HYBRIDGE

Distributed Control and Stochastic Analysis of Hybrid Systems Supporting Safety Critical Real-Time Systems Design

WP5: Control of uncertain hybrid systems

Randomised algorithms and implementation

A. Lecchini[1] , X. Papageorgiou[2] , J. Lygeros[3]
J. Maciejowski[1] and K. Kyriakopoulos[2]

April 20, 2005

Version:	*0.4*
Task number:	*5.3*
Deliverable number:	*D5.3*
Contract:	*IST-2001-32460 of European Commission*

[1] Department of Engineering, University of Cambridge, U.K.
[2] Department of Mechanical Engineering, National Technical University of Athens, Greece
[3] Department of Electrical and Computer Engineering, University of Patras, Greece.

DOCUMENT CONTROL SHEET

Title of document:	*Randomised algorithms and implementation*
Authors of document:	*A. Lecchini, J. Lygeros and J. Maciejowsi*
Deliverable number:	*D5.3*
Contract:	*IST-2001-32460 of European Comission*
Project:	*Distributed Control and Stochastic Analysis of Hybrid Systems Supporting Safety Critical Real-Time Systems Design (HYBRIDGE)*

DOCUMENT CHANGE LOG

Version #	Issue Date	Sections affected	Relevant information
0.1	09/02/04	All	First draft
0.2	26/10/04	All	Second draft
0.3	30/11/04	All	Submitted for internal review
0.4	20/04/05	All	Final version

Version 0.3		Organisation	Signature/Date
Authors	A. Lecchini	UCAM	
	X. Papageorgiou	NTUA	
	J. Lygeros	UPAT	
	J.Maciejowski	UCAM	
	K.Kyriakopoulos	NTUA	
Internal reviewers	P. Lezaud	CENA	
	H. Blom	NLR	

HYBRIDGE, IST-2001-32460
Work Package WP5, Deliverable D5.3

Randomized algorithms and implementation

Prepared by:
A. Lecchini[*], X. Papageorgiou[†], J. Lygeros[‡],
J. Maciejowski[*], and K.J. Kyriakopoulos[†]

Abstract

In this deliverable we present the optimization framework adopted for conflict resolution in WP5 of HYBRIDGE. In particular, we present the Monte Carlo Markov Chain (MCMC) framework, which we adopt in our Model Predictive Control approach to conflict resolution in a stochastic setting, and a neurodynamic optimization approach for efficient conflict resolution when the resolution problem is posed as an optimal control problem in a deterministic setting.

[*]Department of Engineering, University of Cambridge, Cambridge, CB2 1PZ, UK, Tel. +44 1223 33260, Fax. +44 1223 332662, `al394, jmm@eng.cam.ac.uk`

[†]Department of Mechanical Engineering, National Technical University of Athens, Zografou, Athens, GR-15700, GREECE, {`xpapag, kkyria`}`@mail.ntua.gr`

[‡]Department of Electrical and Computer Engineering, University of Patras, Rio, Patras, GR-26500, GREECE, Tel. +30 2610 996 458, Fax. +30 2610 991 812, `lygeros@ee.upatras.gr`

Contents

List of Figures

List of Acronyms

ATC	Air Traffic Control
ATM	Air Traffic Management
BADA	Base of Aircraft Data
CD / CR	Conflict Detection / Resolution
FMS	Flight Management System
MC	Monte Carlo
MCMC	Monte Carlo Markov Chain
MPC	Model Predictive Control
PMM	Point Mass Model

1 Aim and Scope

The objective of WP5 of HYBRIDGE is to develop algorithms for assisting air traffic controllers and pilots with conflict resolution maneuvers. In Deliverable D5.1 we gave an introduction to the structure of Air-Traffic Management and to the role of air traffic controllers in the current system. In Deliverable D5.2 we proposed a Model Predictive Control formulation of the conflict resolution task in Air-Traffic Control. The main aim of Deliverable D5.3 of HYBRIDGE is summarized in Task 5.3 of WP5:

> The optimization problem is likely to be computationally demanding. Randomized algorithms will be employed to obtain efficient estimates of the optimum and confidence bounds.

In this deliverable we present the optimization framework adopted for conflict resolution in WP5 of HYBRIDGE. The optimization problems that arise in the context of conflict resolution are notoriously complex computationally. We consider two different methodologies for efficiently computing approximate solutions to these optimization problems. The first part of the deliverable is devoted to the stochastic optimization framework developed for the Model Predictive Control approach to conflict resolution in a stochastic setting. The second part is devoted to a neurodynamic optimization approach for efficient conflict resolution when the resolution problem is posed as an optimal control problem in a deterministic setting.

In Section 2 we introduce the context of optimization for Stochastic Model Predictive Control. In Section 3.1, we discuss the penalty formulation of constrained optimization problems in a stochastic setting. The Metropolis-Hasting algorithm, introduced in Section 3.2, and Simulated Annealing, introduced in Section 3.3, will be used as optimization tools to solve the optimization problem. Simulated Annealing was originally developed for a deterministic setting, Section 3.4 is devoted to present the use of this method for the optimization of an expected value criterion. In Section 3.5 we discuss implementation issues and improvements. The reader is referred to Deliverable D5.4 for an illustration of the use of these optimisation algorithms for conflict resolution. This illustration will be given through simulations benchmarks inspired by current ATC practice.

Section 4 initiates the second part of the deliverable by introducing conflict resolution as an optimal control problem. The resulting non-linear optimization problem is formulated in Section 4.1-2. Section 4.3 describes the use of neural networks for efficient approximation of the solution.

2 Optimization for Model Predictive Control

The essence of the Model Predictive Control (MPC) approach to the control of uncertain systems is the following algorithm:

Step 1: Calculate the control inputs that optimize the performance of the controlled variables, as predicted by a prediction model, according to a performance criterion defined over a prediction horizon;

Step 2: Apply the initial segment of the optimized control input trajectory to the system and estimate the new state of the system from the latest measurements. Jump back to Step 1 and repeat.

The use of the latest measurements, combined with the *receding horizon strategy*, introduces feedback which combats the uncertainty in the evolution of the system. A unique advantage of MPC, compared with other control methodologies, is that constraints can be introduced into the problem formulation; a constrained optimization problem must then be solved [15]. MPC relies on the repeated solution of an optimization problem in real time; advances in MPC are therefore heavily dependent on advances in optimization.

Most research on MPC has assumed a deterministic setting in which uncertainty (unknown disturbances and modeling errors) is described by hard-bounded intervals which represent admissible ranges of values for the unknown quantities in the system. This approach leads to criteria of worst-case type, typically leading to min-max optimization problems [10, 15]. On the other hand, stochastic descriptions of uncertainty are very common and widely used. In a stochastic setting, a natural criterion is the expected value of the performance over the prediction horizon. The optimization of an expected value criterion is a difficult task. Except for a very few special cases, it is usually impossible or computationally intractable to evaluate analytically the integrals involved in calculating the expectations. One of the few examples, for the case of Stochastic Predictive Control, is [11] where, motivated by the particular application, Gaussian and linear-in-the-instruments models are considered and therefore exact calculations can be made. In general, currently there exists no usable method of applying the MPC approach to more complex stochastic nonlinear and hybrid dynamical systems. Here, the term hybrid denotes dynamical systems where the continuous evolution is interlaced by abrupt changes regulated by finite discrete, and possibly probabilistic, dynamics [3, 5, 8].

In HYBRIDGE we address the solution of such problems, based on recent advances in simulation-based Monte Carlo methods. The speed and power of modern computers allow such solutions to be implemented in real-time for an increasing range of applications — among these applications is Air-Traffic Control, where substantial computational power is available and computation times of the order of one minute are acceptable.

Our simulation-based approach is motivated by conflict resolution in Air-Traffic Control. Here, we introduce the context of research in Air-Traffic Control and explain why this approach represents an important novelty in this area.

In the current organisation of Air-Traffic Management the centralised Air-Traffic Control is in complete control of the air-traffic and ultimately responsible for safety. The main objective of Air-Traffic Control is to maintain safe separation between aircraft by issuing proper instructions to the pilots. A conflict is defined as the situation of loss of minimum safe separation between two aircraft. If it is possible, Air-Traffic Control tries also to fulfil the (possibly conflicting) requests of aircraft and airlines; for example, desired paths to avoid turbulence or desired time of arrivals to meet schedule.

In order to improve performance of Air-Traffic Control, mainly in anticipation of increasing levels of traffic, research effort has been spent in the last decade on creating tools for conflict detection and resolution. A review of research work in this area of Air-Traffic Control is presented in [12].

Uncertainty is introduced in air-traffic by the action of the wind field, incomplete knowledge of the physical coefficients of the aircraft and unavoidable imprecision in the execution of ATC instructions. In conflict detection the objective is to evaluate conflict probability over a certain future horizon starting from the current positions and flight plans of the aircraft. In conflict resolution the objective is to calculate suitable maneuvers to avoid a predicted conflict. A number of conflict resolution algorithms has been proposed for a deterministic setting, for example [7, 9, 21]. These methods are characterised by the use of extremely simplified models, which are forced by the computational complexity of the algorithms. Moreover, in conflict resolution several resolution maneuvers usually coexist and the choice of the actual resolution maneuver implies the solution of the combinatorial problem of selecting the proper coordination between the aircraft. Thus the computational complexity grows explosively with the number of aircraft. In a stochastic setting, resolution strategies virtually do not exist. The research has concentrated mainly on efficient algorithms for conflict detection. The main reason for this is the complexity of stochastic prediction models which, even if it does not make it impossible to estimate conflict probability through Monte Carlo methods, it makes the quantification of the effects of possible control actions intractable. To the best of the authors knowledge the only probabilistic approach to conflict resolution is the decentralized approach of [18], based on stochastic variant of "gradient descent" type methods used in robotics. Again, the dynamics considered in this work are very simplistic.

The main bottleneck in conflict resolution using optimization methods is the computational complexity of the algorithms. As discussed above, this has forced most authors to restrict their attention to extremely simplistic (and hence unrealistic) models. Instead of approximating the model, we approximate the optimization computation. Two different approximations schemes are considered. One is probabilistic and is based on stochastic optimization methods and the other is deterministic and is based on the principles of neuro-dynamic programming.

3 Randomized approximation methods

3.1 Penalty formulation of an expected value optimisation problem with constraints

In our approach we formulate conflict resolution as a constrained optimisation problem. Detailed formulations for two practical ATM problems are given in [13, 14]. Here it suffices to say that, given a set of aircraft, a maneuver is determined by a parameter ω which defines the nominal paths of the aircraft during the maneuver and that the actual execution of the maneuver is affected by uncertainty. Therefore, the sequence of actual positions of the aircraft during the resolution maneuver (for example: the sequence of positions every 6 seconds which is a typical time interval between two successive radar sweeps) *a-priori* of its execution is a random variable denoted by X. A conflict is defined as the event that the positions of two aircraft during the execution of the maneuver are too close. The objective is to select ω in order to maximise the expected value of some measure of performance associated to the execution of the resolution maneuver while ensuring a small probability of conflict. In this section we introduce the formulation of the problem in a general fashion.

Let X be a random variable whose distribution depends on some parameter ω. The distribution of X is denoted by $p_\omega(x)$ with $x \in \mathbf{X}$. The set of all possible values of ω is denoted by $\boldsymbol{\Omega}$. We assume that a constraint on the random variable X is given in terms of a feasible set $\mathbf{X_f} \subseteq \mathbf{X}$. We say that a realisation x, of random variable X, violates the constraint if $x \notin \mathbf{X_f}$. Moreover, we assume that for a realisation $x \in \mathbf{X_f}$ some definition of performance of x is given. In general performance can depend also on the value of ω, therefore performance is measured by a function $\mathrm{perf}(\omega, x)$, $x \in \mathbf{X_f}$, $\omega \in \boldsymbol{\Omega}$. We assume that $\mathrm{perf}(\omega, x)$ takes values in $(0, 1]$. The probability of satisfying the constraint is denoted by $\mathrm{P}(\omega)$

$$\mathrm{P}(\omega) \;=\; \int_{x \in \mathbf{X_f}} p_\omega(x) dx \,. \tag{1}$$

The probability of violating the constraint is denoted by $\bar{\mathrm{P}}(\omega) = 1 - \mathrm{P}(\omega)$. The expected performance for a given $\omega \in \boldsymbol{\Omega}$ is denoted by $\textsc{Perf}(\omega)$, where

$$\textsc{Perf}(\omega) = \int_{x \in \mathbf{X_f}} \mathrm{perf}(\omega, x) p_\omega(x) dx \,. \tag{2}$$

Ideally one would like to maximise the performance over all ω, subject to a bound on the probability of constraint satisfaction. Given a bound $\bar{\mathbf{P}} \in [0, 1]$, this corresponds to solving the constrained optimization problem

$$\textsc{Perf}_{\max|\bar{\mathbf{P}}} = \sup_{\omega \in \boldsymbol{\Omega}} \textsc{Perf}(\omega) \tag{3}$$

$$\text{subject to } \bar{\mathrm{P}}(\omega) < \bar{\mathbf{P}}. \tag{4}$$

Clearly, a necessary condition for the problem to have a solution is that there exists $\omega \in \mathbf{\Omega}$ such that $\bar{\mathrm{P}}(\omega) \leq \bar{\mathbf{P}}$, or, equivalently,

$$\bar{\mathrm{P}}_{\min} = \inf_{\omega \in \mathbf{\Omega}} \bar{\mathrm{P}}(\omega) < \bar{\mathbf{P}}. \tag{5}$$

This optimization problem is generally difficult to solve, or even to approximate by randomised methods. Here we approximate this problem by an optimisation problem with penalty terms. We show that with a proper choice of the penalty term we can enforce the desired maximum bound on the probability of violating the constraint, provided that such a bound is feasible, at the price of sub-optimality in the resulting expected performance. Let us introduce the function $u(\omega, x)$ defined as

$$u(\omega, x) = \begin{cases} \mathrm{perf}(\omega, x) + \Lambda & x \in \mathbf{X_f} \\ \\ 1 & x \notin \mathbf{X_f}, \end{cases} \tag{6}$$

where $\Lambda \geq 1$. The parameter Λ represents a reward for constraint satisfaction. The expected value of $u(\omega, x)$ is given by

$$U(\omega) = \int_{x \in \mathbf{X}} u(\omega, x) p_\omega(x) dx \qquad \omega \in \mathbf{\Omega}\,. \tag{7}$$

Instead of the constrained optimization problem (3)–(4) we solve the unconstrained optimization problem:

$$U_{\max} = \sup_{\omega \in \mathbf{\Omega}} U(\omega). \tag{8}$$

Assume the supremum is attained and let $\bar{\omega}$ denote the optimum solution, i.e. $U_{\max} = U(\bar{\omega})$. For $\bar{\omega}$ we would like to obtain bounds on the probability of violating the constraints and the level of suboptimality of $\textsc{Perf}(\bar{\omega})$ over $\textsc{Perf}_{\max|\bar{\mathbf{P}}}$. A basic bound on the probability of violating the constraint at $\bar{\omega}$ is the following.

Proposition 3.1 $\bar{\mathrm{P}}(\bar{\omega})$ *satisfies*

$$\bar{\mathrm{P}}(\bar{\omega}) \leq \frac{1 + (\Lambda - 1)\bar{\mathrm{P}}_{\min}}{\Lambda}\,. \tag{9}$$

Proof: The optimisation criterion $U(\omega)$ can be written in the form

$$\begin{aligned} U(\omega) &= \int_{x \in \mathbf{X_f}} (\mathrm{perf}(\omega, x) + \Lambda) p_\omega(x) dx + \int_{x \notin \mathbf{X_f}} p_\omega(x) dx \\ &= \textsc{Perf}(\omega) + \Lambda - (\Lambda - 1)\bar{\mathrm{P}}(\omega)\,. \end{aligned}$$

By the definition of $\bar{\omega}$ we have that $U(\bar{\omega}) \geq U(\omega)$ for all $\omega \in \mathbf{\Omega}$. We therefore can write

$$\textsc{Perf}(\bar{\omega}) + \Lambda - (\Lambda - 1)\bar{\mathrm{P}}(\bar{\omega}) \geq \textsc{Perf}(\omega) + \Lambda - (\Lambda - 1)\bar{\mathrm{P}}(\omega) \qquad \forall \omega.$$

Since $0 < \text{perf}(\omega, x) \leq 1$, $\textsc{Perf}(\omega)$ satisfies

$$0 < \textsc{Perf}(\omega) \leq P(\omega)\,. \tag{10}$$

Therefore

$$P(\bar{\omega}) - (\Lambda - 1)\bar{\mathrm{P}}(\bar{\omega}) \geq -(\Lambda - 1)\bar{\mathrm{P}}(\omega) \qquad \forall\omega$$

or, equivalently,

$$\bar{\mathrm{P}}(\bar{\omega}) \leq \frac{1 + (\Lambda - 1)\bar{\mathrm{P}}(\omega)}{\Lambda} \qquad \forall\omega$$

We obtain (9) by taking a minimum to eliminate the quantifier on the right-hand side of the above inequality. ∎

Proposition 3.1 suggests a method for choosing Λ to ensure that the solution $\bar{\omega}$ of the optimization problem will satisfy $\bar{\mathrm{P}}(\bar{\omega}) \leq \bar{\mathbf{P}}$. The following immediate corollaries make this observation more explicit.

Corollary 3.1 *Any*

$$\Lambda \geq \frac{1 - \bar{\mathrm{P}}_{\min}}{\bar{\mathbf{P}} - \bar{\mathrm{P}}_{\min}} \tag{11}$$

ensures that $\bar{\mathrm{P}}(\bar{\omega}) \leq \bar{\mathbf{P}}$.

Typically such a bound will not be useful in practice, since the value of $\bar{\mathrm{P}}_{\min}$ will be unknown. If we know that there exists a parameter $\omega \in \mathbf{\Omega}$ for which the constraints are satisfied almost surely a tighter (and potentially more useful) bound can be obtained.

Corollary 3.2 *If there exists* $\omega \in \mathbf{\Omega}$ *such that* $\bar{\mathrm{P}}(\omega) = 0$, *then any*

$$\Lambda \geq \frac{1}{\bar{\mathbf{P}}} \tag{12}$$

ensures that $\bar{\mathrm{P}}(\bar{\omega}) \leq \bar{\mathbf{P}}$.

For cases where the existence of such an ω cannot be guaranteed, it suffices to know $\bar{\mathrm{P}}(\omega)$ for some $\omega \in \mathbf{\Omega}$ with $\bar{\mathrm{P}}(\omega) < \bar{\mathrm{P}}$ to obtain a bound.

Corollary 3.3 *If there exists* $\omega \in \mathbf{\Omega}$ *for which* $\hat{\mathrm{P}} = \bar{\mathrm{P}}(\omega) \leq \bar{\mathbf{P}}$ *is known, then any*

$$\Lambda \geq \frac{1 - \hat{\mathrm{P}}}{\bar{\mathbf{P}} - \hat{\mathrm{P}}} \tag{13}$$

ensures that $\bar{\mathrm{P}}(\bar{\omega}) \leq \bar{\mathbf{P}}$.

The last bound will of course be more conservative than those of the previous two corollaries. In addition to bounds on the probability that $\bar{\omega}$ satisfies the constraints, we would also like to obtain a bound on how far the performance $\textsc{Perf}(\bar{\omega})$ is from the ideal performance $\textsc{Perf}_{\max|\bar{\mathbf{P}}}$. The following proposition provides a basic bound in this direction.

Proposition 3.2 *The performance of the maximiser, $\bar{\omega}$, of $U(\omega)$ satisfies*

$$\textsc{Perf}(\bar{\omega}) \geq \textsc{Perf}_{\max|\bar{\mathbf{P}}} - (\Lambda - 1)(\bar{\mathbf{P}} - \bar{\mathrm{P}}_{\min}). \tag{14}$$

Proof: By definition of $\bar{\omega}$ we have that $U(\bar{\omega}) \geq U(\omega)$ for all $\omega \in \mathbf{\Omega}$. In particular, we know that

$$\textsc{Perf}(\bar{\omega}) \geq \textsc{Perf}(\omega) - (\Lambda - 1)\left[\bar{\mathrm{P}}(\omega) - \bar{\mathrm{P}}(\bar{\omega})\right] \qquad \forall \omega \,:\, \bar{\mathrm{P}}(\omega) \leq \bar{\mathbf{P}}\,.$$

Taking a lower bound of the right-hand side, we obtain

$$\textsc{Perf}(\bar{\omega}) \geq \textsc{Perf}(\omega) - (\Lambda - 1)\left[\bar{\mathbf{P}} - \bar{\mathrm{P}}_{\min}\right] \qquad \forall \omega \,:\, \bar{\mathrm{P}}(\omega) \leq \bar{\mathbf{P}}\,.$$

Taking the maximum and eliminating the quantifier on the right-hand side we obtain the desired inequality. ■

Clearly to minimise the gap between the optimal performance and the performance of $\bar{\omega}$ we need to select Λ as small as possible.

This observation, together with Corollaries 3.1–3.3 leads to the following.

Corollary 3.4 *If $\bar{\mathrm{P}}_{\min}$ is known, the choice*

$$\Lambda = \frac{1 - \bar{\mathrm{P}}_{\min}}{\bar{\mathbf{P}} - \bar{\mathrm{P}}_{\min}}$$

ensures that $\bar{\mathrm{P}}(\bar{\omega}) \leq \bar{\mathbf{P}}$ and $\textsc{Perf}(\omega) \geq \textsc{Perf}_{\max|\bar{\mathbf{P}}} + \bar{\mathbf{P}} - 1$. The same bound is achieved if an $\omega \in \mathbf{\Omega}$ such that $\bar{\mathrm{P}}(\omega) = 0$ is known to exist by setting $\Lambda = \frac{1}{\bar{\mathbf{P}}}$.

Corollary 3.5 *If there exists $\omega \in \mathbf{\Omega}$ for which $\hat{\mathrm{P}} = \bar{\mathrm{P}}(\omega)$ is known and $\hat{\mathrm{P}} < \bar{\mathbf{P}}$, then the choice*

$$\Lambda = \frac{1 - \hat{P}}{\bar{\mathbf{P}} - \hat{\mathrm{P}}}$$

ensures that $\bar{\mathrm{P}}(\bar{\omega}) \leq \bar{\mathbf{P}}$ and

$$\textsc{Perf}(\bar{\omega}) \geq \textsc{Perf}_{\max|\bar{\mathbf{P}}} - \frac{\bar{\mathbf{P}} - \bar{\mathrm{P}}_{\min}}{\bar{\mathbf{P}} - \hat{\mathrm{P}}}[1 - \bar{\mathbf{P}}].$$

3.2 The Metropolis-Hastings algorithm

Metropolis-Hastings is an algorithm to generate extractions from a desired arbitrary distribution $h(\omega)$ by simulating a Markov Chain $\Omega(k)$ whose stationary distribution is $h(\omega)$. In the algorithm, $g(\omega|\bar{\omega})$ is an instrumental (or *proposal*) distribution, freely chosen by the user. The only requirement is that $g(\omega|\bar{\omega})$ covers the support of $h(\omega)$.

Metropolis-Hastings algorithm

Given $\omega(k)$:

1 Extract $\tilde{\Omega} \sim g\left(\omega|\omega(k)\right)$

2 Extract the new state of the chain as

$$\omega(k+1) = \begin{cases} \tilde{\Omega} & \text{with probability} \quad \rho(\omega(k), \tilde{\Omega}) \\ \omega(k) & \text{with probability} \quad 1 - \rho(\omega(k), \tilde{\Omega}) \end{cases}$$

where

$$\rho(\omega, \tilde{\omega}) = \min\left\{1, \frac{h(\tilde{\omega})}{h(\omega)} \frac{g(\omega|\tilde{\omega})}{g(\tilde{\omega}|\omega)}\right\} \tag{15}$$

A well-known and, in a large number of cases, very useful property of the Metropolis-Hastings algorithm is that the desired distribution $h(\omega)$ can also be known only up to a scaling factor which, in fact, would be irrelevant in the calculation of $\rho(\omega, \tilde{\omega})$.

Under very general conditions the stationary distribution of the chain produced by the algorithm is $h(\omega)$. If we denote by $\mathbf{\Omega}$ the support of $h(\omega)$, we have that for every conditional distribution $g(\omega|\bar{\omega})$ whose support include $\mathbf{\Omega}$, $h(\omega)$ is a stationary distribution of the chain produced by the Metropolis-Hastings algorithm. This result can be easily proved by calculating the kernel of the Markov Chain $\Omega(k)$ which is:

$$\kappa(\bar{\omega}, \omega) = \rho(\bar{\omega}, \omega) g(\omega|\bar{\omega}) + \left[1 - \int_{\mathbf{\Omega}} \rho(\bar{\omega}, z) g(z|\bar{\omega}) dz\right] \delta_{\bar{\omega}}(\omega)\,. \tag{16}$$

Then, it can be shown that $\kappa(\bar{\omega}, \omega)$ satisfies the *detailed balance condition* $\kappa(\bar{\omega}, \omega) h(\bar{\omega}) = \kappa(\omega, \bar{\omega}) h(\omega)$ which implies that $h(\omega)$ is the stationary distribution of the chain. Notice that, even if the search distribution $g(\bar{\omega}|\omega)$ changes at each step due to the conditioning with respect to the current state of the chain, the kernel of the chain is homogeneous. The ergodicity of $\Omega(k)$ (i.e. convergence in distribution to the stationary distribution) can also be proved with minimal assumptions.

In case $g(\omega|\bar{\omega}) = g(\omega)$ (i.e. no conditioning is considered), the following result on the convergence rate to stationary distribution is available.

Proposition 3.3 [19, Theorem 6.3.1] *The Markov Chain $\Omega(k)$ is uniformly ergodic if*

$$M = \sup_{\omega \in \mathbf{\Omega}} \frac{h(\omega)}{g(\omega)} \tag{17}$$

is finite. In this case, let $K^n(\omega, \cdot\,)$ denote the n-transitions kernel of $\Omega(k)$, then

$$\|K^n(\omega, \cdot\,) - h(\cdot)\|_{TV} \leq 2\left(1 - \frac{1}{M}\right)^n \qquad \forall \omega \in \mathbf{\Omega} \tag{18}$$

where $\|\cdot\|_{TV}$ denotes the total variation norm.

In case of a general proposal distribution $g(\omega|\bar{\omega})$ there are instead no general results and, though the $\Omega(k)$ is ergodic by construction, in some cases it could even be not uniformly ergodic.

3.3 Simulated Annealing

Simulated Annealing is a randomised strategy for approximate global optimisation of a deterministic criterion $C(\omega)$[22]. It relies on extractions of a random variable Ω whose distribution has modes which coincide with the optimal points of $C(\omega)$. These extractions are obtained through Monte Carlo Markov Chain simulation [19]. The problem of optimising the expected criterion is then reformulated as the problem of estimating the optimal points from extractions concentrated around them.

To introduce Simulated Annealing we consider the case in which ω is a discrete set. In particular, denote $\mathbf{\Omega}_{opt} = \{\omega_i^{opt}, i = 1, \ldots, \mathcal{R}_{opt}\}$ the set of global minimisers of $C(\omega_i)$, then the essence of Simulated Annealing is the construction of a Markov Chain with stationary distribution $h_c(\omega_i)$, with tunable parameter c, that converges, as c tends to 0, to a uniform distribution on the set $\mathbf{\Omega}_{opt}$, i.e.

$$\lim_{c \to 0} h_c(\omega_i) = \pi_i \tag{19}$$

where

$$\pi_i = \begin{cases} \mathcal{R}_{opt}^{-1} & \text{if} \quad \omega_i \in \mathbf{\Omega}_{opt} \\ 0 & \text{elsewhere} \end{cases} . \tag{20}$$

Sufficient conditions for $h_c(\omega_i)$ to satisfy (19) and (20) are given in the following theorem.

Theorem 3.1 **[20]** *Let $h_c(\omega_i)$ be in the form*

$$h_c(\omega_i) = \frac{\psi(C(\omega_i), c)}{\sum_{j=1}^{\mathcal{R}} \psi(C(\omega_j), c)} \tag{21}$$

where $\psi(C, c)$ is a two arguments function satisfying the following three conditions

$$\lim_{c \to 0} \psi(C, c) = \begin{cases} 0 & \textit{if} \quad C > 0 \\ \infty & \textit{if} \quad C < 0 \end{cases} \tag{22a}$$

$$\frac{\psi(C_1, c)}{\psi(C_2, c)} = \psi(C_1 - C_2, c) \tag{22b}$$

$$\psi(0, c) = 1 \quad \forall c > 0 \tag{22c}$$

then $h_c(\omega_i)$ satisfies (19) (20).

The statement of the above theorem can be easily proved by noticing that $h_c(\omega_i)$ for $\omega_i \in \mathbf{\Omega}_{opt}$, by using properties (22b-22c), can be also written as

$$h_c(\omega_i) = \frac{1}{\mathcal{R}_{opt} + \sum_{j:\, \omega_j \in \mathbf{\Omega} - \mathbf{\Omega}_{opt}} \psi(C(\omega_j) - C(\omega_i), c)} .$$

Then, since the terms $C(\omega_j) - C(\omega_i)$ are positive, we obtain that the sum in the above expression goes to 0 as $c \to 0$ due to (22a). We can write a similar expression for the case $\omega_i \notin \mathbf{\Omega}_{opt}$ and in this case we obtain a denominator that tends to infinity as $c \to 0$. Eventually we obtain exactly conditions (19) and (20).

In the original formulation of Simulated Annealing, the following distribution was adopted:

$$h(\omega_i, c) = \frac{e^{\frac{C(\omega_i)}{c}}}{\sum_{j=1}^{\mathcal{R}} e^{\frac{C(\omega_i)}{c}}} . \tag{23}$$

Since $h_c(\omega_i)$ can be evaluated point-wise up to the normalising factor, the construction of the Markov Chain with the desired stationary distribution $h_c(\omega_i)$ can be performed through Metropolis-Hastings algorithm. From equations (19) and (20) we obtain that for a sufficiently small c and for a sufficient long simulation time the Ωs obtained by Markov Chain simulation will be concentrated around the global minimisers.

In practice, in the application of the optimisation algorithm, the parameter c must be decreased during the optimisation procedure. Therefore a sequence $c(k)$ satisfying

$$\lim_{k \to \infty} c(k) = 0; \tag{24a}$$

$$c(k) \geq c(k+1)\, k = 0, 1, \ldots \tag{24b}$$

is usually assumed. In doing so, care must be exercised in order to maintain convergence. In fact the Markov Chain becomes inhomogeneous and the previous convergence results do not apply straightforwardly.

A number of conditions, implying convergence of the inhomogeneous chain to the stationary distribution, have been derived for the form:

$$c(k) = \frac{\Gamma}{\log(k)} \qquad \forall k \geq 0 .$$

where the constant Γ depends on the particular structure of the problem [22]. As an example, in the basic case of target distribution (23) with uniform proposal distribution, among the neighbours states of the current one, a possible value for Γ is

$$\Gamma = \mathcal{R} \Delta C_{max}$$

where $\Delta C_{max} = \max\{C(\omega_i)\} - \min\{C(\omega_i)\}$.

3.4 Optimisation of an expected value criterion

In this section we recall a simulation-based procedure, to find approximate maximisers of $U(\omega)$, which has been defined in (7). The only requirement for applicability of the procedure is the possibility of obtaining realisations of the random variable X with distribution

$p_\omega(x)$ and of evaluating $u(\omega, x)$ pointwise, no other particular assumptions are imposed on the optimisation criterion. This optimisation procedure is in fact a general procedure for the optimisation of expected value criteria. It has been originally proposed in Bayesian statistics literature [16]. This approach can be seen as the counterpart of Simulated Annealing for a stochastic setting. A formal parallel between these two strategies has been derived in [17].

The problem of maximising $U(\omega)$ can be equivalently stated as the problem of minimising $\ln[U(\omega)]$. Notice that $U(\omega) > 0\,\forall\omega$ since $u(x, \omega) > 0 \quad \forall\omega,\, x$. In order to apply Simulated Annealing techniques, we consider a target distribution defined by:

$$h_c(\omega) \propto e^{\frac{1}{c}\ln(U(\omega))} = U(\omega)^{1/c} \tag{25}$$

In the remainder of this section we show that it is in fact possible to construct such a Markov Chain $\Omega(k)$ with the desired target distribution through the Metropolis-Hastings algorithm.

The optimisation procedure can be described as follows. Consider a stochastic model formed by a random variable Ω, whose distribution has not been defined yet, and J conditionally independent replicas of random variable X with distribution $p_\Omega(x)$. Let us denote by $h(\omega, x_1, x_2, \dots, x_J)$ the joint distribution of $(\Omega, X_1, X_2, X_3, \dots, X_J)$. It is straightforward to see that if

$$h(\omega, x_1, x_2, \dots, x_J) \propto \prod_j u(\omega, x_j) p_\omega(x_j) \tag{26}$$

then the marginal distribution of Ω, say $h(\omega)$, satisfies

$$h(\omega) \propto \left[\int u(\omega, x) p_\omega(x) dx\right]^J = U(\omega)^J\,. \tag{27}$$

This means that if we can extract realisations of $(\Omega, X_1, X_2, X_3, \dots, X_J)$ then the extracted Ω's will be concentrated around the optimal points of $U(\Omega)$ for a sufficiently high J. These extractions can be used to find an approximate solution to the optimisation of $U(\omega)$.

Realisations of the random variables $(\Omega, X_1, X_2, X_3, \dots, X_J)$, with the desired joint probability density given by (26), can be obtained through Monte Carlo Markov Chain simulation. The algorithm is presented below. In the algorithm, $g(\omega|\omega(k))$ is known as the instrumental (or *proposal*) distribution and is freely chosen by the user; the only requirement is that $g(\omega|\omega(k))$ covers the support of $h(\omega)$.

MCMC algorithm

Given $\omega(k)$, $x_j(k)$, $j = 1, \dots, J$ realisations of random variable $X(k)$ with distribution $p_{\omega(k)}(x)$, and $u_J(k) = \prod_{j=1}^{J} u(\omega(k), x_j(k))$:

1 Extract

$$\tilde{\Omega} \sim g(\omega|\omega(k))$$

2 Extract

$$\tilde{X}_j \sim p_{\tilde{\Omega}}(x) \quad j = 1, \ldots, J$$

and calculate

$$\tilde{U}_J = \prod_j u(\tilde{\Omega}, \tilde{X}_j)$$

3 Extract the new state of the chain as

$$[\omega(k+1),\, u_J(k+1)] = \begin{cases} [\tilde{\Omega},\, \tilde{U}_J] & \text{with probability } \rho(\omega(k), u_J(k), \tilde{\Omega}, \tilde{U}_J) \\ \\ [\omega(k),\, u_J(k)] & \text{with probability } 1 - \rho(\omega(k), u_J(k), \tilde{\Omega}, \tilde{U}_J) \end{cases}$$

where

$$\rho(\omega, u_J, \tilde{\omega}, \tilde{u}_J) = \min\left\{1, \frac{\tilde{u}_J}{u_J}\frac{g(\omega|\tilde{\omega})}{g(\tilde{\omega}|\omega)}\right\}$$

This algorithm is a formulation of the Metropolis-Hasting algorithm for a desired distribution given by $h(\omega, x_1, x_2, \ldots, x_J)$ and proposal distribution given by

$$g(\omega|\omega(k)) \prod_j p_\omega(x_j)\,.$$

In this case, the acceptance probability for the standard Metropolis-Hastings algorithm is

$$\frac{h(\tilde{\omega}, \tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_J)}{h(\omega, x_1, x_2, \ldots, x_J)} \frac{g(\omega|\tilde{\omega}) \prod_j p_\omega(x_j)}{g(\tilde{\omega}|\omega) \prod_j p_\omega(\tilde{x}_j)}\,.$$

By inserting (26) in the above expression one obtains the probability $\rho(\omega, u_J, \tilde{\omega}, \tilde{u}_J)$. Under minimal assumptions, the Markov Chain $\Omega(k)$ is uniformly ergodic with stationary distribution $h(\omega)$ given by (27). Results that characterise the convergence rate to the stationary distribution can be found for example in [19].

A general guideline to obtain faster convergence is to concentrate the search distribution $g(\omega)$ where $U(\omega)$ assumes nearly optimal values. The algorithm represents a trade-off between computational effort and the "peakedness" of the target distribution. This trade-off is tuned by the parameter J which is the power of the target distribution and also the number of extractions of X at each step of the chain. Increasing J concentrates the distribution more around the optimisers of $U(\omega)$, but also increases the number of simulations one needs to perform at each step. Obviously if the peaks of $U(\omega)$ are already quite sharp, this implies some advantages in terms of computation, since there is no need to increase further the peakedness of the criterion by running more simulations. For the specific $U(\omega)$ proposed in Section 3.1, a trade-off exists between its peakedness and the parameter Λ, which is related to probability of constraint violation. In particular, the greater Λ is the less peaked the criterion $U(\omega)$ becomes, because the relative variation of $u(\omega, x)$ is reduced, and therefore more computational effort is required for the optimisation of $U(\omega)$.

3.5 Implementation and improvements

We are currently working on the following aspects:

- In order to improve the performance, allow parallel implementation and introduce recursion we are currently investigating the particle implementation of the algorithm presented above. Here we recall a particle version of the algorithm as derived in [1]. The particle implementation of the algorithm is defined by the interaction of N chains through an importance sampling and selection step (1-4) followed by the usual Metropolis step (5-7).

  MCMC algorithm (particle implementation)
  Given $\omega_i(k)$, $i = 1, \dots, N$:

  1 Extract
  $$\tilde{\Omega}_i \sim g_{IS}(\omega|\omega_i(k)) \quad i = 1, \dots, N\,.$$

  2 Extract
  $$\tilde{X}_{ij} \sim p_{\tilde{\Omega}_i}(x) \quad j = 1, \dots, J\,;\, i = 1, \dots, N\,.$$
  and calculate
  $$\tilde{U}_i^J = \prod_j u(\tilde{\Omega}_i, \tilde{X}_{ij})\,.$$

  3 Calculate the normalised weights
  $$W_i \propto \frac{\tilde{U}_i^J}{g_{IS}\left(\tilde{\Omega}_i|\omega_i(k)\right)}\,.$$

  4 Resample $(\hat{\Omega}_1, \dots \hat{\Omega}_N)$ from $(\tilde{\Omega}_1, \dots \tilde{\Omega}_N)$ according to a multinomial distribution with weights W_i.

  5 Extract
  $$\bar{\Omega}_i \sim g_{MH}(\omega|\hat{\Omega}_i) \quad i = 1, \dots, N\,.$$

  6 Extract
  $$\bar{X}_{ij} \sim p_{\bar{\Omega}_i}(x) \quad j = 1, \dots, J\,;\, i = 1, \dots, N\,.$$
  and calculate
  $$\bar{U}_i^J = \prod_j u(\bar{\Omega}_i, \bar{X}_{ij})\,.$$

  7 For each $i = 1, \dots, N$ extract the new states of the chains as
  $$\Omega_i(k+1) = \begin{cases} \bar{\Omega}_i & \text{with probability } \rho(\hat{\Omega}_i, \hat{U}_i^J, \bar{\Omega}_i, \bar{U}_i^J) \\ \hat{\Omega}_i & \text{with probability } 1 - \rho(\hat{\Omega}_i, \hat{U}_i^J, \bar{\Omega}_i, \bar{U}_i^J) \end{cases}$$

where

$$\rho(\hat{\Omega}, \hat{U}^J, \bar{\Omega}, \bar{U}^J) = \min\left\{1, \frac{\bar{U}^J}{\hat{U}^J}\frac{g_{MH}(\hat{\Omega}|\bar{\Omega})}{g_{MH}(\bar{\Omega}|\hat{\Omega})}\right\}$$

We are currently testing this version of the algorithm for optimisation problems in conflict resolution. We are also investigating on the development of the particle implementation in order to obtain a recursive optimisation procedure for receding horizon optimisation problems, by suitable definition of the importance sampling step [4]

- A penalty formulation of the problem has been considered here which guarantees constraint satisfaction but delivers a suboptimal solution. Our current research is concerned with overcoming the suboptimality imposed by the need to provide constraint satisfaction guarantees. A possible way is to use the Monte Carlo Markov Chain procedure to obtain optimisation parameters that satisfy the constraint and to optimise over this set in a successive step.

- The use of the optimisation algorithm presented in this deliverable will be illustrated in Deliverable D5.4 in simulation examples regarding sequencing of aircraft in Terminal Airspace and approach maneuvers in Approach Sectors.

4 Dynamic programming

An alternative approach to efficiently computing conflict resolution maneuvers based on the optimal control formulation is to consider a dynamic programming approach. Dynamic programming requires one to compute the value function of the optimal control problem: roughly, how much is it worth to be in a certain state at a certain time, or equivalently, how much cost/reward will be accumulated when going optimally from the given state and time to the terminal time. The "gradient" of the value function can then be used to compute the optimal controls.

In general, the dynamic programming approach suffers from the so called curse of dimensionality. The computation of the value function when the state space is continuous typically requires gridding the state space; since the number of grid points grows exponentially in the dimension of the space this approach scales very badly. An alternative approach is to approximate the value function by a function in a certain well behaved class which does not require gridding. A class of functions commonly used in practice for such approximations are functions generated by neural networks. This approach gives rise to the so called neuro-dynamic programming [2]. In this section we adapt this approach to the conflict resolution problem.

Our data consists of the initial position and the orientation of each aircraft. Also, the goal of each aircraft is known and that means that we know the final aircraft's destination (position and orientation). In addition we have a time line in which each aircraft is necessary to complete its trajectory.

In this part of the document, we must clarify our assumptions. We have solved the problem in the two dimensions, which means that all aircraft are moving on the same plane. So, the aircraft can't move down and up in order to avoid the conflict with another aircraft. The only move which is allowable is turn left and right and move forth (apart from the take-off and the landing).

Therefore, our ultimate goal is to determine the way-points of each aircraft, in other words the optimal trajectory, without conflict and without unnecessary annoyance for the passengers.

As we have investigated, the analytical solution of the optimization problem will had been pointless due to the nonlinear nature of the equations that describe the problem. So, our interest turns to finding a numerical solution to this problem, using machinery from the field of Nonlinear Programming. After suitable formulation of the problem, we have solved it for several scenarios. This procedure takes time, so it is important to find a methodology faster than the present methodology. We decided to apply the methodology of Neural Networks. We have collected all the data from nonlinear programming procedure in order to have the necessary training data for the neuro-dynamic procedure.

4.1 Air Traffic Modelling for dynamic programming

We know the initial conditions which include the initial and the final position and orientation of each aircraft q_{io}, q_{if}, respectively, and the time between the transition t.

$$q_{i0}(t) = [x_{i0}(t) \quad y_{i0}(t) \quad \vartheta_{i0}(t)]^T$$

$$q_{if}(t) = [x_{if}(t) \quad y_{if}(t) \quad \vartheta_{if}(t)]^T$$

where x, y are the Cartesian coordinates of aircraft, ϑ is the orientation, $i = 1 \dots N$ and N is the number of aircraft.

We can define the state variables of our problem which are as follows:

$$q_i(t) = [x_i(t) \quad y_i(t) \quad \vartheta_i(t)]^T$$

Also, the control variables of the problem are the linear velocity and the angular velocity of each aircraft.

$$u_i(t) = [V_i(t) \quad \omega_i(t)]^T$$

Furthermore, the problem is described from the kinematics equations, which are based on the relationship between the velocities and the states of a unicycle in a plane.

$$\dot{x}_i(t) = V_i \cdot \cos\vartheta_i$$

$$\dot{y}_i(t) = V_i \cdot \sin\vartheta_i$$

$$\dot{\vartheta}_i(t) = \omega_i$$

In its general description the problem could be written as:

$$\dot{q}(t) = f(q(t), u_1(t), \dots, u_N(t))$$

where

$$\dot{q}(t) = [\dot{q}_1(t), \dots, \dot{q}_N(t)]^T$$

and

$$f(q(t), u_1(t), \dots, u_N(t)) = A(q) \cdot u$$

with

$$A(q) = \begin{bmatrix} \cos\vartheta_1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \sin\vartheta_1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & & & \ddots & & & \vdots \\ 0 & & & & 0 & \cos\vartheta_N & 0 \\ 0 & & & & 0 & \sin\vartheta_N & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$$

Another important issue, in order to solve the problem, is the constrains which the movement of each aircraft must satisfy. Such constrains are the upper and the lower bound of the linear velocity and the angular velocity of aircraft.

$$V_{min} \leq V_i(t) \leq V_{max}$$

$$-\omega_o \leq \omega_i(t) \leq \omega_o$$

where the bounds V_{max}, V_{min} and ω_o are taken from the BADA database [6].

There are also constrains which arise from the protection zones around the aircraft. The constrain requires that the distance between two aircraft should not be less than a desired separation minimum d.

$$((x_i(t) - x_j(t))^2 + (y_i(t) - y_j(t))^2) \geq d^2$$

where x, y are the coordinates of aircraft at the same time instant, and $i, j = 1, \ldots, N$ with $i \neq j$.

4.2 Non-Linear Programming

In a nonlinear programming problem it is necessary to define the function which we would like to minimize or to maximize and constrains which describe the problem.

In our problem, we would like to evaluate the optimal control law which will minimize the function presented below. The aim of this function is to save the energy (we do not want wide fluctuations in the aircraft velocity and tight turns).

$$\min_u J = \min_u \left[\frac{1}{2} \cdot \sum_{i=1}^{n} \left(q_i^T \cdot Q \cdot q_i + u_i^T \cdot R \cdot u_i \right) \cdot \Delta t \right]$$

The matrices, in the above equation, are constant symmetric matrices and additionally Q is a positive semi-definite matrix and R is a positive definite matrix.

Also, we need to define the constrains of the problem. In order to solve the problem numerically it is important to transfer it in its discrete form. We must write the kinematic equations in a discrete time form by integration between time instants.

$$\left.\begin{array}{c} x_{k+1} = \frac{V_k}{\omega_k} \cdot (\sin(\omega_k \cdot \Delta t + \vartheta_k) - \sin(\vartheta_k)) + x_k \\ y_{k+1} = \frac{V_k}{\omega_k} \cdot (\cos(\vartheta_k) - co\,s(\omega_k \cdot \Delta t + \vartheta_k)) + y_k \\ \vartheta_{k+1} = \omega_k \cdot \Delta t + \vartheta_k \end{array}\right\}$$

When the value of angular velocity ω_k is very small this discrete time form becomes:

$$\left.\begin{aligned} x_{k+1} &= V_k \cdot \cos(\vartheta_k) \cdot \Delta t + x_k \\ y_{k+1} &= V_k \cdot \sin(\vartheta_k) \cdot \Delta t + y_k \\ \vartheta_{k+1} &= \omega_k \cdot \Delta t + \vartheta_k \end{aligned}\right\}$$

The other constrains are as we have defined above (linear and angular velocities, separation minimum constrains). The state variables of the system which we would like to find are:

$$q_{ij} = \left[\begin{array}{ccc} x_{ij} & y_{ij} & \vartheta_{ij} \end{array}\right]^T$$

where $i = 1, \ldots, N$, and $j = 2, \ldots, (n-1)$ the number of time steps. The behavior of subscript j is different because the values of the state variables are known at the first and at the last time instant.

Furthermore, we would like to evaluate the control variables for each time instant.

$$u_{ij} = \left[\begin{array}{cc} V_{ij} & \omega_{ij} \end{array}\right]^T$$

In this equation the subscript j takes all the values $j = 1, \ldots, n$.

4.3 Neural Networks

Suppose that we are interested in approximating a function $J : S \rightarrow \mathbb{R}$, where S is some set, S will usually be the state space of a dynamic programming problem and J will be the optimal cost-to-go function. Supposing r be the vector of parameters and we claim the functional form $J(k) \approx \tilde{J}(k, r)$. The architecture which we have chosen is described by the function $\tilde{J}$ which is a known function and its value is easy to estimate for fixed vector r values. Thereinafter, it is necessary to decide the values of the parameter vector r, in order to achieve the minimum distance between the function J that we are trying to fit and the approximant $\tilde{J}$.

We have used a nonlinear architecture in which the dependence of $\tilde{J}(k, r)$ on r is nonlinear. In this case, we can use the data sets in order to solve the least squares problem of minimizing

$$\sum_k \left(J(k) - \tilde{J}(k, r)\right)^2$$

The nonlinear architecture which we used is the multilayer perceptron or feed-forward neural network with a single hidden layer. Under this architecture, the state k is encoded as a vector X with components $x_p(k)$, $p = 1, \ldots, L$, which is then transformed linearly through a "linear layer", involving the coefficients $r_1\,(j, p)$, to give the λ scalars

$$\sum_{p=1}^{L} r_1\,(j,p) \cdot x_p\,(k), \qquad j = 1, \ldots, \lambda$$

Each of these scalars becomes the input to a function σ, called a sigmoidal function, which is differentiable, monotonically increasing and has the property:

$$-\infty < \lim_{\xi \to -\infty} \sigma\,(\xi) < \lim_{\xi \to \infty} \sigma\,(\xi) < \infty$$

One common choice is the hyperbolic tangent function:

$$\sigma\,(\xi) = \tanh\,(\xi) = \frac{e^{\xi} - e^{-\xi}}{e^{\xi} + e^{-\xi}}$$

At the output of the sigmoidal functions, the scalars

$$\sigma\left(\sum_{p=1}^{L} r_1\,(j,p) \cdot x_p\,(k)\right), \text{ j} = 1, \ldots, \lambda$$

are obtained. These scalars are linearly combined using coefficients $r_2(j)$ to produce the final output:

$$\tilde{J}\,(k, r) = \sum_{j=1}^{\lambda} r_2\,(j) \cdot \sigma\left(\sum_{p=1}^{L} r_1\,(j,p) \cdot x_p\,(k)\right).$$

The parameter vector r consists of the coefficients $r_2(j)$ and $r_1(j, p)$, which are also known as the weights of the network.

Our goal is to train the neural network, i.e. to evaluate the parameter vector r as with a given input in the network to take the respective output. The important issue at this point is to choose the variables of the input and the output. We decided to use as an input $x_p(k)$ of the network the known parameters of beginning and final position and orientation of each aircraft and the time of interest. As an output $y_i(k)$ we defined the control variables of each aircraft at every time instant.

We let the data sets the optimal solutions for a various scenarios from the nonlinear programming problem and we want to minimize the distance between the output of the network and the real values of the control variables.

$$\min_r f\,(r) = \min_r \left(\frac{1}{2} \cdot \sum_{k=1}^{m} \|g_k\,(r)\|^2\right)$$

where

$$g_k\,(r) = (u_k - r_2 \cdot \sigma\,(r_1 \cdot X_k))$$

is a continuously differentiable function, m is the number of training data, and u_k is the vector with the optimal control variables.

References

[1] B Amzal, F.Y. Bois, E. Parent, and C.P. and Robert. Bayesian optimal design via interacting mcmc. Technical report, Cahiers du Ceremade, 2003. Available from World Wide Web: `http://www.ceremade.dauphine.fr/~xian/publications.html`.

[2] D.P. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, 1996.

[3] M.H.A. Davis. *Markov Models and Optimization.* Chapman & Hall, 1993.

[4] A. Doucet, N de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice.* Springer-Verlag, 2001.

[5] A. Doucet, N.J. Gordon, and V. Krishnamurty. Particle Filters for State Estimation of Jump Markov Linear Systems. *IEEE Transactions on Signal Processing*, 49(3), 2001.

[6] EUROCONTROL Experimental Centre. *User Manual for the Base of Aircraft Data (BADA) — Revision 3.3.* 2002. Available from World Wide Web: `http://www.eurocontrol.fr/projects/bada/`.

[7] E. Frazzoli, Z.H. Mao, J.H. Oh, and E. Feron. Aircraft conflict resolution via semidefinite programming. *AIAA Journal of Guidance, Control, and Dynamics*, 24(1):79–86, 2001.

[8] J. Hu, J. Lygeros, and S. Sastry. Towards a theory of stochastic hybrid systems. In *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 160–173. Springer-Verlag, 2000.

[9] J. Hu, M. Prandini, and S. Sastry. Optimal Coordinated Maneuvers for Three-Dimensional Aircraft Conflict Resolution. *AIAA Journal of Guidance, Control and Dynamics*, 25(5), 2002.

[10] E.C. Kerrigan and J.M. Maciejowski. Robustly stable model predictive control using a single linear program. *Int. Jnl. of Robust and Nonlinear Control*, 2004.

[11] B. Kouvaritakis, M. Cannon, and V. Tsachouridis. Recent developments in stochastic MPC and sustainable development. *Annual Reviews in Control*, 28(1):23–35, 2004.

[12] J.K. Kuchar and L.C. Yang. A review of conflict detection and resolution methods. *IEEE Transactions on Intelligent Transportation Systems*, 1(4):179–189, 2000.

[13] A. Lecchini, W. Glover, J. Lygeros, and J. Maciejowski. Air-Traffic Control in Approach Sectors: simulation examples and optimisation. 2004. Available from World

Wide Web: http://www-control.eng.cam.ac.uk/~al394/. Accepted at HSCC: Hybrid Systems Computation and Control, Zurich, 2005.

[14] A. Lecchini, W. Glover, J. Lygeros, and J. Maciejowski. Air Traffic Control with an expected value criterion. Technical Report WP5, Deliverable D5.2, HYBRIDGE, 2004. Available from World Wide Web: http://www-control.eng.cam.ac.uk/~al394/. Submitted to IFAC World Congress 2005.

[15] J.M. Maciejowski. *Predictive Control with Constraints.* Prentice-Hall, 2002.

[16] P. Mueller. Simulation based optimal design. In *Bayesian Statistics 6*, pages 459–474. J.O. Berger, J.M. Bernardo, A.P. Dawid and A.F.M. Smith (eds.), Oxford University Press, 1999.

[17] P. Mueller, B. Sanso, and M. De Iorio. Optimal Bayesian design by inhomogeneous Markov chain simulation. Technical report, 2003. Available from World Wide Web: http://www.ams.ucsc.edu.

[18] M. Prandini, J. Hu, J. Lygeros, and S. Sastry. A probabilistic framework for aircraft conflict detection. *IEEE Transactions on Intelligent Transportation Systems*, 1(4):199–220, December 2000.

[19] C.P. Robert and G. Casella. *Monte Carlo Statistical Methods.* Springer-Verlag, 1999.

[20] F. Romeo and A.L. Sangiovanni-Vincentelli. Probabilistic hill climbing algorithms: Properties and applications. In *1985 Chaple Hill Conference on VLSI*, Chapel Hill, USA, 1985.

[21] C. Tomlin, G. Pappas, and S. Sastry. Conflict resolution for Air Traffic Management: a case study in multi-agent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):509–521, 1998.

[22] P.J.M. van Laarhoven and E.H. Aarts. *Simulated Annealing: Theory and Applications.* D.Reidel Publishing Company, 1987.