ROBÓTICA 2007 - 7th Conference on Mobile Robots and Competitions
Centro Paroquial de Paderne, Albufeira, Portugal
April 27, 2007

Collision avoidance among multiple aerial robots and other non-cooperative aircraft based on velocity planning

Juan José Rebollo, Aníbal Ollero and Iván Maza

***Abstract*— This paper presents a collision avoidance method for multiple UAV sharing the same aerial space with non-cooperative aircraft based on velocity planning. The proposed method finds a safe trajectory, modifying the velocity profile of the different vehicles involved in the collision and considering mobile obstacles. The proposed method applies two steps: Search Tree and Tabu Search. The objective is to find the nearest solution to the initially planned UAVs trajectories while meeting the time constraints on the execution of the algorithms.**

I. INTRODUCTION

Multiple UAVs are cooperatively used to carry out tasks that can not be easily done by a single robot. In this context it appears the collision avoidance problem we address in this paper. The collision avoidance problem can be solved in two different ways. Collision free trajectories can be initially calculated before the vehicles start moving. This method has no significant computation time constraints. On the other hand, initial trajectories would be calculated without taking into account the trajectories of the other vehicles, and the potential collision would be solve in real time once they are detected. In this case computing time plays an important role. This is the problem we address in this paper.

Recently the problem of motion planning for multiple robots has received a great deal of attention. In [12], the problem is written as a linear program subject to mixed integer constraints, known as a mixed-integer linear program (MILP). This can be solved using commercial software written by the Operation Research community. This problem has a significant complexity because of the high number of constraints and because it does not consider mobile obstacles. In [13] a method is proposed to geometrically construct a collision-free trajectory in (x,y,t) space. First, the authors evaluate the position and speed of the mobile obstacles. Assuming that the obstacles speed remain constant, they compute a set of oblique cylinders in (x,y,t) space to be avoided. The problem is then to find a trajectory connecting the initial position to a vertical line representing the goal. This method does not solve a multirobot motion planning problem, in which the trajectory of more than one robot can change to solve collisions. The method in [10] has the same drawback for the multi-robot problem considered in this paper. In this case a new trajectory of a vehicle, which has only mobile or static obstacles in its trajectory, is found in the velocity space.

A different approach is set in [2], where alternative solution paths are obtained by generating small variations of robot motions in space and in time. The Stop & Go strategy or Shape Changing is used to avoid collisions once the new path is found. It assumes that the vehicles have rather simple dynamics, and does not consider mobile obstacles. In [11] a plan is proposed for steering multiple vehicles between assigned independent start and goal configurations and ensuring collision avoidance. All the agents cooperate by following the same traffic rules. They move with constant velocity, a safety area is defined and the velocity of movement of the safety area can be zero. However, this method usually leads to the modification of the paths which could be not needed if the collisions are avoided by simply modifying the velocity.

A geometrical approach is presented in [3]. In this paper this direction angle and the velocity of the mobile robots are used as control variables for navigation and collision avoidance. This method only ensures collision avoidance for two vehicles. A speed planning method with mobile obstacle avoidance was presented in [1]. Mobile obstacles are included as vehicle's motion constraints. This method does not acknowledge the possibility of modifying the trajectory of all the vehicles involved in the collision.

Kant and Zucker [7] proposed the decomposition of the collision avoidance problem into the path planning problem (PPP) and the velocity planning problem (VPP). Once a path has been planned, a velocity profile that avoids collisions in that path is found by means of the proposed VPP method.

A 3-D multirobot motion planning problem for multiple UAVs sharing the space with non-cooperative aircraft (mobile obstacles) is solved in this paper. A new velocity profile for all the different UAVs involved in the collision will be calculated. The paper presents a new heuristic approach that finds suboptimal solutions much faster than optimal methods by exploring a discretized space. The objective is to find a collision-free solution that changes our initial trajectory as little as possible, by changing the velocity profile of the vehicles. Most vehicles have significant dynamics limitations that do not allow them to stop or modify their trajectories as fast as needed. Then, in this paper the dynamic model and physical constraints of the vehicles are considered.

II. PROBLEM FORMULATION

The first step in solving the collision avoidance problem is to detect potential collisions. In this paper, the collision detection algorithm is based on a discretized space. The

This was partially funded by the Information Society Technologies priority of the European Comission and the Spanish Science and Education Ministry.

Juan José Rebollo, Aníbal Ollero and Iván Maza are with the Robotics Control and Vision Group, University of Seville, 41092 Seville, Spain, juanjorebollo@hotmail.com, aollero@cartuja.us.es, imaza@cartuja.us.es.

xyz-space is divided into cubic cells. A trajectory can be described as a sequence of cells with an entrance time and departure time. Therefore, to ensure a collision-free trajectory, only one vehicle can be in each cell at a time. Each UAV knows the trajectories, not as the whole trajectory but as a list of cells, of the other UAVs. This makes it easier to check whether a collision will occur, because each UAV simply has to find temporal overlapping between a cell of its trajectory and a cell that belongs to another UAV trajectory.

The 3-D grid proposed in this paper decreases the data transmission among vehicles, because they do not have to transmit the whole trajectory. This strategy also decrease the time needed to detect potential collisions in methods that use the whole trajectories. The objective is to find time conditions for each cell that give us a free collision trajectory.

In our method it is possible to change the trajectories of all vehicles involved in the collision to find a better solution. In case some vehicle cannot change its trajectory to reach its goal properly it could be considered as a *mobile obstacle* and we would keep its initial trajectory.

Let jkh be the identifier of a cell in the space. $C_{ijkh} = 1$, if vehicle i passes through cell jkh, or $C_{ijkh} = 0$ if it does not. Let tin_{ijkh} be the instant in which vehicle i enters in cell jkh, and $tout_{ijkh}$ the instant vehicle i leaves cell jkh. For $C_{ijkh} = 1$, T_{ijkh}=$[tin_{ijkh}, tout_{ijkh}]$, there is a collision in cell jkh, if:

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

There is a conflict among N vehicles in cell jkh, if:

$$\sum_{i=1}^{M} C_{ijkh} = N \tag{2}$$

being M the total number of vehicles.

The problem is the computation of v_{oi} and T_{ijkh}, so that in each conflict cell:

$$\bigcap_{i:C_{ijkh}=1} T_{ijkh} = \emptyset, \tag{3}$$

considering:

$$\Delta t_{ijkh} = tout_{ijkh} - tin_{ijkh} \in [tmin_{il}, tmax_{il}] \tag{4}$$

and minimizing:

$$J = \sum_{i=0}^{N-1} a(v_{oi} - v_{refoi})^2 + b(\Delta t_i - \Delta t_{refi})^2 \tag{5}$$

where N is the number of cells of the trajectories, v_{refoi} is the initial velocity in cell number i of the reference trajectory, v_{oi} is the initial velocity of the cell number i of the solution trajectory, Δt_{refi} is the time that the reference trajectory takes to pass through cell number i and Δt_i is the time that the solution trajectory takes to pass through the cell number i of the trajectory. In the cost J the parameters a and b weight the reference initial velocities and reference times in cells. The objective of the cost (5) is to find a solution trajectory close to the reference trajectory, which is the trajectory of the vehicles before detecting the potential collision. Minimizing this cost the trajecories change is minimun.

Equation 4 takes into consideration the model of the UAV. The time each vehicle stays in a cell depends on the model of the vehicle. In $tmin_{il}$ and $tmax_{il}$, l indicates a certain conflict, and i points out a certain vehicle. The maximum and minimum time a vehicle stays in a cell depends on the distance the vehicle covers in the cell.

To solve collisions we have to consider the vehicles that are going to crash and all the vehicles whose trajectories cut the trajectories of the vehicles that are going to crash. The algorithm proposed here considers three kinds of vehicles involved in a collision. We call the vehicles that are involved in the potential detected collision *direct involved*. The vehicles whose trajectories are cut by the trajectories of the *direct involved* can be either *indirect involved* cooperative UAVs or *non-cooperative aircraft* also called *mobile obstacles*. The *indirect involved* vehicles are those cooperative vehicles that could be involved in the new collision that appear when solving the collision between *direct involved* vehicles. The trajectories of both direct and indirect cooperative vehicles can be changed. However the problem also involves *non-cooperative aircraft* or *mobile obstacles* which trajectories can not be changed by the collision avoidance system. Therefore, the proposed method changes the trajectories of the *direct* and *indirect involved* UAVs, not the trajectories of the vehicles considered *mobile obstacles*. It is necessary to consider vehicles that did not detect a potential collision originally because if we do not consider them we can find a solution that avoids the initial collision but causes new collisions with another UAVs. Sometimes it is beneficial to consider some cooperative vehicles as *mobile obstacles* because it decreases the information exchanged by those UAVs to solve collisions and the computation time of the algorithms. This strategy is important when a vehicle's task does not allow its trajectory to change.

III. THE PROPOSED COLLISION AVOIDANCE METHOD

The collision avoidance problem is very difficult to solve because of the wide range of possible solutions. We can make the problem easier by considering cells in the space. Cells do not allow us to find an optimal solution, but they make it possible to use fast search algorithms to reach our goal. Now the objective is to find how much time each vehicle stays in each cell of the trajectory. Even for this new problem, it is not possible to find an optimal solution in polynomial time, and it is important to find a fast solution because otherwise the collision may not be avoided. Because of these reasons, in this paper we have developed an heuristic method based on the combination of both the Search Tree Algorithm and the Tabu Search Algorithm. These two heuristic algorithms are used to find a solution to the collision avoidance problem by creating a new velocity profile. The algorithms are based

on the idea of considering some logical rules to find the solution. The Search Tree algorithm finds a collision-free solution that does not employ the cost index (5), but it is the initial solution that the Tabu Search algorithm needs to find the solution to the problem we address in this paper. The algorithms consider the UAV model and the distance travelled by the UAVs in each cell.

III-A. SEARCH TREE ALGORITHM

This algorithm searches for an approximate solution, by assuming that each vehicle involved in the collision goes as fast as possible. This algorithm is based on the idea that there is no collision-free solution if there is a vehicle traveling as fast as possible and another, as slow as possible, comes from behind and collides with the first.

Let us define the *order of passing* as the order in which the vehicles pass through a conflict. A conflict with N vehicles involved, has $N!$ different orders.

The algorithm explores the different *orders of passing* in each conflict until a solution is found. First of all, it looks for a solution exploring the most logical *order of passing*–in which the vehicle that has to travel less distance to arrive to the conflict would pass first. For each order, it is possible to determine if a solution exists in short computational time. If there is a solution, it is more probable to find that solution in the first order we check. If there is no solution for a certain *order of passing*, the algorithm permutes the *order of passing* of the conflict by changing the order of the vehicles that have to travel more distance to arrive to the conflict.

If the problem has m conflicts with N_i vehicles involved in the i conflict, there are $N_0!N_1!...N_{m-1}!$ orders to check. When a certain order is explored and there is no solution, the algorithm permutes the order of one of the conflicts. First it permutes the i conflict which cost J_i defined as

$$J_i = \mu_i - \sigma_i \tag{6}$$

is larger, where μ_i and σ_i are the mean and the standard deviation of the distance that each vehicle involved has to travel to arrive to a certain conflict.

The algorithm permutes first the order of the vehicles involved in conflicts that are farther from the beginning of the trajectory, because when a conflict is near the beginning of the trajectory the vehicles that have to travel less distance to arrive to it have a greater chance to pass first in the solution of the problem. However in a conflict that is farther from the beginning of the trajectory, another conflict closer to the beginning could affect the search criterion defined above (the vehicle which has to travel less distance to arrive to the conflict passes first). Differences in the distance the vehicles travel to arrive to a certain conflict make the initial *order of passing* more suitable. The term σ_i in (6) copes with this concept.

Each UAV trajectory will be associated with a tree. When we go from one node to another we are passing through a cell. Then if we go from the first node to the last one, we will cover the whole trajectory of the UAV associated with

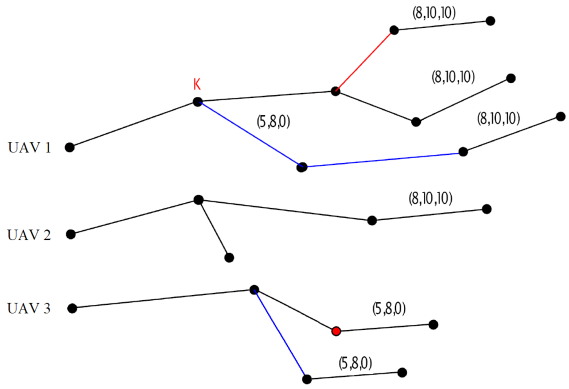

Fig. 1. New branches

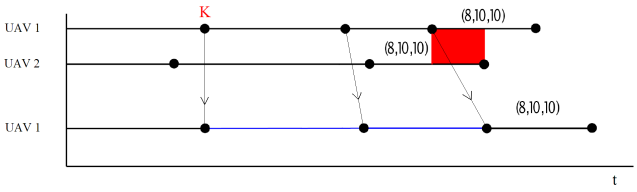

Fig. 2. Temporal diagram

that tree. A solution is found if all the trees are completely built. The distance between two consecutive nodes is directly related to the time the UAV spends in the associated cell. The algorithm 1 shows how the trees are built or how the solution to our problem is found.

Basically, trees grow by calculating the amount of time every vehicle stays in their trajectory cells traveling at top speed, until a conflict cell is found. When this conflict is detected, an associated branch could be created or not depending on the tree's turn. That turn corresponds to the *order of passing* the tree is checking in this iteration. Therefore, it is a tree's turn if the branches of other trees associated to the same conflict cell and associated to UAVs that have to pass through the conflict before were already created. If it is the tree's turn, it checks if a collision occurs in the conflict by checking if there is temporal overlapping among the times associated to other trees branches associated to the same conflict. If a collision appears, the algorithm has to change the velocity in the previous cells by rebuilding branches and the UAV associated to this tree can not travel at its top speed before this conflict.

A bifurcation appears in a tree each time it has to rebuild a branch because a collision was found. Fig. 1 shows how collisions that appear in the algorithm are solved. The algorithm backtracks and creates new branches. Those that are in red are not valid because we would need to change additional branches lengths to avoid collision. The blue branches are valid, they allow us to arrive later to cell (8,10,10). Fig. 2 shows us that there was a temporal overlapping between vehicles 1 and 2 in cell (8,10,10) which indicates a collision. After creating new branches that collision disappears.

The changes made by the Search Tree Algorithm to avoid temporal overlapping could affect another trees. Fig. 1 shows that the tree associated with UAV3 has to rebuild its branches, because UAV1 passes first through cell (5,8,0) and UAV3 would collide with UAV1 if the tree associated to UAV3 does not rebuild any branches before the red node. It has to recalculate the time that UAV3 remains in the cells previous to (5,8,0), because the condition that has to be fulfilled in the conflict has changed.

Algorithm 1 Search Tree Algorithm

```
while there is no solution or all the orders of passing have not been explored do
  while the trees are not complete do
    for each tree, if it is not complete do
      Advance at the maximum speed V_max up to the next conflict, creating the associated branches
      if the end was not reached then
        if it is a turn in the conflict then
          Pass through the conflict, creating the new associated branch
          if there is a collision then
            Go back and create new branches that solve the collision. Another tree may have to backtrack as well to ensure collision-free trajectory
            If the beginning of the tree is reached upon backtracking, then there is no solution for the order of passing being considered.
          end if
        end if
      end if
    end for
  end while
end while
```

If the Search Tree algorithm does not find a solution for a certain *order of passing*, all the trees start from the beginning again using the next *order of passing* set by the search criterion we defined in (6).

Direct and *indirect involved* UAVs build the tree in the same way. However, the trees for mobile obstacles are built from the beginning and they are not changed because their trajectories can not be changed.

The Search Tree Algorithm allows us to find a solution in a reduced time, as compared with methods that solve the problem of collision avoidance without considering a cell-divided space method.

III-B. TABU SEARCH

The Search Tree Algorithm finds solutions to our problem but it does not consider the cost index (5). The Tabu Search Algorithm[5] (TS) modifies the solution that the search tree found, by minimizing this cost. Tabu search enhances the performance of a local search method by using memory structures to avoid local minima. Several elements have to be defined for the right performance of the Tabu Search:

- *Initial solution x*: The solution obtained by the Search Tree Algorithm will be the initial solution. That solution is formed by a series of time variables direct related with the time the vehicles stays in each cell. Each time variable is the time that the vehicle spends in a particular group of cells. By grouping cells, the number of variables is reduced which allows us to find the desired solution faster.
- *Neighborhood N(x)*: In each iteration, the TS algorithm explores the neighborhood N(x), and choose the best solution. The neighborhood consist of all the solutions obtained by increasing Δt, one of the time variables of x, which have no collisions and meet dynamic model constraints.
- *Tabu list*: The TS uses memory to remember the last explored solutions and avoid local minimums. According to [6], 7 has been shown to be a good value for the size for the list. The Tabu solutions are all the solutions which distance to the solution in the list is less than $\Delta t/2$ (see Fig. 3).
- *Aspiration criterion*: A solution that is in the neighborhood is valid if it is a non-Tabu solution. But some solutions have to be considered even if they are a Tabu solution i.e. solutions that are better than the best solution found thus far [4].
- *End criterion*: A condition that stops the TS algorithm. We consider a maximum number of iterations without decreasing the current optimal value of the cost index.

Algorithm 2 shows TS pseudocode.

Algorithm 2 TS pseudocode

```
Choose x ∈ X and consider x_op = x
while Ending criterion is not true do
  Search x' ∈ N(x) minimizing f(x) being x' a non-Tabu solution or meeting the aspiration criteria
  if f(x') < f(x_op) then x_op = x' then
    Include x' in the Tabu list
  end if
end while
```

This algorithm changes the time that the direct and *indirect involved* vehicles spend in each cell. Fig. 3 shows how the solution moves from the initial solution that the search Tree found to another solution in which cost index value is smaller and which is thus a better solution to the problem we want to solve.

IV. SIMULATIONS

In the simulation we present in this paper, three UAVs are considered. UAV1 and UAV2 are sweeping a region, and UAV3 is a non-cooperative teleoperated UAV, which will be considered as a *mobile obstacle*. Fig. 4 shows the xy-projection of the whole paths of the three UAVs. The dashed area in Fig. 4, in which the potential collision we are solving is detected and solved, is shown magnified in xyz-space in Fig. 5.

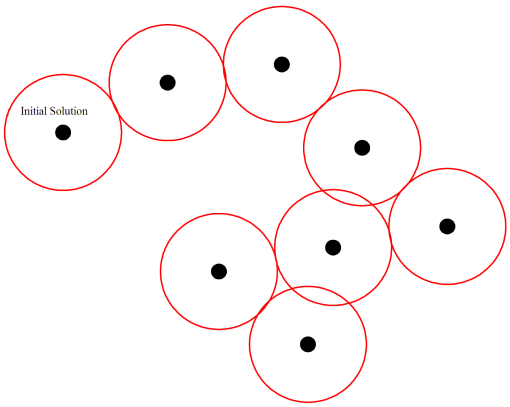

Fig. 3. Improving Search Tree solution

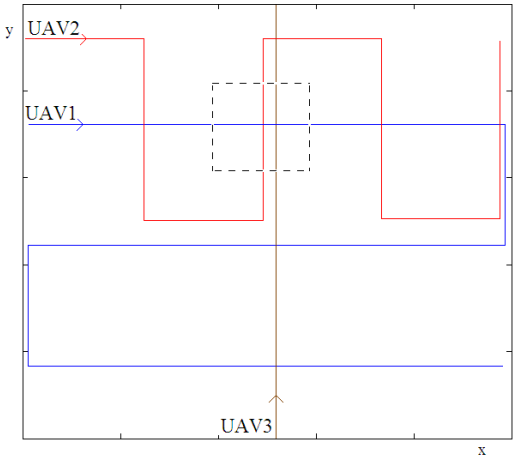

Fig. 4. xy-projection of the xyz-paths

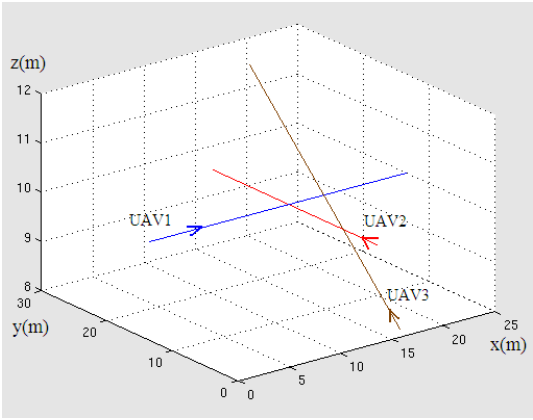

Fig. 5. Paths in which the problem is solved

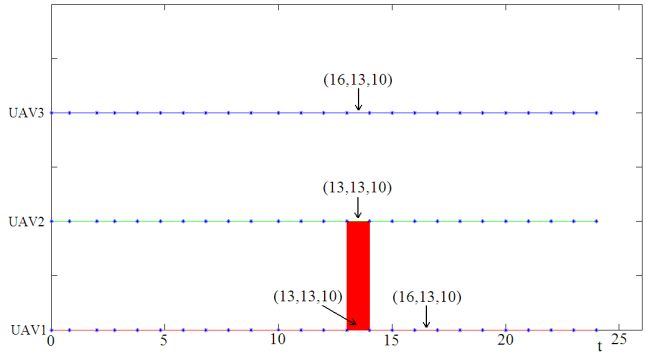

Fig. 6. Temporal diagram of the initial trajectories

IV-A. UAV model

The UAVs move along their trajectories according to the model [9]:

$$\begin{aligned} \dot{x}_i &= v_i cos(\dot{\psi}_i) \\ \dot{y}_i &= v_i sin(\dot{\psi}_i) \\ \dot{\psi}_i &= \alpha_\psi(\psi^c - \psi) \\ \dot{v}_i &= \alpha_v(v^c - v) \\ \ddot{h}_i &= -\alpha_{\dot{h}} h_i + \alpha_h(h_i^c - h_i) \end{aligned} \tag{7}$$

where α_ψ, α_v, α_h and $\alpha_{\dot{h}}$ are known constants that depend on the implementation of the UAV. Regarding the heading rate and velocity, we consider:

$$\begin{aligned} -c < \dot{\psi}_i < c \\ v_{min} < v_i < v_{max} \end{aligned} \tag{8}$$

where c, v_{min} and v_{max} are positive constants that depend on the dynamic capability of the particular UAV.

The maximum and minimum times that an UAV can stay in a cell are calculated by using the above model. For that calculus, we need the distance each vehicles travels in a certain cell. As we saw in (4), the maximum and minimum time a vehicle can stay in a cell depends on the model and on the trajectory in the cell.

Both algorithms presented in this paper work using the dynamic model of the UAVs. The Search Tree calculates the time associated to each branch considering the dynamic model. Then, in the TS each new neighbor has to be a solution that meets the dynamic model.

IV-B. Simulation results

In this section we present the results of proposed method. UAV1 and UAV2 are considered *direct involved* vehicles and UAV3 as a *mobile obstacle*. Fig. 6 is a temporal diagram of the original trajectories, in which each point signifies a transition into the following cell in the trajectory. There are two conflicts one in cell (13,13,10) between UAV1 and UAV2 and another in cell (16,13,10) between UAV1 and UAV3. Our objective is to find a collision-free solution as similar to the initial trajectories as possible.

Fig. 7 shows the results obtained by the Search Tree Algorithm. We can see that the trajectory of the UAV3

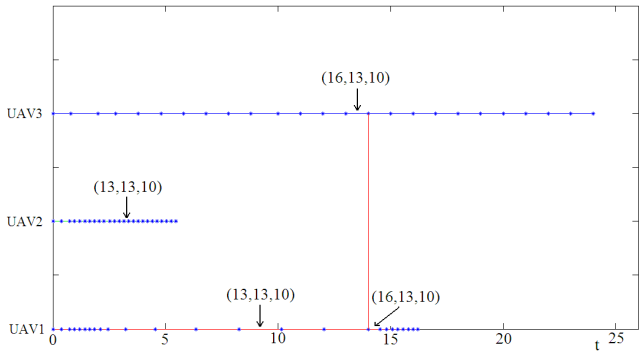

Fig. 7. Search Tree Algorithm solution

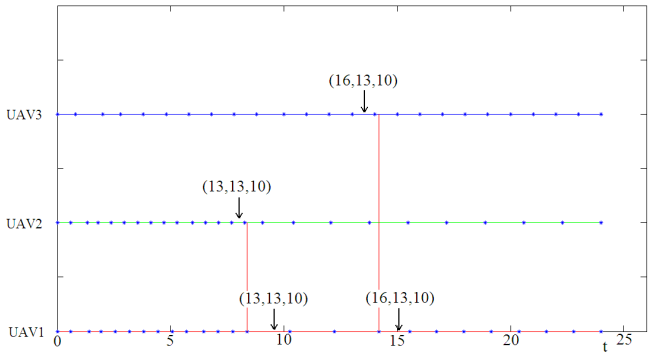

Fig. 8. Search Tree Algorithm and Tabu Search solution

did not change because it was considered as a *mobile obstacle*. However, UAV1 and UAV2 went as fast as possible considering that collisions in cells with conflict have to be avoided. UAV1 goes slower than UAV2 because the conflict between UAV1 and UAV3 set a temporal constraint in cell (16,13,10) that it should be satisfied. UAV3 is nearer to the conflict than UAV1, so the algorithm first checks the solution where UAV3 passes first. Therefore, UAV1 passes behind UAV3 and has a temporal constraint in cell (16,13,10).

Now if we execute the Tabu Search algorithm for the solution we found above, we have the solution that Fig. 8 shows. This solution is almost equal to the initial trajectories we saw in Fig. 6, but there are no collisions.

A problem involving three UAVs where each UAV has a 30-cell trajectory, and $\Delta t = 0{,}1$ seconds was solved in less than a second, which meets the temporal constraints of the problem. The computation time does not depend on shape of the path, because each path is a sequence of cells and the algorithm deals with them in the same way.

V. CONCLUSIONS AND FUTURE WORK

In this paper, a new strategy to solve the collision avoidance problem between multiple cooperative UAVs and non-cooperative (mobile obstacles) aircraft sharing the same space has been presented. The objective was to find a solution by changing in real time the trajectories of the vehicles as little as possible. This complex problem was solved by using two heuristic search algorithms to find efficient solutions. The results obtained were satisfactory because the solution found allowed the initial trajectories to remain nearly unchanged and the execution time met our time restrictions. A problem involving 3 UAVs, having a 30-cell trajectory each UAV was solved, meeting the temporal constraints.

In situations where it is not possible to find a solution without changing the path, some strategy has to be used to change it. We could modify the altitude [14] or create a roundabout [8] to solve the collisions. But once the new path is obtained, the algorithm presented in this paper could be used to find the velocity profile that give us a solution according to our requirements.

The proposed collision avoidance method can change or hold constant the trajectories of the different vehicles involved in a collision. Future work will include the design of a higher level entity that will decide how each vehicle should be treated to ensure a near optimal solution.

VI. ACKNOWLEDGMENTS

This work has been partially funded by the AWARE FP6 Project of the European Commission (IST-2006-33579) and the Spanish AEROSENS project (DPI2005-02293) of the Spanish National R&D Programme.

REFERENCES

[1] A. Cruz, A. Ollero, V. Muñoz, and A. García-Cerezo. Speed planning method for mobile robots under motion constraints. In *Intelligent Autonomous Vehicles (IAV)*, pages 123–128, March 1998.

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

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

[4] M. Gengreau. *An introdution to tabu search*. Département dínformatique et de recherche opérationnelle, Université de Montréal.

[5] F. Glover and M. Laguna. *"Tabu search"*. Kluwer academic publishers, 1997.

[6] A. Hertz, E. Taillard, and D. de Werra. *A tutorial on Tabu Search*. Departament of mathematics, University of Montreal.

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

[8] M. Massink and N. De Francesco. Modelling free flight with collision avoidance. In *Proceedings of the Seventh International Conference on Engineering of Complex Computer Systems*, pages 270–279, 2001.

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

[10] E. Owen and L. Montano. Motion planning in dynamic environments using the velocity space. In *Intelligent Robots and Systems, 2005. (IROS 2005)*, pages 2833–2838, August 2005.

[11] L. Pallottino, V. G. Scordio, E. Frazzoli, and A. Bicchi. Decentralized cooperative policy for conflict resolution in multi-vehicle systems. 2006.

[12] A. Richards and J. P. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. 2002.

[13] T. Tsubouchi and S. Arimoto. Behavior of a mobile robot navigated by an iterated forecast and planning scheme in the presence of multiple moving obstacles. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 2470–2475, 1994.

[14] S. Wollkind. *Using Multi-Agent Negotiation Techniques for the Autonomuos Resolution of Air Traffic Conflicts*. PhD thesis, University of Texas, 2004.