A Safe and Flexible CP-based Approach for Velocity Tuning Problems

Michaël Soulignac[1], Michel Rueher[2], and Patrick Taillibert[3]

[1]ISEN Lille, 41 Boulevard Vauban, 59046 Lille Cedex, France
michael.soulignac@isen.fr
[2]Nice Sophia Antipolis University, I3S/CNRS, BP 145, France
michel.rueher@gmail.com
[3]THALES Aerospace, 2 Avenue Gay Lussac, 78852 Elancourt, France
patrick.taillibert@fr.thalesgroup.com

Abstract. This paper introduces a new velocity tuning approach for autonomous vehicles based on Constraint Programming (CP) over continuous domains. We use CP to compute a safe approximation of configurations where collisions with obstacles may occur or technological limits may be violated. The use of CP leads to a flexible approach, facilitating the incorporation of new characteristics, e.g., constraints modeling the influence of currents. We illustrate these capabilities offered by CP in the context of UAV missions. Experimental results obtained on actual wind charts are provided.

1 Introduction

Recent advances made in the field of autonomous vehicles suggest that, in a near future, Unmanned Air Vehicles (UAVs) or Autonomous Underwater Vehicles (AUVs) will be more and more deployed in order to achieve various missions such as surveillance, intelligence or search and rescue. For physical or strategic reasons, these vehicles may not be able to receive directly orders from a headquarter in real-time. Thus, they have to embed their own motion planner. Because the environment is often changing or unknown, this planner has to be very reactive.

Motion planning among mobile obstacles is commonly divided into two tasks: path planning and velocity tuning. Numerous algorithms have been proposed for specific instances of velocity tuning. The main drawback of these approaches is that minor changes in the characteristics of the application may require deep changes in the algorithms.

That is why we propose a flexible approach based on Constraint Programming (CP) techniques. Practically, we use CP on continuous domains to identify the unreachable regions of a 2D space-time, modeling potential collisions with obstacles and/or violations of the technological limits of the vehicle. Thus, we can ensure that the deduced time-minimal trajectory is collision-free and feasible. An essential contribution of CP comes from the fact that it allows to easily incorporate new constraints. We illustrate this point by showing that constraints modeling the presence of currents can be added to the initial constraint system with a limited rework effort. First experimental results obtained on real wind charts are encouraging.

2 The problem

2.1 Informal description

A punctual vehicle, starting from a site A, has to compute a time-optimal trajectory to a goal site B, in a planar environment containing static and mobile obstacles, as illustrated in figure 1. This computation is done in two steps.

First, a path planning step, computing a path $\mathcal{P}$ avoiding static obstacles. Any path planning method from literature can be used in this step. The only assumption about the path $\mathcal{P}$ is that it is made up of successive line segments.

Second, a velocity tuning step, minimizing the arrival time at B and avoiding mobile obstacles, with a bounded velocity. This second problem is addressed in this paper.

2.2 Formalization

The environment is modeled by a 2-D Euclidean space E, with a frame of reference $R = (0, x, y)$. In R, the coordinates of a vector $\overrightarrow{u}$ are denoted (u_x, u_y) and its magnitude u. In particular, the vehicle's velocity vector relative to the frame R (ground speed) is denoted $\overrightarrow{v}$, and its magnitude v.

The path $\mathcal{P}$ is defined by a list V of n viapoints, denoted V_i. Each viapoint V_i is located on $\mathcal{P}$ at curvilinear abscissa l_i. Two successive viapoints (V_{i-1} and V_i) are linked by a line segment. In other terms, $\mathcal{P}$ is made up of successive line segments.

Each mobile obstacle O_i is modeled by a rectangle of size $S_{ix} \times S_{iy}$, performing successive straight line moves at constant velocity (dashed lines in fig. 1). This rectangle corresponds to a punctual mobile surrounded by a rectangular safety zone. Note that this safety zone can be easily extended to a polygon.

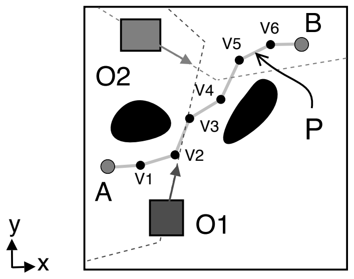

Fig. 1. A velocity tuning problem with 2 mobile obstacles (grey rectangles). The precomputed path $\mathcal{P}$ already avoids static obstacles (dark regions).

Our problem consists in finding a timing function $\sigma : M \in \mathcal{P} \mapsto t \in [0, T]$ minimizing the arrival time $t_B = \sigma(B)$, with respect to the following constraints: (1) maximal velocity: $v \leq \nu_{max}$, (2) mobile obstacles avoidance and (3) the latest arrival time T to B, due for instance to the embedded energy or visibility conditions. In other terms, our problem can be seen as mapping the path $\mathcal{P}$ to a trajectory.

3 Why CP?

The existing velocity tuning approaches generally work in a 2-D space-time. The first dimension $l \in [0, L]$ (where L is the length of the path) is the curvilinear abscissa on the path. The second one, $t \in [0, T]$ (where T is the latest arrival time), is the elapsed time since departure. As illustrated in figure 2, each point of the path is represented by a vertical line in the space-time. In particular, start and goal sites are represented by the extreme left and right vertical lines. Moreover, each mobile obstacle O_i generates a set of *collision surfaces* S_j in the space-time. These surfaces contain all pairs (l, t) leading to a collision between the vehicle and O_i.

Once the space-time is built, the initial velocity tuning problem can be reformulated into a path planning problem in this space-time. However, this space-time has specific constraints, notably due to time monotony or velocity bounds. Therefore, specific methods have been developed [10][20].

These methods are based on a discretization of the environment into elementary entities: line segments (fig. 3a) or polygonal cells (fig. 3b). These entities are then modeled as nodes of a graph.

The initial -concrete- path planning problem is thus reformulated into an abstract one: finding the shortest path in a graph. This problem is generally solved by applying an adaptation of a classical search algorithm, such as A^* [6] or one of its numerous variants.

As we can see, in the above approaches, unreachable regions of the space-time only materialize collisions with mobile obstacles. Other constraints about

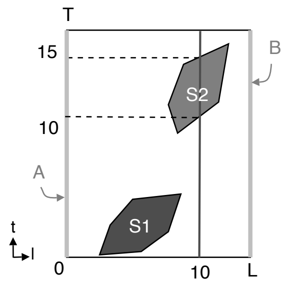

Fig. 2. The space-time corresponding to the example of figure 1, containing two collision surfaces S_1 and S_2. A collision necessarily occurs at abscissa $l = 10$ between $t = 10$ and $t = 15$.

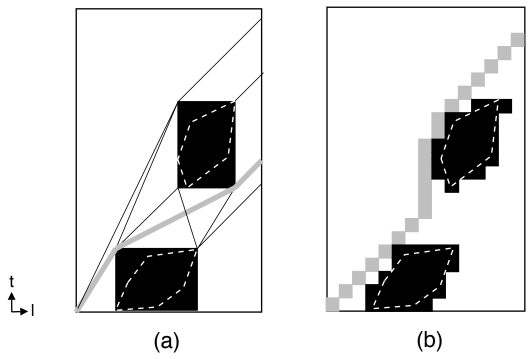

Fig. 3. (a) path (in light grey) found using the visibility graph method [10]; (b) path found using the cell decomposition method [20].

the vehicle and the environment are implicit, and are taken into account in the search algorithm. This has two major drawbacks: (1) the space-time does not reflect all the constraints of the problem, in particular the technological limits of the vehicle and (2) the search algorithm is customized to exploit some particular properties of the considered instance. Therefore, if the properties of this instance evolve, significant changes in the design and the implementation of this algorithm may be required.

The ability to add new constraints in a declarative way is the major advantage of using CP.

4 How CP?

As said before, we use CP on continuous domains to compute a safe approximation of unreachable regions in the space-time. This is done in two steps:

The first step consists in computing the *collision surfaces* described in section 3, that is regions in the space-time materializing collisions between obstacles and the vehicle. This step is done by solving a set of local CSPs, called $ColSurf\_ij$.

The second step consists in aggregating unreachable regions of the space-time, due to mobile obstacles (collision surfaces) and those due to the technological limits of the vehicle. This step is done by solving a global CSP, called *UnreachReg*. We will show that this CSP can be easily extended to include new constraints about the vehicle or the environment. We will illustrate this capability by adding currents in the initial problem.

All CSPs are defined on continuous domains. This choice is first explained, and next the two steps mentioned above are presented.

4.1 Modeling space-time regions with constraints over continuous domains

Modeling space-time regions in velocity-tuning problems can be done either with constraints over finite domains or over continuous domains. As explained in our previous paper [18], discrete domains impose to discretize quantities which are continuous by nature in the tuning problem, such as time. This leads to two issues:

1. If both time horizon T and the required precision on time τ are high, domains for variables modeling time may become huge. For instance, if $T = 10h$ and $\tau = 1s$, each domain contains 36000 values.
2. Discretizing time may lead to "miss" some collisions. Indeed, if no collision is detected at time t_{i-1}, and no collision is detected at $t_i = t_{i-1} + \tau$, it is generally assumed that no collision can occur between t_{i-1} and t_i. In fact, this assumption becomes more and more probable when τ tends towards 0, but it is never guaranteed.

To avoid these issues, we choose to model 2D space-time regions with constraints over continuous domains. The weak point of constraints over continuous domains is that they cannot in general guarantee that a solution exists in a box, even if this box is very small. The strong property of constraints over continuous domains is their refutation capabilities.

For this reason, the CSP described below is not used to model reachable regions but to describe an over-approximation of unreachable regions.

Indeed, since unreachable regions are over-estimated the remaining regions are necessarily reachable. However, as explained later, our approach is incomplete, because this over-estimation may eliminate the solutions of some very constrained problems.

4.2 Preliminaries: Computing collision surfaces in the space-time

In a CSP that we call $ColSurf\_ij$ (for "Collision Surface", pair of moves (i, j)), we model the following situation: (1) the vehicle and a mobile obstacle are moving on line segments i and j of their respective trajectory, and (2) a collision occurs between them. Solving this CSP may lead to two results:

1. The CSP is inconsistent, then we are sure that no collision will occur during this pair of moves.
2. The CSP is consistent, some collisions may occur in a time interval denoted D^c_{ij}.

Therefore, if the interval D^c_{ij} is not empty, it represents a part of a collision surface. By repeating the process described above for all possible values of i and j, we are able to compute an outer approximation of collisions surfaces.

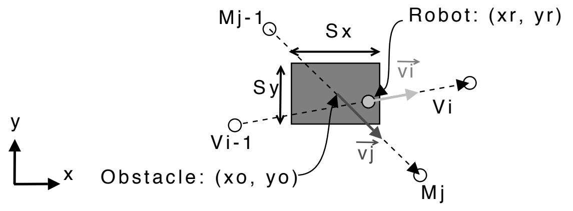

Fig. 4. A collision between the vehicle and a mobile obstacle: the vehicle lies to the rectangular safety zone (dark grey).

1) Formulation of the CSP LocalCol_ij

Let us consider that the vehicle and a mobile obstacle are moving line segments i and j of their respective trajectory:

- the vehicle is moving between viapoints $V_{i-1} = (x_{i-1}, y_{i-1})$ and $V_i = (x_i, y_i)$,
- an obstacle, of size $S_x \times S_y$ is performing its jth straight line move, at velocity v_j, between two points $M_{j-1} = (x_{j-1}, y_{j-1})$ and $M_j = (x_j, y_j)$, starting at time t_{j-1}.

As depicted in figure 4, a collision between the vehicle and the obstacle occurs if the vehicle lies to the rectangular safety zone.

If we denote (x_r, y_r) the position of the vehicle, (x_o, y_o) the position of the obstacle, and t_c the collision time between them, this situation can be modeled by the following equations:

$$\begin{cases} a_i = y_{i-1} - y_i \\ b_i = x_i - x_{i-1} \\ c_i = y_i \cdot x_{i-1} - x_i \cdot y_{i-1} \\ a_i \cdot x_r + b_i \cdot y_r + c_i = 0 \\ (x_r - x_{i-1}) \cdot (x_r - x_i) \leq 0 \\ (y_r - y_{i-1}) \cdot (y_r - y_i) \leq 0 \end{cases} \qquad (VehicleMotion)$$

[The vehicle is moving on the line segment $[V_{i-1}, V_i]$, of equation $a_i \cdot x_r + b_i \cdot y_r + c_i = 0$]

$$\begin{cases} d_{j_x} = x_j - x_{j-1} \\ d_{j_y} = y_j - y_{j-1} \\ d_j = \sqrt{d_{j_x}^2 + d_{j_y}^2} \\ v_{j_x} = v_j \cdot (d_{j_x}/d_j) \\ v_{j_y} = v_j \cdot (d_{j_y}/d_j) \\ x_o = v_{j_x} \cdot (t_c - t_{j-1}) + x_{j-1} \\ y_o = v_{j_y} \cdot (t_c - t_{j-1}) + y_{j-1} \end{cases} \qquad (ObstMotion)$$

[The obstacle is moving on the line segment $[M_{j-1}, M_j]$ at velocity v_j, starting at time t_{j-1}. The traveled distance is denoted d_j]

$$\begin{cases} x_r \geq x_o + S_x/2 \\ x_r \leq x_o - S_x/2 \\ y_r \geq y_o + S_y/2 \\ y_r \leq y_o - S_y/2 \end{cases} \qquad (Collision)$$

[The vehicle is within the rectangular safety zone of the obstacle, of size $S_x \times S_y$]

In the above equations, the domain of variable t_c represents the set of times where vehicle may collide the obstacle. This domain is used later to approximate collision surfaces.

Of course, in this very simple case –linear moves for the vehicle and obstacles– the domain could be computed analytically or symbolically. However, symbolical computation may become costly as soon as the constraints are more complex (e.g, 3D spaces). On top of it, our goal is to offer a great flexibility, and thus, we do not want to use any *adhoc* technique.

For instance, consider that the vehicle has to perform a set of circles, modeling a surveillance task. This could easily be done by replacing line equations by circle equations in the above model whereas the analytic resolution would require much more work.

2) Computing collision surfaces

Let us define the interval of *collision times*, denoted T_i^{col}. This interval has the following meaning: if $t_i \in T_i^{col}$, then the vehicle collides a mobile obstacle between V_{i-1} and V_i. Using the CSP described above, each interval T_i^{col} can be computed by the following procedure:

For each mobile obstacle O_k do:

```
1. Let $L_k$ denote the number of line segments performed by $O_k$
2. For $j$ from 1 to $L_k$:
   (a) Initialize $T_i^{col}$ by : $T_i^{col} \leftarrow \emptyset$
   (b) Solve the CSP $LocalCol\_ij$
   (c) If $LocalCol\_ij$ is consistent
       i. Let $D_{ij}^c$ denote the domain of variable $t_c$
       ii. Update $T_i^{col}$ by: $T_i^{col} \leftarrow T_i^{col} \cup D_{ij}^c$
```

In the general case, the interval T_i^{col} computed by this procedure is an union of subintervals, i.e. $T_i^{col} = \cup_{j=1}^{s_i}[t_j^-, t_j^+]$, where s_i is the number of possible collisions between V_{i-1} and V_i.

As shown in figure 5, the whole T_i^{col} models an outer approximation of collision surfaces introduced in section 3. Thus, all possible collisions are captured by T_i^{col}.

That is, contrary to most existing approaches, even if forbidden surfaces are approximated, our approach remains correct: the behavior of the vehicle is guaranteed to be safe. On the other hand, our approach is incomplete, because a too rough approximation can obstruct the space-time. In those conditions, the problem could be considered as unsolvable, even if solutions exist.

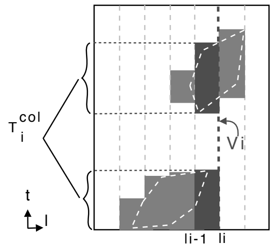

Fig. 5. Collision times T_i^{col} in the space-time of fig. 2, discretizing collision surfaces by a set of bands.

4.3 Computing the unreachable regions of the space time

We define a CSP, called *UnreachReg* (for "Unreachable Regions"), which aggregates the collision surfaces computed in the last step with the unreachable regions due to the technological limits of the vehicle. Here, the only limits taken into account are velocity bounds.

Note that only the principle of the CSP is explained below. A detailed description of constraints can be found in the chapter 7 of [16].

1) Unreachable regions due to excessive velocity

Let us consider a move of the vehicle on an arbitrary line segment $[V_{i-1}, V_i]$. If we denote respectively v_i, t_i and d_i the vehicle's velocity, the travel time and the traveled distance on this line segment, these variables are linked by $t_i = t_{i-1} + d_i/v_i$.

Using this relationship, we can model a *too early* arrival at viapoint V_i, that is, an arrival violating the maximal velocity constraint. Indeed, all times t_i verifing this relationship with $v_i > \nu_{max}$, are, by definition, forbidden. The corresponding constraints generate a "floor" in the space-time, as illustrated in figure 6a.

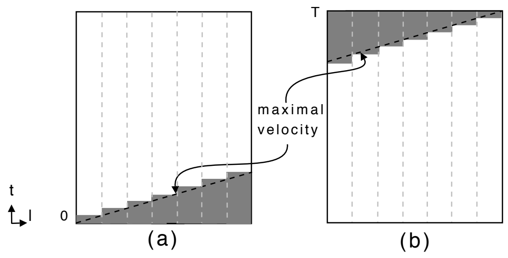

Fig. 6. (a) floor modeling a too early arrival; (b) ceiling modeling a too late arrival. Ceiling and floor are delimited by the dashed line of slope ν_{max}.

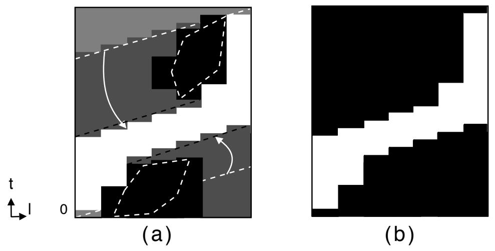

Fig. 7. (a) Shadows (dark grey) obtained by temporally shifting light grey regions (due to velocity bounds); (b) the final space-time, containing extended ceiling and floor.

The same reasoning can be done for too late arrival, with regard to the lattest arrival time T. The corresponding constraints generate a "ceiling" in the space-time, as illustrated in figure 6b.

2) Merging unreachable areas

Unreachable areas due to the mobile obstacles and the technological limits of the vehicle are interconnected. For instance, we have to respect to velocity bounds, if we want to speed up the vehicle so that it will arrive at a given point before some obstacle.

In the space-time, collision surfaces will lead to a temporal shifting of the ceiling and the floor of figure 6, as depicted in figure 7a.

Let us explain the shifting of the floor.

The presence of mobile obstacles extends the concept of a too early arrival. Indeed, if an obstacle O_j occupies the viapoint V_i during $D_j = [t_j^-, t_j^+]$, then all arrival time lying in D_j have to the shifted to t_j^+. In other terms, if the vehicle reach V_i at time $t_i \in D_j$, then it will have to wait until time t_j^+ (that is, until V_i is obstacle-free).

The same reasoning can be done for the ceiling.

The corresponding constraints generate extended ceiling and floor in the space-time (see figure 7b). These unreachable areas merge the effect of moving obstacles and technological limits of the vehicle. Graphically, they can be interpreted as "shadows" behind collision surfaces.

4.4 Deducing the time-minimal path

Once the CSP is numerically solved, the final space-time, containing the extended floor and ceiling is known. Two situations may arise:

1. A corridor exists between the floor and the ceiling (case of fig. 7b). In this case, it is very simple to deduce the time-minimal path in this space-time. For instance, it can be obtained by linking the upper edges of the floor, as depicted in figure 8.

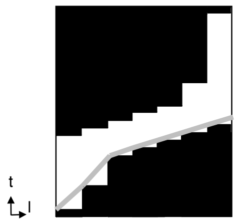

Fig. 8. The time-minimal path (in grey) obtained in the space-time of fig. 7b.

2. The space-time is obstructed (the floor intersects the ceiling). In this case the problem is *a priori* unsolvable, but we cannot be sure about that. However, it is possible that unreachable regions are too roughly approximated.

4.5 Implementation

The approach introduced in this paper has been implemented in our trajectory planner, *Airplan*, first described in [17].

In Airplan, the initial velocity tuning module, based on the clpfd solver [5] (discrete domains + arc-consistency), has been replaced by another one, using the Interlog solver [4] (continuous domains + 2B-consistency). This new version of Airplan will be demonstrated during the conference.

We choose to use a weak local consistency like 2B-consistency [12], because it is light to implement and computationally efficient in our case (see experimental results). Of course, stronger consistencies like Box [7], HC4 / HC4-Revise [3], Quad [11] or I-CSE [2] would probably provide sharper approximations but their implementation is much more costly.

5 Added value of CP

We showed that a first added value of CP on continuous domains was the correctness: since non-solution spaces may be over-estimated, the computed trajectory is guaranteed to be safe, that is, collision-free.

In this section, we illustrate another benefit of CP: flexibility. In section 4, we presented a global CSP modeling the "core" velocity tuning problem introduced in section 2. Specific constraints have to be added to this CSP to take into account some vehicle's specificities, such as curvature or acceleration constraints, sensors capacities or configuration, etc.

We focus here on the main specificity of Unmanned Aerial Vehicles (UAVs): their sensibility to currents. This can be easily modeled by adding some constraints in the core model.

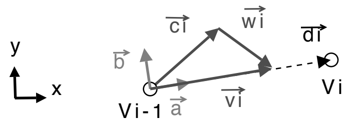

Fig. 9. The velocity composition law between V_{i-1} and V_i.

5.1 Constraints formulation

In presence of currents, constraints related to the vehicle velocity have to model two additional properties:

- velocity composition law: the velocity of the vehicle depends both on the engine command and on the velocity of the current.
- path following: the vehicle has to compensate the disturbances of the currents to stay on the path. Specific methods have been developed to address this problem [14], but here, the flexibility of CP avoids to use them.

To illustrate this point, let us consider the straight line move $\vec{d_i}$, between the viapoints V_{i-1} and V_i. For this move, we define: (1) $\vec{c_i}$ the average velocity of the current, (2) $\vec{v_i}$ the vehicle's velocity relative to the frame R, and (3) $\vec{w_i}$ the vehicle's velocity relative to $\vec{c_i}$.

As shown in figure 9, velocities $\vec{v_i}$ and $\vec{w_i}$ are linked by the composition law $\vec{v_i} = \vec{w_i} + \vec{c_i}$.

Moreover, since we want to impose the vehicle to stay on the path, $\vec{v_i}$ and $\vec{d_i}$ are collinear.

Using the local frame $R_i = (V_{i-1}, \vec{a}, \vec{b})$, with $\vec{a} = \vec{d_i}/d_i$ and $\vec{a} \perp \vec{b}$, unreachable regions due to an excessive velocity (i.e. greater than ν_{max}) can be expressed as follows in presence of currents:

$$\begin{cases} v_i = c_{ia} + w_{ia} \\ 0 = c_{ib} + w_{ib} \end{cases} \qquad (VehicleVel)$$

[velocity composition law in the frame R_i]

$$\begin{cases} c_a = c_i \cdot \cos\alpha \\ c_b = c_i \cdot \sin\alpha \\ \cos\alpha = \vec{c_i} \cdot \vec{d_i}/c_i \cdot d_i = (c_{ia}d_{ia} + c_{ib}d_{ib})/c_i \end{cases} \qquad (CurrentVel)$$

[coordinates of $\vec{c_i}$ in R_i]

$$\begin{cases} {w_i}^2 = w_{ia}^2 + w_{ib}^2 \\ w_i > \nu_{max} \end{cases} \qquad (VelBounds)$$

[maximal velocity relative to the current c_i]

To integrate the influence of currents on the vehicle, the unique constraint $v > \nu_{max}$ introduced in the problem statement has simply to be replaced by the sets of equations $(VehicleVel)$, $(CurrentVel)$ and $(VelBounds)$ described above.

5.2 Result

Figure 10 shows the impact of currents on the previous space-time, and on the solution.

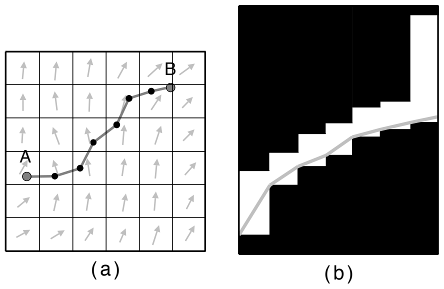

Fig. 10. (a) Adding currents in the initial problem (grey arrow = velocity vectors). (b) Corresponding space-time and solution

6 Experimental results

The key idea of this paper is that velocity tuning based on CP leads to a safe and flexible approach. To evaluate the capabilities of this approach, we have performed experiments on actual wind chart for UAV missions.

6.1 The test-cases

As shown in figure 11, a test-case is a simple UAV mission. The UAV takes off at A, and then evolves on a precomputed path $\mathcal{P}$ until reaching B. The UAV has to optimize its velocity on $\mathcal{P}$, to take the wind and mobile obstacles into account.

Each test-case has been generated as follows:

- Wind charts were daily collected during three months on the Meteo France website [13], leading to 100 different realistic environments with currents.
- A (start) and B (goal) points: randomly placed on the wind chart.
- Path: computed using the sliding wavefront expansion, described in [19]. This algorithm provided feasible and smooth paths, even in presence of strong currents.

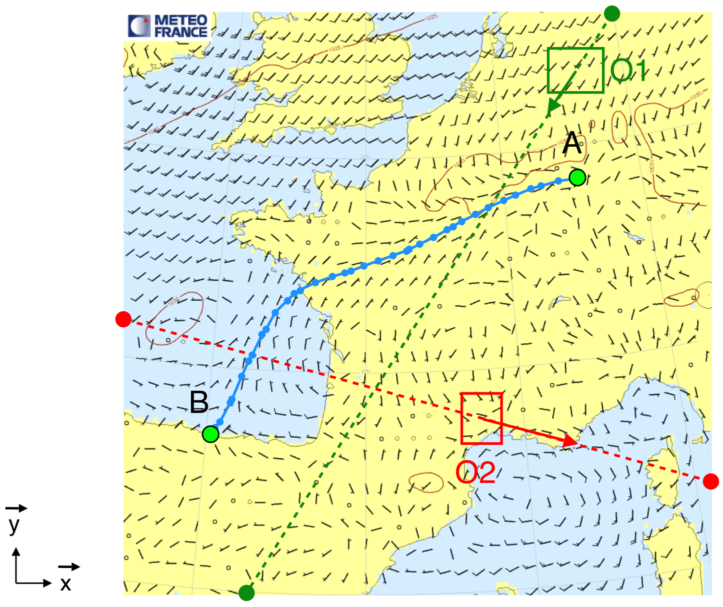

Fig. 11. A test-case with $m = 2$ mobile obstacles. The path is made up of $n = 36$ viapoints, which leads to a CSP containing about 400 variables and 650 constraints.

– Mobile obstacles: each mobile obstacle goes across the environment by performing a straight line move $P_1 \to P_2$ at constant velocity. P_1 and P_2 are randomly chosen on two borders of the environment, until an intersection I between the path $\mathcal{P}$ and the line segment $[P_1, P_2]$ is detected. Their velocity and size are randomly set in realistic intervals of values.

Note that, in the particular context of UAV missions, both the fact that mobiles obstacles' trajectories are piecewise linear and known in advance are realistic. These assumptions model well flight plans of potential airplanes in the area (possibly updated in real-time from the headquarter) or of other autonomous vehicles, in other to perform a synchronized mission.

6.2 Computation time

The resulting computation times are plotted in figure 12. Each dot is the mean time obtained on 500 test-cases. These results have been obtained by running the last version of Airplan on a 1.7Ghz PC with 1Go of RAM. Note that better results could be obtained by using state of art software such as, for instance: RealPaver [15], ALIAS [1], Icos [9] or Ibex [8].

From a strictly qualitative point of view, the global computation time remains reasonable (under 1 second) even in presence of many moving obstacles. Therefore, our approach is both flexible and computationally efficient.

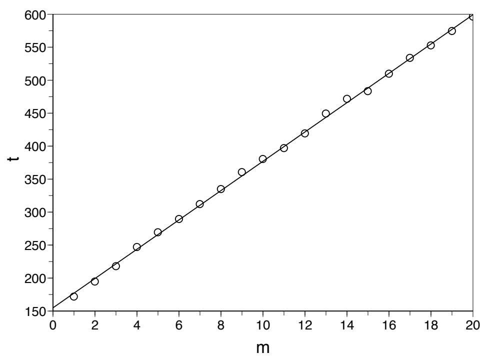

Fig. 12. Computation time t (in ms) in presence of m moving obstacles (white dots: experimental results; black line: linear regression).

From a statistical point of view, the computation time appears to be linear. A linear regression confirms this assumption in the range of our tests (with a coefficient of determination $R^2 > 0.999$).

This seems natural, because the size of our CSP is in $O(n \cdot m)$, where n is the number of viapoints on the vehicle's path $\mathcal{P}$, and m the number of obstacles. Therefore, it was expected that the average computation time was in $O(\widetilde{n} \cdot m)$, where $\widetilde{n}$ is the average number of points on $\mathcal{P}$ ($\widetilde{n} \approx 20$ in our tests). However, a deeper study is required to confirm these results in the general case.

7 Conclusion

In this paper, we proposed a constraint-based flexible approach for handling velocity tuning problems. CP allows to model the "core" velocity tuning problem, and then to easily extend it to more complex constraints, in particular the presence of currents.

We also showed that CP techniques on continuous domains have interesting properties: (1) the use of continuous domains allows not to discretize the time, and thus to guarantee a collision-free solution; (2) in the particular case of velocity tuning, 2B-consistency techniques are computationally efficient and thus potentially usable in on-boards planners.

Future work could concern more complete experiments and the evaluation of the capabilities of higher consistencies. Higher consistencies would probably provide a sharper approximation but we have to check whether the computation cost is not too high with respect to the constraints of on-board planners.

References

1. Alias website. `http://www-sop.inria.fr/coprin/logiciels`.
2. Araya, I., Trombettoni, G., Trombettoni, and Neveu, B. Filtering numerical csps using well-constrained subsystems. In *Constraint Programming, LNCS 5732* (2009), p. 158172.
3. Benhamou, F., and Granvilliers, L. *Continuous and Interval Constraints. Chapter 16: Handbook of Constraint Proof Constraint Programming.* Elsevier.
4. Botella, B., and Taillibert, P. Interlog: constraint logic programming on numeric intervals. In *International Workshop on Software Engineering, Artificial Intelligence and Expert Systems* (1993).
5. Carlsson, M., Ottosson, G., and Carlson, B. An open-ended finite domain constraint solver. In *Proceedings of Programming Languages: Implementations, Logics, and Programs* (1997).
6. Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik* (1959), 269–271.
7. Hentenryck, P. V., McAllester, D., and Kapur, D. Solving polynomial systems using a branch and prune approach. *SIAM Journal on Numerical Analysis 34* (1997).
8. Ibex website. `http://www.ibex-lib.org`.
9. Icos website. `http://sites.google.com/site/ylebbah/icos`.
10. Ju, M.-Y., Liu, J.-H., and Hwang, K.-S. Real-time velocity alteration strategy for collision-free trajectory planning of two articulated robots. *Journal of Intelligent and Robotic Systems 33* (2002), 167–186.
11. Lebbah, Y., Michel, C., Rueher, M., Daney, D., and Merlet, J. Efficient and safe global constraints for handling numerical constraint systems. *SIAM Journal on Numerical Analysis 42* (2005), 20762097.
12. Lhomme, O. Consistency techniques for numeric csps. In *Proceedings of International Joint Conference on Artificial Intelligence* (1993), pp. 232–238.
13. Meteo france website. `http://marine.meteofrance.com/marine/accueil?51759.path=marine\%252Fimgmervent`.
14. Nelson, D., Barber, D. B., McLain, T. W., and Beard, R. W. Vector field path following for small unmanned air vehicles. In *Proceedings of the American Control Conference* (2006).
15. Real paver website. `http://realpaver.sourceforge.net`.
16. Soulignac, M. Planification de trajectoire en prsence de courants. applications aux missions de drones. *PhD Thesis - THALES/Nice Sophia-Antipolis university* (2009), 179–203.
17. Soulignac, M., and Taillibert, P. Fast trajectory planning for multiple site surveillance through moving obstacles and wind. In *Proceedings of the Workshop of the UK Planning and Scheduling Special Interest Group* (2006), pp. 25–33.
18. Soulignac, M., Taillibert, P., and Rueher, M. Velocity tuning in currents using constraint logic programming. In *Proceedings of the Workshop of the UK Planning and Scheduling Special Interest Group* (2007), pp. 106–113.
19. Soulignac, M., Taillibert, P., and Rueher, M. Adapting the wavefront expansion in presence of strong currents. In *Proceedings of International Conference on Robotics and Automation* (2008), pp. 1352–1358.
20. van den Berg, J., and Overmars, M. H. Roadmap-based motion planning in dynamic environments. *Transactions on Robotics and Automation 21* (2005), 885–897.