Obstacle avoidance in optimal switching of a formation geometry ⋆

Fernando A.C.C. Fontes * **Dalila B.M.M. Fontes** **
Amélia C.D. Caldeira ***

* *Faculdade de Engenharia and ISR Porto, Universidade do Porto, 4200-465 Porto, Portugal (e-mail: faf@fe.up.pt).*
** *Faculdade de Economia and LIAAD-INESC Porto L.A., Universidade do Porto, 4200-464 Porto, Portugal (e-mail: fontes@fep.up.pt).*
*** *Departamento de Matemática, Instituto Superior de Engenharia do Porto, 4200-072 Porto, Portugal (e-mail: acd@isep.ipp.pt).*

Abstract: We address the problem of dynamically switching the geometry of a formation of a number of undistinguishable agents, while avoiding collisions among agents and with external obstacles. The need to switch formation geometry arises in situations when mission requirements change or there are obstacles or boundaries along the path for which the current geometry is inadequate. Here we propose a strategy to determine which agent should go to each of the new target positions, avoiding collisions among agents and assuming no agent communication. In addition, in order to avoid obstacles, each agent can also modify its path by changing its curvature, which is a main distinguishing feature from previous work. Among all possible solutions we seek one that minimizes the total formation switching time, i.e. that minimizes the maximum time required by all agents to reach their positions in the new formation geometry. We describe an algorithm based on dynamic programming to solve this problem. (Copyright © IFAC Controlo 2012).

Keywords: Autonomous agents, optimization, dynamic programming, vehicle formations, formation geometry, formation switching, collision avoidance, obstacle avoidance.

1. INTRODUCTION

Consider the problem of switching the geometry of a formation of undistinguishable vehicles by minimizing some performance criterion. Given the initial positions and a set of final desirable positions, the questions addressed are:

(1) Which vehicle should go to a specific final position?
(2) How to avoid collision between the vehicles?
(3) Which should be the traveling velocities of each vehicle between the initial and final positions?

Each vehicle can also modify its path, from initial to final position, by changing its curvature, in order to avoid obstacles. In this work, we are particularly interested in exploiting this last possibility.

The performance criterion used in the example explored is to minimize the maximum traveling time, that is, we seek the allocation that minimizes the total formation switching time, but the method developed - based on dynamic programming - is sufficiently general to accommodate many different criteria.

The specific problem of switching the geometry of a formation arises in many cooperative vehicles missions, due to the need to adapt to environmental changes or to adapt to new tasks. An example of the first type is when a formation has to go through a narrow passage, or deviate from obstacles, and must reconfigure to a new geometry (see Figure 1). Examples of adaption to new tasks arise in robot soccer teams: when a team is in an attack formation and looses the ball, it should switch to a defense formation more appropriate to the new task, (see Lau et al. (2009)).

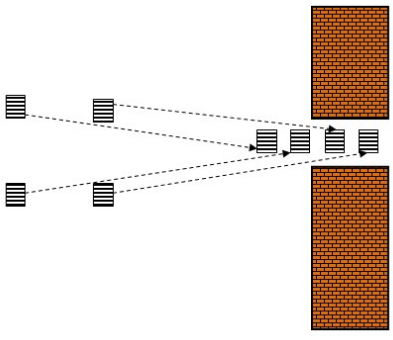

Fig. 1. Reconfiguration of a formation to avoid obstacles.

Another example is in the detection and containment of a chemical spillage, the geometry of the formation for the initial task of surveillance, should change after detection occurs, switching to a formation more appropriate to determine the perimeter of the spill.

⋆ Research supported by FCT, FEDER & COMPETE through Projects PTDC/EEA-CRO/100692/2008 and PTDC/EEA-CRO/116014/2009.

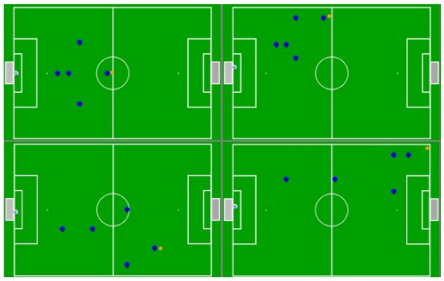

Fig. 2. Different formation of soccer robots used for different ball position (from Lau et al. (2009)).

Research in coordination and control of teams of several vehicles (that may be robots, ground, air or underwater vehicles) has been growing fast in the past few years. Application areas include unmanned aerial vehicles, autonomous underwater vehicles, automated highway systems and mobile robotics. While each of these application areas poses its own unique challenges, several common threads can be found. In most cases, the vehicles are coupled through the task they are trying to accomplish, but are otherwise dynamically decoupled, meaning that the motion of one does not directly affect the others. For a survey in cooperative control of multiple vehicles systems see, for example, the work Murray (2007). Regarding research on the optimal formation switching problem, it is not abundant, although it has been addressed by some authors. See e.g. Desai et al. (2001), Hu and Sastry (2001), Yamagishi (2004), Rasmussen et al. (2004)), Rasmussen and Shima (2006)),Schumacher et al. (2002, 2003)), and Jin et al. (2006)). None of these works address velocity nor curvature issues.

The problem of formation switching has also been addressed in Fontes and Fontes (2008) and Fontes and Fontes (2010) using dynamic programming. The possible use of different velocities for each vehicle was addressed in Fontes et al. (2012). (This last reference provides a discussion on the works cited.)

The possibility of slowing down some of the vehicles might, as we will show, achieve better solutions while avoiding collision between vehicles. Here we use a dynamic programming approach to solve the problem of formation switching with collision avoidance, vehicles velocities selection, and path curvature selection. The main distinguishing feature of this work from previous is precisely the fact that each vehicle can also modify its path by changing its curvature.

The formation switching performance is given by the time required for all vehicles to reach their new position, which is given by the maximum traveling time amongst individual vehicle traveling times. Since we want to minimize the time required for all vehicles to reach their new position, we have to solve a min-max problem. However, our methodology can be used with any separable performance function. The problem addressed here should be seen as a component of a framework for multi-vehicle coordination, incorporating also the trajectory control component, that allows to maintain or change formation while following a specified path in order to perform cooperative tasks.

2. THE PROBLEM

In our problem a team of N identical vehicles has to switch from their current formation to some other formation, with collision avoidance. To address collision among agents, we impose that the trajectories of the vehicles must satisfy the separation constraint that at any time the distance between any two of them is at least ε, for some positive ε. To avoid collision with external obstacles, each vehicle can also modify its path by changing its curvature.

Regarding the new formation, it can be either a pre-specified formation or a formation to be defined according to the information collected by the vehicles. In both cases, we do a pre-processing analysis that allows us to come up with the desired locations for the next formation.

This problem can be restated as the problem of allocating to each new position exactly one of the vehicles, located in the old positions, and determine each vehicle velocity. From all the possible solutions we are only interested in the ones where vehicle and obstacle collision is prevented. Among these, we want to find one that minimizes the time required for all vehicles to move to the target positions, that is an allocation which has the least maximum individual vehicle traveling time.

To formally define the problem, consider a set of N vehicles moving in a space $\mathbb{R}^d$, so that at time t, vehicle i has position $\mathbf{q}_i(t)$ in $\mathbb{R}^d$ (we will refer to $\mathbf{q}_i(t) = (x_i(t), y_i(t))$ when our space is the plane $\mathbb{R}^2$). The position of all vehicles is defined by the N-tuple

$$Q(t) = [\mathbf{q}_i(t)]_{i=1}^N$$

in $\mathbb{R}^{d\times N}$. We assume that each vehicle is holonomic and that we are able to choose its velocity, so that its kinematic model is a simple integrator

$$\dot{\mathbf{q}}_i(t) = \overline{v}_i(t) \quad \text{a.e. } t \in \mathbb{R}^+.$$

The initial positions at time $t = 0$ are known and given by

$$A = [a_i]_{i=1}^N = Q(0).$$

Suppose a set of M (with $M \geq N$) final positions in $\mathbb{R}^d$ is specified as

$$F = \{f_1, f_2, ..., f_M\}.$$

The problem is to find an assignment between the N vehicles and N of the M final positions in F. That is, we want to find a N-tuple $B = [b_i]_{i=1}^N$ of different elements of F, such that at some time $T > 0$,

$$Q(T) = B$$

and all $b_i \in F$, with $b_i \neq b_k$.

There are

$$\binom{M}{N} \cdot N!$$

such N-tuples (the permutations of a set of N elements chosen from a set of M elements) and we want to find a procedure to choose an N-tuple minimizing a certain criterion that is more efficient than total enumeration.

The criterion to be minimized can be very general since the procedure developed is based on dynamic programming which is able to deal with general cost functions.

Examples can be:

- minimizing the total distance traveled by the vehicles
  $Minimize \quad \sum_{i=1}^{N} \|b_i - a_i\|,$
- or, the total traveling time
  $Minimize \quad \sum_{i=1}^{N} \|b_i - a_i\| / \|\overline{v}_i\|,$
- or, the maximum traveling time
  $Minimize \quad \max_{i=1,...,N} \|b_i - a_i\| / \|\overline{v}_i\|,$

We are also interested in selecting the traveling velocities of each vehicle. Assuming constant velocities, these are given by

$$\overline{v}_i(t) = \overline{v}_i = v_i \frac{b_i - a_i}{\|b_i - a_i\|},$$

where the constant speeds are selected from a discrete set

$$\Upsilon = \{V_{\min}, ..., V_{\max}\}.$$

Moreover, we are also interested in avoiding collision between vehicles. We say that two vehicles i, k (with $i \neq k$), do not collide if their trajectories maintain a certain distance apart, at least ε, at all times. The non-collision conditions is

$$\|\mathbf{q}_i(t) - \mathbf{q}_k(t)\| \geq \varepsilon, \qquad \forall t \in [0, T], \tag{1}$$

where the trajectory, in the linear case, is given by

$$\mathbf{q}_i(t) = a_i + \overline{v}_i(t)\ t, \qquad t \in [0, T].$$

We can then define a logic-valued function c (see Figure 3) as

$$c(a_i, v_i, b_i, a_k, v_k, b_k) = \begin{cases} 1 & \text{if collision between } i \text{ and } k, \\ 0 & \text{otherwise.} \end{cases}$$

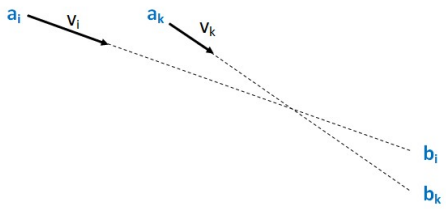

Fig. 3. Linear trajectories of two vehicles.

With these considerations, the problem (in the case of minimizing the maximum traveling time) can be formulated as follows

$$\min_{b_1,...,b_N,v_1,...,v_N} \max_{i=1,...,N} \quad \|b_i - a_i\| \ / \ v_i,$$

Subject to

$$\begin{array}{ll} b_i \in F, & \forall i, \\ b_i \neq b_k, & \forall i, k \ \text{ with } \ i \neq k, \\ v_i \in \Upsilon, & \forall i, \\ c(a_i, v_i, b_i, a_k, v_k, b_k) = 0, & \forall i, k \ \text{ with } \ i \neq k. \end{array}$$

Instead of using the set F of d-tuples, we can define a set $J = \{1, 2, ..., M\}$ of indexes to such d-tuples, and also a set $I = \{1, 2, ...N\}$ of indexes to the vehicles. Let j_i in J be the target position for vehicle i, that is, $b_i = f_{j_i}$. Define also the distances $d_{ij} = \|f_j - a_i\|$ which can be pre-computed for all $i \in I$ and $j \in J$. Redefining, without changing the notation, the function c to take as arguments the indexes to the vehicle positions instead of the positions, i.e.

$$c(a_i, v_i, f_{j_i}, a_k, v_k, f_{j_k}).$$

is simply represented as

$$c(a_i, v_i, j_i, a_k, v_k, j_k).$$

The problem can be reformulated into the form

$$\min_{j_1,...,j_N,v_1,...,v_N} \max_{i=1,...,N} \ d_{ij} \ / \ v_i,$$

Subject to

$$\begin{array}{ll} j_i \in J, & \forall i \in I, \\ j_i \neq j_k, & \forall i, k \in I \ \text{ with } \ i \neq k, \\ v_i \in \Upsilon, & \forall i \in I, \\ c(a_i, v_i, j_i, a_k, v_k, j_k) = 0, & \forall i, k \in I \ \text{ with } \ i \neq k. \end{array}$$

We are finally in position to consider the full problem with obstacle avoidance. Consider that the vehicle a is moving in space $\mathbb{R}^2$ and travels at velocity v_a to position b that is reached in time T_a. Suppose that there is an obstacle, and if the vehicle a travels straight ahead collides with the obstacle, then, the vehicle a should circumvents the obstacle on the top or on the bottom (see Figure 4).

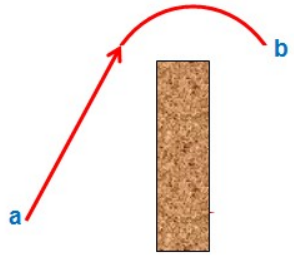

Fig. 4. Vehicle a circumvents the obstacle on top.

Now, the problem can be formulated in a similar way where $d_{ij}(p_i)$ is the length of the path from i to j using path p_i and c is also dependent on the path p_i.

$$\min_{j_1,...,j_N,v_1,...,v_N,p_1,...,p_N} \max_{i=1,...,N} \ d_{ij}(p_i) \ / \ v_i$$

Subject to

$$\begin{array}{ll} j_i \in J, & \forall i \in I, \\ j_i \neq j_k, & \forall i, k \in I \ \text{ with } \ i \neq k, \\ v_i \in \Upsilon, & \forall i \in I, \\ p_i \in P_i(j), & \forall i \in I \\ c(a_i, v_i, j_i, a_k, v_k, j_k, p_i, p_k) = 0, & \forall i, k \in I \ \text{ with } \ i \neq k. \end{array}$$

where $P_i(j)$ is the set of all paths from i to j.

3. DYNAMIC PROGRAMMING FORMULATION

Dynamic Programming (DP) is an effective method to solve combinatorial problems of a sequential nature. It provides a framework for decomposing an optimization problem into a nested family of subproblems. This nested structure suggests a recursive approach for solving the original problem using the solution to some subproblems. The recursion expresses an intuitive *principle of optimality* Bellman (1957) for sequential decision processes, that is, once we have reached a particular state, a necessary condition for optimality is that the remaining decisions must be chosen optimally with respect to that state.

3.1 Dynamic programming recursion for the simplest problem

We start by deriving a DP formulation for a simplified version of problem: where collision is not considered and different velocities are not selected Fontes and Fontes (2008). The collision avoidance, the selection of velocities for each vehicle, and the changing of the curvature path of each vehicle are introduced later.

Consider that there are N vehicles $i = 1, 2, ..., N$ to be relocated from known initial location coordinates to a target locations indexed by set J. We want to allocate exactly one of the vehicles to each position in the new formation. In our model a stage i contains all states S such that $|S| \geq i$, meaning that i vehicles have been allocated to the targets in S. The DP model has N stages, with a transition occurring from a stage $i-1$ to a stage i, when a decision is made about the allocation of vehicle i.

Define $f(i, S)$ to be the value of the best allocation of vehicles $1, 2, ..., i$ to i targets in set S, that is, the allocation requiring the least maximum time the vehicles take to go to their new positions. Such value is found by determining the least maximum vehicle traveling time between its current position and its target position. For each vehicle i, the traveling time to the target position j is given by

$$d_{ij} \ / \ v_i.$$

By the previous definition, the minimum traveling time of the $i-1$ vehicles to the target positions in set $S\backslash\{j\}$ is given by $f(i-1, S\backslash\{j\})$. From the above, the minimum traveling time of all i vehicles to the target positions in S they are assigned to, given that vehicle i travels at velocity v_i, without vehicle collisions, is obtained by examining all possible target locations $j \in S$ (see Figure 5).

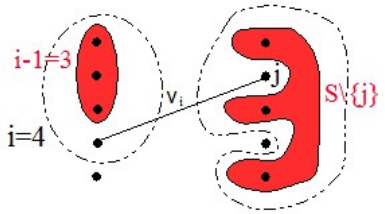

Fig. 5. Dynamic Programming Recursion with N = 5 and stage i = 4.

The *dynamic programming recursion* is then defined as

$$f(i, S) = \min\{d_{ij} \ / \ v_i \ \vee \ f(i-1, S\backslash\{j\})\}. \qquad (2)$$

where $X \vee Y$ denotes the maximum between X and Y.

The *initial conditions* for the above recursion are provided by

$$f(1, S) = \min\{d_{1j} \ / \ v_1\}, \qquad \forall S \subseteq J \qquad (3)$$

and all other states are initialized as not yet computed.

Hence, the optimal value for the performance measure, that is, the minimum traveling time needed for all N vehicles to assume their new positions in J, is given by

$$f(N, J). \qquad (4)$$

3.2 Considering collision avoidance and velocities selection

Recall function c for which $c(i, v_i, j, a, v_a, b)$ takes value 1 if there is collision between pair of vehicles i and a traveling to positions j and b with velocities v_i and v_a, respectively, and takes value 0 otherwise. To analyze if the vehicle traveling through a newly defined trajectory collides with any vehicle traveling through previously determined trajectories, we define a recursive function. This function checks the satisfaction of the collision condition, given by equation (1), in turn, between the vehicle which had the trajectory defined last and each of the vehicles for which trajectory decisions have already been made.

So, if the vehicles i and a go straight ahead, i.e., the path for both will be straight ahead the collision between these two vehicles occurs if the following condition is satisfied:

$$\|(x_i, y_i) + v_i \frac{(x_j, y_j) - (x_i, y_i)}{\|(x_j, y_j) - (x_i, y_i)\|} t - \left[(x_a, y_a) + v_a \frac{(x_b, y_b) - (x_a, y_a)}{\|(x_b, y_b) - (x_a, y_a)\|} t\right]\| < \varepsilon$$

for some $t \in [0, \min\{T_i, T_a\}]$ (see Fontes and Fontes (2010)).

We note that by trajectory we understand not only the path between the initial and final positions but also a timing law and an implicitly defined velocity.

Consider that we are in state (i, S) and that we are assigning vehicle i to target j. Further let v_{i-1} be the traveling velocity for vehicle $i-1$. Since we are solving state (i, S) we need state $(i-1, S\backslash\{j\})$, which has already been computed (if this is not the case, then we must compute it first). In order to find out if this new assignment is possible, we need to check if at any point in time vehicle i, traveling with velocity v_i, will collide with any of the vehicles $1, 2, ..., i-1$ for which we have already determined the target assignment and traveling velocities.

Let us define a recursive function

$$C(i, v_i, j, k, V, S)$$

that assumes the value one if a collision occurs between vehicle i traveling with velocity v_i to j and any of the vehicles $1, 2, ...k$, with $k < i$, traveling to their targets, in set S, with their respective velocities $V = [v_1 \ v_2 \ ... \ v_k]$ and assumes the value zero if no such collisions occurs. This function works in the following way.

(1) first it verifies $c(i, v_i, j, k, v_k, Be_j)$, that is, it verifies if there is collision between trajectory $i \longrightarrow j$ at velocity v_i and trajectory $k \longrightarrow Be_j$ at velocity v_k, where Be_j is the optimal target for vehicle k when targets in set $S\backslash\{j\}$ are available for vehicles $1, 2, ..., k$. If this is the case it returns the value 1;
(2) Otherwise, if they do not collide, it verifies if trajectory $i \longrightarrow j$ at velocity v_i collides with any of the remaining vehicles. That is, it calls the collision function $C(i, v_i, j, k-1, V', S')$, where $S' = S\backslash\{Be_j\}$ and $V = [V' \ v_k]$.

The collision recursion is therefore written as:

$$C(i, v_i, j, k, V, S) = \{c(i, v_i, j, k, v_k, Be_j) \vee \vee \ C(i, v_i, j, k-1, V', S')\} \qquad (5)$$

where

$$Be_j = Best_j\left(k, V', S'\right),$$

and

$$V = \left[V'\ v_k\right], \qquad S' = S \backslash \{j\}.$$

The initial conditions for recursion (5) are provided by

$$C\left(i, v_i, j, 1, v_1, \{k\}\right) = \{c\left(i, v_i, j, 1, v_1, k\right)\}$$

$\forall i \in I; \quad \forall j, k \in J \quad$ with $j \neq k; \quad \forall v_i, v_1 \in \Upsilon$. All other states are initialized as not yet computed.

The dynamic programming recursion for the minimal time switching problem with collision avoidance and velocities selection is then

$$f\left(i, V, S\right) = \min\left\{ d\left(i, j\right) \ / \ v_i \ \vee \ f(i-1, V', S') \vee \right.$$
$$\left. \vee \ \widetilde{M} * C\left(i, v_i, j, i-1, V', S'\right) \right\}, \quad (6)$$

where $V = \left[V'\ v_i\right], \quad S' = S \backslash \{j\}$ and C is the collision function where $\widetilde{M}$ is a sufficiently large number so that any solution with collision is eliminated.

The initial conditions are given by

$f\left(1, v_1, \{j\}\right) = \min\left\{d\left(1, j\right) \ / \ v_1\right\}, \ \forall j \in J$ and $\forall v_1 \in \Upsilon$. All other states being initialized as not yet computed.

To determine the optimal value for our problem we have compute

$$\min_{\text{all } N\text{-tuples } V} \quad f\left(N, V, J\right).$$

3.3 Considering obstacles

The possibility of changing the path to deviate from obstacles is the main distinguishing feature of the work described in this article.

Consider the following pair of vehicles moving in space $\mathbb{R}^2$: vehicle i which travels at velocity v_i to position j that is reached in time T_i and the vehicle a which travels at velocity v_a to position b that is reached in time T_a. Collision between these two vehicles occurs if the following condition is satisfied (see equation (1)):

$$\left\| q_i(t) - q_a(t) \right\| < \varepsilon \quad (7)$$

where $q_i(t) = \left(x_i\left(t\right), y_i\left(t\right)\right)$ and $q_a(t) = \left(x_a\left(t\right), y_a\left(t\right)\right)$.

Assuming now that there is an obstacle, and if the vehicle a travels straight ahead collides with the obstacle (see Figure 6). What should the vehicle a do?

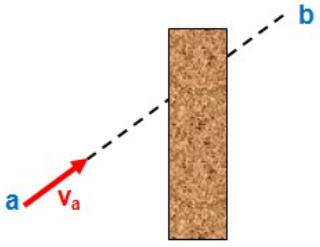

Fig. 6. Vehicle a travels straight ahead and intersects an obstacle.

Vehicle a travels straight ahead and intersects an obstacle.

Suppose that the obstacle is the rectangle $[Q_1Q_2Q_3Q_4]$ (see Figure 7), if not we can always put the obstacle inside of one rectangle. Our vehicle a is in the position $A = (x_a, y_a)$ and the goal is the position $B = (x_b, y_b)$ which is reached in time T_a. If the straight line $[AB]$ does not intersect the obstacle then the path will be straight ahead. Otherwise, we take the midpoint P of the straight line resulting from the intersection of the line $[AB]$ with the obstacle. Calculate the point $A_1 = (x_1, y_1)$, so that the point P is the midpoint of the straight line $[A_1B]$.

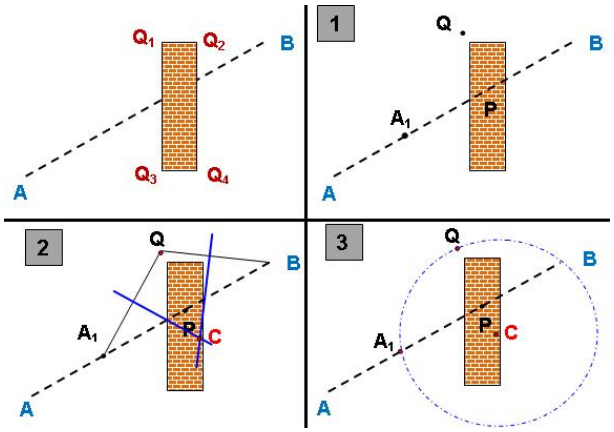

Fig. 7. Trajectory of the vehicle a from the position A to position B if there is an obstacle in the way.

Let $Q_0 \in \{Q_1, Q_2, Q_3, Q_4\}$. Considering the minimum between d_1 and d_2, where d_1 is the maximum distance between P and Q_1 and P and Q_2 and d_2 is the maximum distance between P and Q_3 and P and Q_4. The point Q_0 is the vertex of the rectangle that checks this minimum. If point Q_0 is determined by taking into account the distance d_1 then the vehicle circumvents the obstacle on top, otherwise, if Q_0 is determined by taking into account the distance d_2 then the obstacle is outlined below (see Figure 7-1).

Let $\xi \in \mathbb{R}^+$ and consider $Q = (x_q, y_q) = Q_0 + \xi\overrightarrow{PQ_0}$. Consider the arc of the circumference that passes through the points A_1, B and Q. The center of this circumference is determined by the intersection of the line bisection of the segment $[A_1Q]$ and the line bisection of the segment $[BQ]$. Let $C = (c_1, c_2)$ be the center of this circumference. The radius is $r = \overline{A_1C}$ (see Figure 7-2).

Defining $\alpha = \arctan\left(\frac{y_q - y_c}{x_q - x_c}\right)$ and $\theta = \arccos\left(\frac{d}{r}\right)$ (see Figure 8), where d is the distance between the midpoint of the line segment $[QB]$ and C. After time τ the vehicle travels from α to $\alpha - 2\theta$ at speed v_a taking $T_a - \tau = \frac{2\theta r}{v_a}$ seconds.

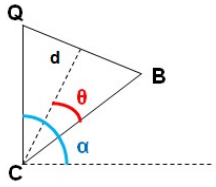

Fig. 8. The angles θ and α.

So the vehicle follows a straight line from point A to point Q which is reached in time τ, then it follows an arc of

circumference from point Q to point B defined as follows (see Figure 7-3):

$$\begin{cases} x(t) = c_1 + r\cos(\zeta), \\ y(t) = c_2 + r\sin(\zeta), \end{cases}, \qquad t \in [\tau, T_a].$$

where $\zeta = \alpha - 2\frac{t-\tau}{T_a-\tau}\theta$.

The trajectory of the vehicle a is, if $\tau \neq T_a$, then $q_a(t) = (x_a(t), y_a(t))$ where:

$$q_a(t) = \begin{cases} (x_a(0) + v_a t, y_a(0) + v_a t), & \text{if } t \in [0, \tau], \\ (c_1 + r\cos(\zeta), c_2 + r\sin(\zeta)), & \text{if } t \in [\tau, T_a]. \end{cases} \quad (8)$$

otherwise

$$q_a(t) = (x_a(0) + v_a t, y_a(0) + v_a t) \quad \text{for} \quad t \in [0, T_a]. \quad (9)$$

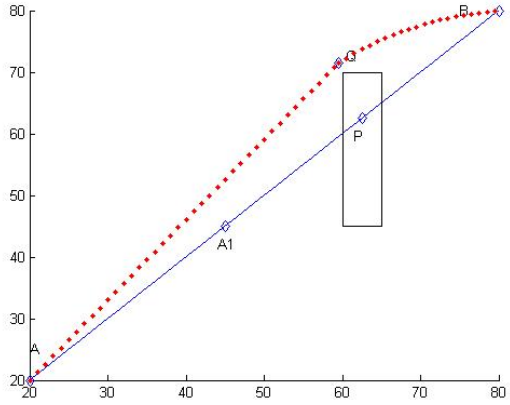

Fig. 9. Trajectory of vehicle a.

The logic-valued collision function C that checks whether a pair of vehicles collides is now redefined using condition (7) with the trajectories given by (8) or (9). Thus,

$$c(i, v_i, j, a, v_a, b, p_i, p_a) = 0$$

if

$$\|q_i(t) - q_a(t)\| < \varepsilon,$$

where $q_i(t)$ an $q_a(t)$ are defined by equations (8), that define the position along the path p_i (or p_a) for all times.

The objective function is now dependent on the length along the path p_i, that is $d_{ij}(p_i)$.

The length of the arc of circumference from point Q to point B is $2r\theta$. So,

$$d_{ij}(p_i) = \begin{cases} d_{ij} & \text{if vehicle } a \text{ travels straight ahead} \\ \widetilde{d}_{ij} + 2r\theta & \text{if vehicle } a \text{ intersects an obstacle.} \end{cases}$$

where $\widetilde{d}_{ij} = \left\|\overrightarrow{AQ}\right\|$.

4. CONCLUSION

We provide an optimization algorithm to decide how to reorganize a formation of vehicles into another formation of different shape, which is a relevant problem in cooperative control applications. The algorithms enables agent velocity choices and handles collision avoidance not only among agents but also with external obstacles.

The method proposed here should be seen as a component of a framework for multi-vehicle coordination/cooperation, which must necessarily include other components such as a trajectory control component.

The algorithm proposed is based on a dynamic programming approach and was shown to be a viable solution for problems with a small number of agents.

The proposed methodology is very flexible, in the sense that it easily allows for the inclusion of additional problem features, e.g. imposing geometric constraints on each vehicle or on the formation as a whole, using nonlinear trajectories, different objective functions, among others.

REFERENCES

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton.

Desai, J.P., Ostrowski, P., and Kumar, V. (2001). Modeling and control of formations of nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 17(6), 905–908.

Fontes, D.B.M.M. and Fontes, F.A.C.C. (2008). Optimal reorganization of agent formations. *WSEAS Transactions on Systems and Control*, 3(9), 789–798.

Fontes, D.B.M.M., Fontes, F.A.C.C., and Caldeira, A.C.D. (2012). Optimal formation switching with collision avoidance and allowing variable agent velocities. In A. Sorokin, M.T. Thai, and P.M. Pardalos (eds.), *Dynamics of Information Systems: Mathematical Foundations*, volume 20 of *Springer Proceedings in Mathematics and Statistics*, 165–179. Springer Verlag, New York.

Fontes, F.A.C.C. and Fontes, D.B.M.M. (2010). Minimal switching time of agent formations with collision avoidance. In M. Hirsch, P. Pardalos, and R. Murphey (eds.), *Dynamics of Information Systems: Theory and Applications*, volume 40 of *Springer Optimization and Its Applications Series*, 305–321. Springer Verlag, Berlin.

Hu, J. and Sastry, S. (2001). Optimal collision avoidance and formation switching on riemannian manifolds. *IEEE Conference on Decision and Control*, 2, 1071–1076.

Jin, Z., Shima, T., and Schumacher, C.J. (2006). Optimal scheduling for refueling multiple autonomous aerial vehicles. *IEEE Transactions on Robotics*, 22(4), 682–693.

Lau, N., Lopes, L., Corrente, G., and Filipe, N. (2009). Multi-robot team coordination through roles, positionings and coordinated procedures. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems - IROS*, 5841–5848.

Murray, R.M. (2007). Recent research in cooperative control of multivehicle systems. *Journal of Dynamic Systems, Measurement and Control*, 129, 571–583.

Rasmussen, S.J. and Shima, T. (2006). Branch and bound tree search for assigning cooperating UAVs to multiple tasks. In *American Control Conference 2006*. Institute of Electrical and Electronic Engineers. Minneapolis, Minnesota, USANagoya, Japan.

Rasmussen, S.J., Shima, T., Mitchell, J.W., Sparks, A., and Chandler, P.R. (2004). State-space search for improved autonomous UAVs assignment algorithm. In *IEEE Conference on Decision and Control*. Paradise Island, Bahamas.

Schumacher, C.J., Chandler, P.R., and Rasmussen, S.J. (2002). Task allocation for wide area search munitions via iterative network flow. In *American Institute of Aeronautics and Astronautics, Guidance, Navigation, and Control Conference 2002*. Reston, Virginia, USA.

Schumacher, C.J., Chandler, P.R., and Rasmussen, S.J. (2003). Task allocation for wide area search munitions with variable path length. In *Institute of Electrical and Electronic Engineers, American Control Conference 2003*. New York, USA.

Yamagishi, M. (2004). Social rules for reactive formation switching. Technical Report Technical Report UWEETR-2004-0025, Department of Electrical Engineering, University of Washington, Seattle, Washington, USA.