otro en la celda (16,13,10) entre el UAV1 y el UAV3. El objetivo es encontrar una solución libre de colisiones, que difiera de las trayectorias iniciales lo mínimo posible.

La Figura 7 muestra los resultados obtenidos por el algoritmo de Búsqueda en Árbol. Se puede ver que la trayectoria del

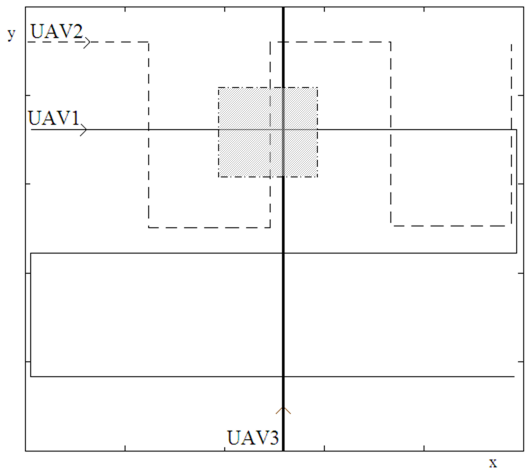

Figura 4. Proyección $x-y$ de las trayectorias de los 3 UAVs. El UAV1 y el UAV2 están barriendo una región, y el UAV3 es un vehículo no cooperativo teleoperado.

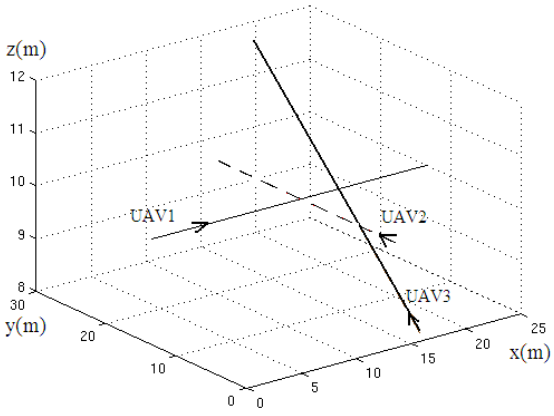

Figura 5. Detalle de la zona rallada de la Fig. 4 en la que se resuelve el problema para los tres UAVs.

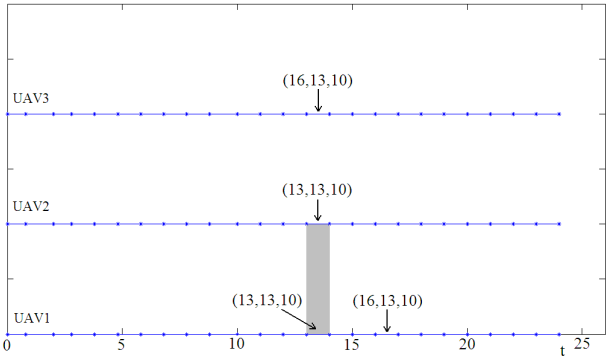

Figura 6. Diagrama temporal de las trayectorias iniciales, en el que cada punto indica una transición hacia la siguiente celda de la trayectoria.

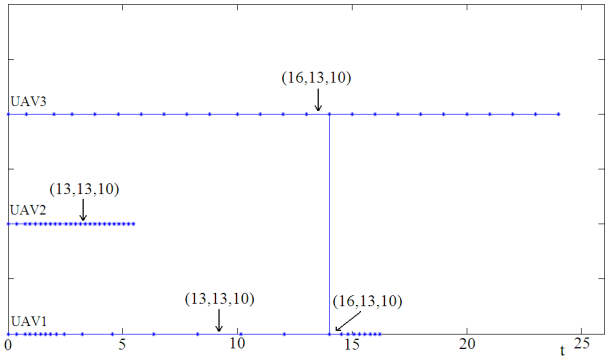

Figura 7. Resultados obtenidos por el algoritmo de Búsqueda en Árbol. Se puede ver que la trayectoria del UAV3 no cambió porque se consideró como *obstáculo móvil*.

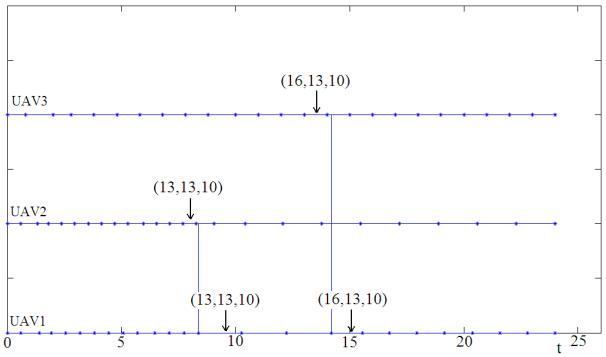

Figura 8. Solución obtenida al ejecutar la Búsqueda Tabú sobre la solución calculada por la Búsqueda en Árbol.

UAV3 no cambió porque se consideró un *obstáculo móvil*. Sin embargo, en la solución para el UAV1 y el UAV2, estos viajan a la velocidad más elevada que asegura que no se produce ninguna colisión. El UAV1 va más lento que el UAV2, ya que el conflicto entre el UAV1 y el UAV3 fija una restricción temporal en la celda (16,13,10) que se debe satisfacer. El UAV3 está más cerca del conflicto que el UAV1, por ello el algoritmo de Búsqueda en Árbol comprueba si hay solución pasando el UAV3 primero. Por lo tanto, dado que hay solución si pasa el UAV3 delante del UAV1, este será el *orden de paso* definitivo. Esto hace que el UAV1 tenga una restricción temporal en la celda (16,13,10).

Ahora si ejecutamos la Búsqueda Tabú para la solución encontrada, se obtiene la solución que puede verse en la Fig. 8. Esta solución está compuesta de unas trayectorias prácticamente iguales a las que se tenían inicialmente, las cuales se observan en la Fig. 6, pero ahora no existen colisiones.

Un problema con 3 UAVs donde cada uno de ellos tiene una trayectoria compuesta por 30 celdas, con $\Delta t = 0{,}1$ segundos se ha resuelto en menos de un segundo, lo cual se ajusta a las restricciones temporales de aplicaciones de navegación de aeronaves en tiempo real. El tiempo de cómputo no depende de la forma de las trayectorias (no tienen por qué ser rectilíneas), porque cada una de ellas será una secuencia de celdas y los algoritmos las tratan siempre de la misma forma.

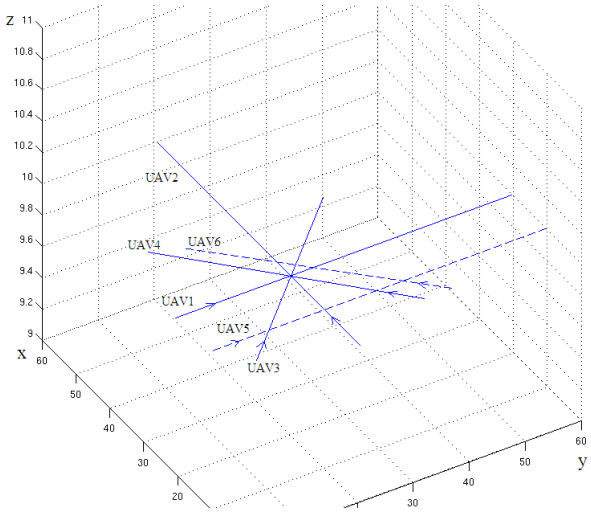

Figura 9. Trayectorias originales en la simulación con 6 UAVs, donde los vehículos UAV1, UAV2, UAV3 y UAV4 (*implicados directos*) colisionan en la celda (20,20,10), y a su vez tenemos el UAV5 y el UAV6 (obstáculos móviles) pasando cerca de la colisión.

4.2 Simulación con 6 UAVs

En este apartado se presentan los resultados obtenidos al resolver un problema con 6 UAVs. De estos 6 UAVs, 4 son los que tendrán que evitar la potencial colisión. Los otros dos UAVs se deberán tener en cuenta para hallar las nuevas trayectorias, debido a que pasan muy cerca de la colisión a resolver. En la Fig. 9 pueden verse las trayectorias originales, donde los vehículos UAV1, UAV2, UAV3 y UAV4 (*implicados directos*) colisionan en la celda (20,20,10), y a su vez tenemos el UAV5 y el UAV6 (obstáculos móviles) pasando cerca de la colisión. En la resolución de este problema no se modificarán las trayectorias del UAV5 y el UAV6, simplemente se impondrán unas condiciones temporales en ciertas casillas, que deberán respetar el resto de UAVs en la resolución del problema. Cada una de las trayectorias de los UAVs está compuesta por 60 celdas, se ha tomado $\Delta t = 0{,}1$ segundos, y se tienen las colisiones y conflictos registrados en la Tabla 1.

En las Figs. 10 y 11, se tiene un diagrama temporal de las trayectorias antes y después de la resolución. En este diagrama, el tiempo existente entre un nodo y el siguiente representa el tiempo de estancia en la celda asociada. Es de destacar que se ha resuelto un problema entre 6 UAVs modificando sólo las velocidades y haciendo que el perfil de velocidades inicial y el encontrado al resolver el problema difieran muy poco. Se puede ver claramente que las Figs. 10 y 11 son muy similares. De los nuevos tiempos de estancia en las celdas, lo más relevante es cómo se ha resuelto la colisión potencial en la celda (20,20,10).

En la Fig. 12 se puede ver el solapamiento espacial y temporal que existía, y en la Fig. 13 se observa cómo después de ejecutar los algoritmos de resolución de colisiones, desaparece tal solapamiento. En la Fig. 14 se tiene una representación espacial de tal información para distintos instantes de tiempo. Se muestra la celda en el plano $x - y$ en la que se encuentra cada uno de los UAVs para cuatro instantes distintos.

Finalmente se muestra cómo evoluciona el tiempo de cómputo en función del número de UAVs. Se ha calculado el tiempo de cómputo considerando primero sólo el UAV1 y el UAV2,

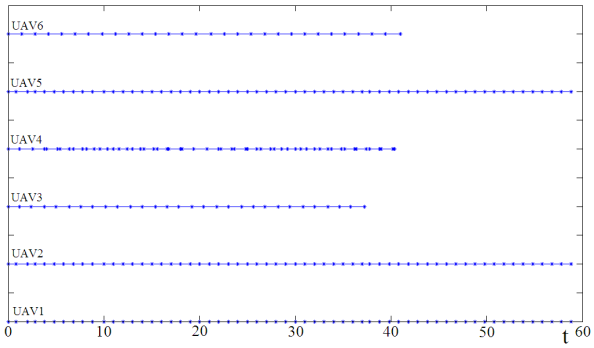

Figura 10. Diagrama temporal con las trayectorias iniciales.

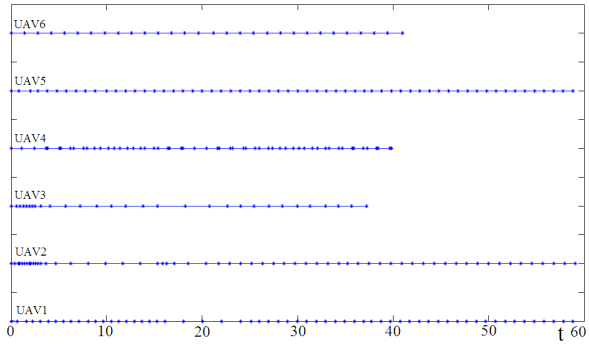

Figura 11. Diagrama temporal tras aplicar el algoritmo de resolución de conflictos.

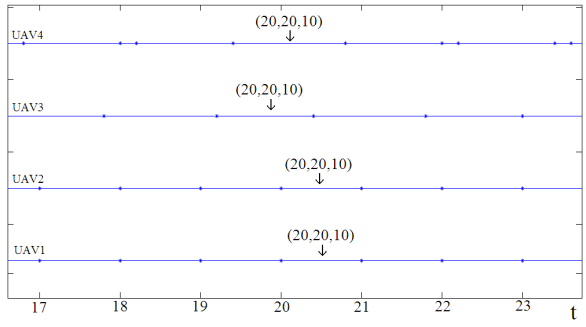

Figura 12. Solapamiento espacial y temporal inicial entre los UAVs del 1 al 4.

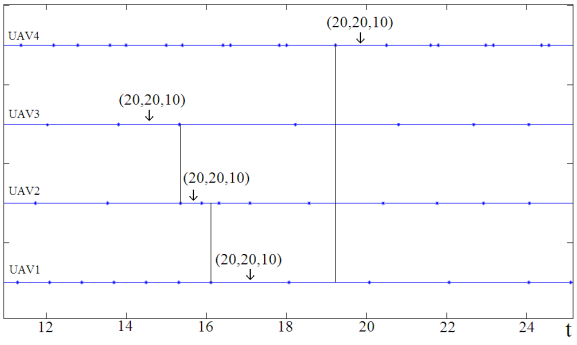

Figura 13. Solución obtenida para el conflicto presentado en la Fig. 12.

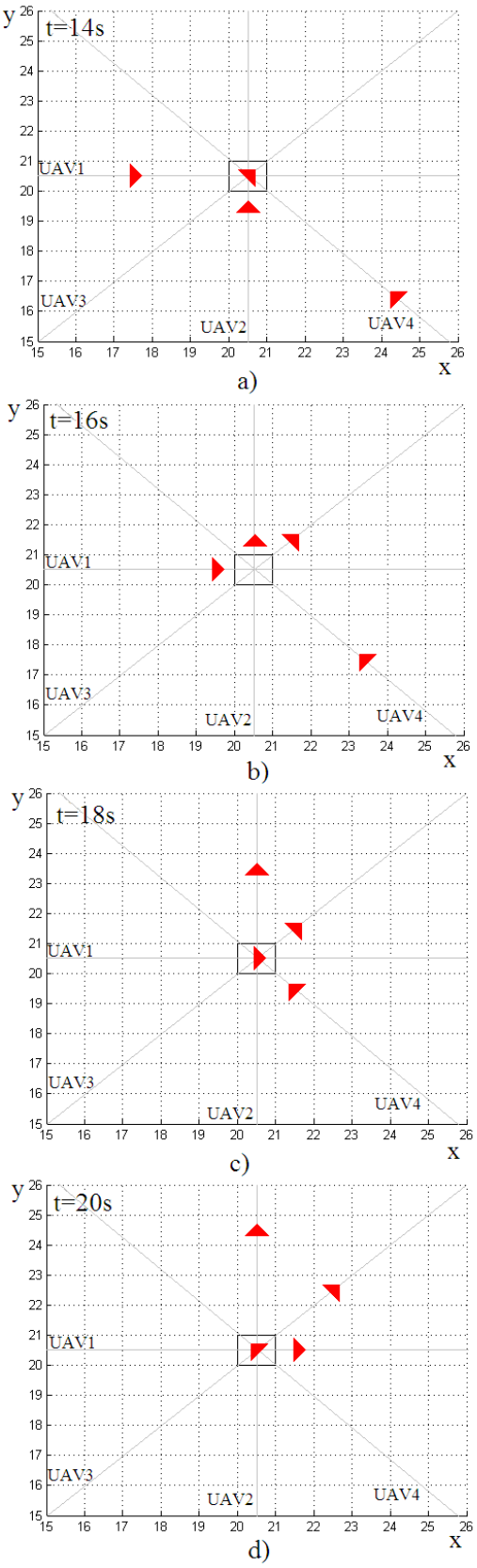

Figura 14. Detalle de la solución al conflicto presentado en la Fig. 12. Posiciones de los UAVs en los instantes: a) t=14s b) t=16s c) t=18s d) t=20s.

Tabla 1. Conflictos y colisiones entre los distintos UAVs en la simulación con 6 vehículos.

Colisiones	Conflictos
(20,20,10) UAV1,UAV2, UAV3,UAV4	(24,20,10) UAV1,UAV6
	(20,24,10) UAV2,UAV6
	(20,10,10) UAV2,UAV5
	(22,22,10) UAV3,UAV6
	(10,10,10) UAV3,UAV5
	(30,10,10) UAV4,UAV5
	(29,10,10) UAV4,UAV5
	(34,10,10) UAV5,UAV6

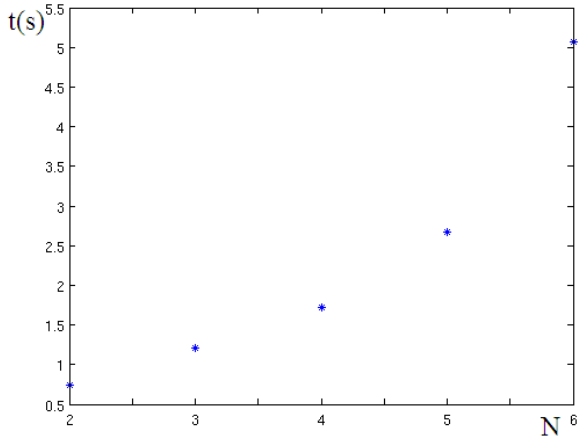

Figura 15. Tiempos de cómputo en función del número de UAVs. Se ha calculado el tiempo de cálculo considerando primero sólo el UAV1 y el UAV2, posteriormente se ha incluido el UAV3, y así sucesivamente hasta llegar a seis UAVs.

posteriormente se ha incluido el UAV3, y así sucesivamente hasta llegar a seis UAVs. Las simulaciones se han llevado a cabo con una CPU de 1.7GHz y 1GB de RAM. Los resultados obtenidos se pueden ver en la Fig. 15. El tiempo crece de forma aproximadamente lineal, si no se considera el caso de 6 UAVs. Esto se debe a que al incluir el UAV6, aparecen 4 nuevas celdas con conflicto, y además 3 de ellos están muy cerca de la celda (20,20,10) donde se produce la colisión cuádruple, con lo que se dificulta el cálculo. A pesar de ello siguen siendo tiempos válidos, ya que resuelve el problema en unos pocos segundos. Además, la Búsqueda Tabú consume la mayor parte de los tiempos mostrados en la Fig. 15. Si solamente se ejecutara la Búsqueda en Árbol, los tiempos estarían comprendidos entre 10 y 20ms.

Si el número total de vehículos del sistema fuese aún más elevado, dado que las colisiones se resuelven localmente, sólo aquellos vehículos que están lo suficientemente cerca de la colisión deberán tenerse en cuenta en su resolución. Esto hará que el tiempo de cálculo siga siendo aceptable a pesar de que se tenga un gran número de UAVs en el sistema, ya que no se resolverían todos simultáneamente.

5. CONCLUSIONES Y TRABAJO FUTURO

En este artículo se ha presentado una nueva estrategia para resolver el problema de resolución de colisiones entre múltiples UAVs cooperativos y no cooperativos (obstáculos móviles), que comparten el espacio aéreo. El objetivo era encontrar una solución en tiempo real modificando las trayectorias de los vehícu-

los lo mínimo posible. Este problema se resolvió mediante dos algoritmos de búsqueda heurísticos, para encontrar soluciones eficientes. Los resultados obtenidos en la simulación presentada son satisfactorios, ya que la solución obtenida difiere poco de las trayectorias que se tenían inicialmente, y el tiempo de ejecución se ajusta a las restricciones temporales del problema. En este artículo se resuelve un problema con 6 UAVs, teniendo cada uno de ellos una trayectoria compuesta por 60 celdas satisfaciéndose las restricciones temporales. Cabe mencionar que las trayectorias de los vehículos no tienen por qué ser rectilíneas para aplicar el método propuesto.

En determinadas situaciones no será posible encontrar una solución cambiando tan solo el perfil de velocidades. En este caso será necesario emplear alguna otra estrategia adicional para modificar la trayectoria. Así por ejemplo, se modificaría la altitud si ello fuera posible (Wollkind, 2004) o se crearía una rotonda (Massink and Francesco, 2001) para resolver las colisiones. Pero una vez que se modifica el camino, los algoritmos presentados en este artículo se podrían usar para calcular un perfil de velocidades, y así obtener la solución deseada.

El método propuesto para evitar las colisiones, puede modificar o dejar invariante las trayectorias de los diferentes vehículos implicados en una colisión. Como trabajo futuro se podría llevar a cabo el diseño de una entidad de nivel superior, que decida cómo debe ser tratado cada vehículo para asegurar una solución cercana a la óptima.

AGRADECIMIENTOS

Este trabajo ha sido parcialmente subvencionado por el Proyecto AWARE FP6 de la Comisión Europea (IST-2006-33579), la Red de Excelencia CONET (INFSO-ICT-224053) y el proyecto SIRE (P06-TEP-01494).

REFERENCIAS

Bichi, A. and L. Pallottino (2000). On optimal cooperative conflict resolution for air traffic management systems. *IEEE Transactions on Intelligent Transportation Systems* **1**(4), 221–231.

Cruz, A., A. Ollero, V. Munoz and A. García-Cerezo (1998). Speed planning method for mobile robots under motion constraints. In: *Intelligent Autonomous Vehicles (IAV)*. pp. 123–128.

Ferrari, C., E. Pagello, M. Voltolina, J. Ota and T. Arai (1997). Multirobot motion coordination using a deliberative approach. In: *Second Euromicro Workshop on Advanced Mobile Robots (EUROBOT '97)*. pp. 96–103.

Fujimori, A. and M. Teramoto (2000). Cooperative collision avoidance between multiple mobile robots. *Journal of Robotic Systems* **17**(3), 347–363.

Gengreau, M. (2002). *An introdution to tabu search*. Departament d'informatique et de recherche operationnelle. Universidad de Montreal.

Glover, F. and M. Laguna (1997). *Tabu search*. Kluwer academic publishers.

Hertz, A., E. Taillard and D. de Werra (1995). A tutorial on tabu search. In: *Proc. of Giornate di Lavoro AIRO'95 (Enterprise Systems: Management of Technological and Organizational Changes)*. Italy. pp. 13–24.

Kant, K. and S. Zucker (1986). Toward efficient trajectory planning: The path-velocity decomposition. *The International Journal of Robotics Research* **5**(3), 72–89.

Massink, M. and N. De Francesco (2001). Modelling free flight with collision avoidance. In: *Proceedings of the Seventh International Conference on Engineering of Complex Computer Systems*. pp. 270–279.

Maza, I. and A. Ollero (2007). *Distributed Autonomous Robotic Systems 6*. Chap. Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms, pp. 221–230. Vol. 6 of *Distributed Autonomous Robotic Systems*. Springer Verlag.

McLain, T. W. and R. W. Beard (2005). Coordination variables, coordination functions, and cooperative timing missions. *Journal of Guidance, Control, and Dynamics* **28**, 150–161.

Munoz, V.F., A. Ollero, M. Prado and A. Simón (1994). Mobile robot trajectory planning with dynamic and kinematic constraints. In: *Proceedings of the IEEE International Conference on Robotics and Automation*. San Diego California. pp. 2802–2807.

Ollero, A. and I. Maza (2007). *Multiple heterogeneous unmanned aerial vehicles*. Springer Tracts on Advanced Robotics. Springer-Verlag.

Owen, E. and L. Montano (2005). Motion planning in dynamic environments using the velocity space. In: *Proceedings of the IEEE International Workshop on Intelligent Robotics and Systems (IROS)*. pp. 2833–2838.

Pallottino, L., V. G. Scordio, E. Frazzoli and A. Bicchi (2007). Decentralized cooperative policy for conflict resolution in multi-vehicle systems. *IEEE Trans. on Robotics and Automation*.

Richards, A. and J. P. How (2002). Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In: *Proceedings of the American Control Conference*. pp. 1936–1941.

Ruz, J.J., O. Arévalo, J.M. de la Cruz and G. Pajares (2006). Using MILP for UAVs trajectory optimization under Radar Detection Risk. In: *Proceedings of the 11th IEEE International Conference on Emerging Technologies and Factory Automation*. pp. 1–4.

Tsubouchi, T. and S. Arimoto (1994). Behavior of a mobile robot navigated by an iterated forecast and planning scheme in the presence of multiple movingobstacles. In: *Proceedings of the IEEE International Conference on Robotics and Automation*. pp. 2470–2475.

Wollkind, S. (2004). Using Multi-Agent Negotiation Techniques for the Autonomuos Resolution of Air Traffic Conflicts. PhD thesis. University of Texas.