"Optimal measurement times for particle filtering and its application in mobile tumor tracking"

Gouverneur, Amaury

ABSTRACT

Particle filter is a powerful algorithm to track the state of discrete dynamic systems from noisy measurements. Under certain circumstances, the number of measurements has to be restricted. In this case, one is interested in finding the optimal measurement time set for particle filtering, either apriori, i.e. before any measurement acquisition, or online, i.e. as the measurements are acquired. These problems are referred to respectively as the apriori problem and the online problem and arise for instance, in the domain of mobile tumor tracking based on X-ray images, where X-ray acquisitions have to be made in a parsimonious way to limit patients' exposure to harmful radiations. This work proposes to address these two problems by designing an algorithm that finds near-optimal measurement time sets. The developed algorithm is based on the nesting of a genetic algorithm, a Monte Carlo algorithm, and a particle filter. An application in mobile tumor tracking is presented and the performance of the method is measured on a simplified lung tumor model. In comparison with measurements performed at regular time intervals, the measurement time set solving the apriori problem reduces the expected mean squared tracking error by 37.5%. The expected tracking performance is improved by 39.8% using the measurement time set solving the online problem with a significantly smaller computational budget.

CITE THIS VERSION

UCLouvain

École polytechnique de Louvain

Optimal Measurement Times For Particle Filtering

and its application in mobile tumor tracking

Author: **Amaury Gouverneur**
Supervisors: **Benoît Macq, Antoine Aspeel**
Readers: **Raphaël Jungers, Laurent Jacques**
Academic year 2019–2020
Master [120] in Mathematical Engineering

Abstract

Particle filter is a powerful algorithm to track the state of discrete dynamic systems from noisy measurements. Under certain circumstances, the number of measurements has to be restricted. In this case, one is interested in finding the optimal measurement time set for particle filtering, either *apriori*, i.e. before any measurement acquisition, or *online*, i.e. as the measurements are acquired. These problems are referred to respectively as the *apriori problem* and the *online problem* and arise for instance, in the domain of mobile tumor tracking based on X-ray images, where X-ray acquisitions have to be made in a parsimonious way to limit patients' exposure to harmful radiations. This work proposes to address these two problems by designing an algorithm that finds near-optimal measurement time sets. The developed algorithm is based on the nesting of a genetic algorithm, a Monte Carlo algorithm, and a particle filter. An application in mobile tumor tracking is presented and the performance of the method is measured on a simplified lung tumor model. In comparison with measurements performed at regular time intervals, the measurement time set solving the *apriori problem* reduces the expected mean squared tracking error by 37.5%. The expected tracking performance is improved by 39.8% using the measurement time set solving the *online problem* with a significantly smaller computational budget.

Acknowledgments

I wish to thank, first and foremost, professor Benoît Macq. This work was made possible thanks to the numerous meeting at his office, his precious help and his unwavering optimism.

I owe my deepest gratitude to Antoine Aspeel for his collaboration. The work presented in this master thesis is equally his. Thank you for your collaboration, your enthusiasm and your generosity.

I would also like to thank professor Raphaël Jungers, for his insightful comments, his guidance, and his constructive criticism which helped improve the quality of this work.

Writing my master's thesis required more than academic assistance, and I wish to thank my parents and my brother for their support and encouragement throughout my study. They never stopped believing in me and for this, I am forever grateful.

Finally, I would like to thank my partner, Ebba, with whom I shared the good times and the hard times, and, most importantly, who made me believe in myself.

Contents

List of Notations and Acronyms **v**

List of Figures **vii**

List of Tables **viii**

1 Introduction **1**

2 Theoretical background **3**
2.1 Some fundamental concepts of probability 3
2.2 Overview of the Monte Carlo method 6
2.3 Tracking problem 7
2.4 Sequential Monte Carlo method 8
2.4.1 Sequential Importance Sampling 9
2.4.2 Sequential Importance Sampling with Resampling 11
2.4.3 The particle filter 12

3 Materials and methods **14**
3.1 Filtering problem with intermittent measurements 14
3.2 Particle filter tracking with intermittent measurements 15
3.3 An optimality criteria for measurement time sets 18
3.4 The expected mean squared tracking error estimator and the *apriori* problem 18
3.5 Adaptation to *online* measurements acquisition 20
3.6 Five combinatorial optimization algorithms to solve the *apriori* problem 22
3.6.1 The Greedy Forward algorithm 22
3.6.2 The Greedy Backward algorithm 24
3.6.3 The Simulated Annealing algorithm 25
3.6.4 The Genetic algorithm 27
3.6.5 The Random Trial algorithm 28

3.7 A note on the *training model* and the robustness of the optimal measurement time set 28

4 Results **30**
4.1 Evaluation of the performance of a measurement time set $\mathcal{M}$ 30
4.1.1 Performance of a measurement time set and statistics on the relative gain, the performance indicators 30
4.1.2 Estimation of the performance indicators 32
4.1.3 Equivalent computational budget between the different optimization algorithms 33
4.1.4 *Apriori* and *online* comparison 35
4.2 Modeling the motion of a lung tumor 35
4.3 Results for the *apriori* comparison 38
4.4 Results for the *online* comparison 41

5 Discussion **44**
5.1 Interpretation of the results 44
5.2 Tailoring the method to tumor tracking using medical data 45
5.3 Improving the algorithm and future work 46

6 Conclusion **48**

Appendices **54**

A Theoretical annex **55**
A.1 Calibrating the model's parameters 55
A.2 Statistical inference in data using Bootstrap 56

B Additional figures and table **58**
B.1 Additional figures 58
B.2 Additional table 63

C Matlab code **64**

List of Notations and Acronyms

t	discrete time
X_t, x_t	state variable at time t and its realization
Y_t, y_t	measurements at time t and its realization
Z_t, z_t	quantity of interest at time t and its realization
$\mathcal{M}$	set of measurements
N	number of measurements
T	size of the time horizon, $t = 0, \ldots, T$
τ	measurement time, $\tau \in \{0, \ldots, T\}$
$p_t(\cdot\|\cdot)$	transition density on X_t
$q_t(\cdot\|\cdot)$	transition density from X_t to Y_t
$h_t(\cdot)$	objective function
$\mathcal{F}$	initial distribution
$\hat{z}_t$	"best estimate" of z_t
$\left(X_t^i, \omega_t^i\right)$	state particle i at time t and its associated weight
Ω_t	total weight of the particles at time t
$\hat{z}_t(\mathcal{M})$	particle filter estimate of $\hat{z}_t$ using the measurement time set $\mathcal{M}$
$PF[\{y_t\}_{t\in\mathcal{M}}]$	particle filter using the measurement time set $\mathcal{M}$
n_{part}	number of particles used by the particle filter
O_{PF}	complexity of the particle filter algorithm
$\text{MSE}(\mathcal{M}), mse(\mathcal{M})$	mean squared tracking error using the measurement time set $\mathcal{M}$ and its realization
$\mathbb{E}_{\text{MSE}}[\mathcal{M}]$	expected mean squared tracking error using the measurement time set $\mathcal{M}$
n_{draws}	number of draws used by the expected mean squared tracking estimator
$\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}]$	estimator of the expected mean squared tracking error using the measurement time set $\mathcal{M}$
O_{GF}	complexity of the greedy forward algorithm
O_{GB}	complexity of the greedy backward algorithm

O_{SA}	complexity of the simulated annealing algorithm
O_{GA}	complexity of the genetic algorithm
$\mathcal{M}_{\mathrm{GF}}$	measurement time set returned by the greedy forward algorithm
$\mathcal{M}_{\mathrm{GB}}$	measurement time set returned by the greedy backward algorithm
$\mathcal{M}_{\mathrm{SA}}$	measurement time set returned by the simulated annealing algorithm
$\mathcal{M}_{\mathrm{GA}}$	measurement time set returned by the genetic algorithm
$\mathcal{M}_{\mathrm{RT}}$	measurement time set returned by the random trials algorithm
$\mathcal{M}(y)$	measurement time set returned by the *online* algorithm
$\mathcal{M}_{\mathrm{REG}}$	regularly spaced measurement time set
$G(\mathcal{M}), g(\mathcal{M})$	relative mean squared tracking error gain between $\mathcal{M}$ and $\mathcal{M}_{\mathrm{REG}}$
gain on $\mathbb{E}_{\mathrm{MSE}}$ of M	gain on the expected mean squared tracking error between $\mathcal{M}$ and $\mathcal{M}_{\mathrm{REG}}$
expected gain of $\mathcal{M}$	expected gain on the mean squared tracking error between $\mathcal{M}$ and $\mathcal{M}_{\mathrm{REG}}$
median gain of $\mathcal{M}$	median gain on the mean squared tracking error between $\mathcal{M}$ and $\mathcal{M}_{\mathrm{REG}}$
prob. of gain of $\mathcal{M}$	prob. of positive gain on the mean squared tracking error between $\mathcal{M}$ and $\mathcal{M}_{\mathrm{REG}}$
$(1-\alpha)$	confidence rate about the margins
n_{tests}	number of test used to estimate the performance indicators
$n_{\mathrm{bootstrap}}$	number of bootsrap set used on the n_{tests} tests to estimate the confidence bounds on the *median gain* and *prob. of gain*
tracking model	dynamic model used to run the particle filter
training model	dynamic model used to run the estimator $\hat{\mathbb{E}}_{\mathrm{MSE}}[\mathcal{M}]$
testing model	dynamic model used to estimate the performance indicators
model parameters	parameters T and N
optimization parameters	parameters of the genetic algorithm: n_{draws}, $n_{\mathrm{part.}}$, pop. size, and max. gen.
test parameters	parameters n_{tests}, $n_{\mathrm{part.\ tests}}$, $n_{\mathrm{bootstrap}}$, and α

List of Figures

2.1 Pseudo-code for the Monte Carlo algorithm 6
2.2 Pseudo-code of the Sequential Importance Sampling algorithm . . . 10
2.3 Pseudo-code of the Sequential Importance Sampling with Resampling algorithm 12

3.1 Pseudo-code describing the particle filter algorithm with intermittent measurements. 17
3.2 Representation of the expected tracking MSE estimator 20
3.3 Representation of the state-space exploration in a greedy forward approach. 24
3.4 Representation of the state-space exploration in a greedy backward approach. 25
3.5 The expected mean squared tracking error estimator interpreted as a function. 29

4.1 Representation of the generation of one draw $g(\mathcal{M})$. 33
4.2 Illustration of a realization of the lung tumor dynamics. 37
4.3 Evolution of the minimum cost $\hat{\mathbb{E}}_{\mathrm{MSE}}$ with respect to the number of cost function evaluations for the different optimization algorithms. . 39
4.4 Histogram of the relative gain $G(\mathcal{M}_{\mathrm{GA}})$ in the *apriori* comparison. . 40
4.5 Comparison over a single draw of true value z_t with the values $\hat{z}_t(\mathcal{M}_{\mathrm{GA}})$ and $\hat{z}_t(\mathcal{M}_{\mathrm{REF}})$. 41
4.6 Histogram of the relative gain $G(\mathcal{M}_{\mathrm{GA}}(y))$ in the *online* comparison. 43

5.1 Estimation of the expected mean squared tracking error using medical data as the training model. 46

B.1 Histogram of the relative gain $G(\mathcal{M}_{\mathrm{GF}})$ in the *apriori* comparison. . 58
B.2 Histogram of the relative gain $G(\mathcal{M}_{\mathrm{GB}})$ in the *apriori* comparison. . 59
B.3 Histogram of the relative gain $G(\mathcal{M}_{\mathrm{SA}})$ in the *apriori* comparison. . 60
B.4 Histogram of the relative gain $G(\mathcal{M}_{\mathrm{RT}})$ in the *apriori* comparison. . 61
B.5 Histogram of the relative gain $G(\mathcal{M}_{\mathrm{GA}})$ in the *online* comparison. . 62

List of Tables

4.1 Values of the model parameters, test parameters and optimization parameters for the *apriori* and *online* comparison. 35
4.2 Parameters used for the lung tumor model 38
4.3 Performance indicators of the measurement time sets returned by the GF, GB, SA, GA and RT optimization algorithms to solve the *apriori problem*. 40
4.4 Performance indicators for the solutions to the *apriori* and *online problem* returned bu the GA algorithm. 42

B.1 Table of the parameters value used for the optimization algorithms in the *apriori comparison.* 63

Chapter 1

Introduction

Particle filter algorithm has shown its ability to track the state of discrete dynamic systems from noisy measurements [1] and has been used for several real-world problems [2], among others, in computer vision [3, 4, 5], positioning and navigation [6, 7], chemistry [8, 9], mechanics [10], robotics [11], and medicine [12]. Under some circumstances, due to energy consumption, economical constraints or health hazards, the measurements have to be performed in a parsimonious manner. This is the case for mobile tumor tracking based on X-ray images, as the number of X-ray acquisitions has to be constrained to limit patients' exposure to undesirable irradiation [13]. Under such constraints, one has a measurement budget and wants to find the optimal moments to measure the system. The optimal moments are defined as the measurement times that minimize the expected value of the mean squared tracking error. Their computation can be performed either *apriori*, i.e. before any measurement acquisition, or *online*, i.e. as the measurements are acquired. These problems are referred to respectively as the *apriori problem* and the *online problem*.

The problem of state tracking under the constraint of sparse measurements have been studied in both linear [14, 15, 16, 17] and the nonlinear case [18, 19, 20, 21, 22] with the use of the particle filter framework. However, in these works, the missing measurements occur randomly. The choice of optimal measurement times, as a parameter to be optimized, has been studied in the particular case of linear systems subject to Gaussian noise processes, using the Kalman filtering framework, in both discrete [23] and continuous-time [24, 25] settings.

The more general formulations have received little attention in the literature. Antoine Aspeel, Raphaël Jungers, Benoît Macq, and I addressed the problem of providing optimal measurement times solving the *apriori problem* by proposing a method nesting a genetic algorithm, a Monte Carlo algorithm, and particle filter. These results are to be presented at the IEEE International Conference on Image Processing 2020 Conference Proceedings and IEEE Xplore ®.

Contribution. The main contributions in this work are (i) to propose an efficient algorithm to solve the problem of finding near-optimal measurement times for particle filtering over a finite time horizon, both *apriori* and *online*, and (ii) to show the added value of selecting measurement times for particle filtering.

Structure of the report. This report is organized as follows: chapter 2 introduces the reader to the theory behind the tracking of stochastic dynamic systems, chapter 3 presents the main contribution of this work, an algorithm that finds near-optimal measurement time sets, chapter 4 evaluates the method on a simplified lung tumor model, and finally chapter 5 analyzes the results obtained and discusses ways of tailoring the method to mobile tumor tracking, possible improvements and perspectives.

A MATLAB (MathWorks, Natick, Massachusetts, USA) implementation of all the presented algorithms and the code that generate all figures are available on GITHUB at
GITHUB.COM/AMAURYGOUVERNEUR/OPTIMAL_MEASUREMENT_TIMES_FOR_PARTICLE_FILTERING.

Chapter 2

Theoretical background

This chapter aims at providing the reader with a general understanding of stochastic dynamic systems and their tracking. Section 2.1 is a reminder about fundamental probability concepts that are at the core of the methods later developed. The Monte Carlo algorithm and its properties are presented in section 2.2. Section 2.3 gives a formal definition to the tracking problem and the "best estimate". Finally, section 2.4 presents the particle filter algorithm by developing the sequential Monte Carlo methods. The theory presented in this chapter is inspired from the excellent course "Computer Intensive Methods in Mathematical Statistics" given by professor Jimmy Olsson at KTH Royal Institute of Technology [26] and from the reference book of the course [27].

2.1 Some fundamental concepts of probability

Before explaining the Monte Carlo method and the particle filter algorithm, some fundamental concepts of probability need to be introduced, hence the following definitions.

Definition 2.1 (probability space)**.** A probability space contains:

- a sample space Ω, which is the set of possible outcomes $\omega \in \Omega$. A set $A\Omega$ of the sample space is called a event.
- a probability measure P assigning a value in $[0, 1]$ to each event such that:
  - (i) $P(\Omega) = 1$.
  - (ii) $P(\cup_{i\in I} A_i) = \sum_{i\in I} P(A_i)$ for any countable collection $(A_i)_{i\in I}$ of pairwise disjoint events.

Given a probability space (Ω, P) a random variable X and its distribution function can be defined.

Definition 2.2 (random variable)**.** A random variable X is a function $\Omega \to X$, where X is called the state space of X.

Definition 2.3 (distribution function)**.** Given a random variable X on some probability space (Ω, P), the function $x \mapsto F(x) = P(X \leq x)$ is called the distribution function of X.

Using the previously introduced distribution function, the probability density function, the expectation and the variance can be defined.

Definition 2.4 (probability density function)**.** In the case where $X \in \mathbb{R}^d$, a function $f : X \mapsto \mathbb{R}_+$ such that for all $x = (x_1, \ldots, x_d)$,

$$P(X \leq x) = \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_d} f(z)dz, \tag{2.1}$$

is called the density of X. Consequently, if F is differentiable in x, $f(x) = \partial_{x_1} \cdots \partial_{x_d} F(x)$ for all x.

Definition 2.5 (expectation)**.** The integral $\mathbb{E}[X] = \int_X xf(x)dx$ is called the expectation of X. The expectation of a random variable is also called the mean. The subscript $\mathbb{E}_f$ can be used to make explicit that the integral is over the probability density $f(x)$.

Definition 2.6 (variance)**.** The variance of a random variable X with mean μ is defined as:

$$\mathbb{V}[X] = \mathbb{E}[|X - \mu|^2]. \tag{2.2}$$

The probability measure, the probability density function or the expectation allows to define the notion of independence between two random variables.

Definition 2.7 (independence)**.** Two random variables X and Y are said to be independent if for all events, arguments and functions ψ and ϕ:

$$P(X \in A, Y \in B) = P(X \in A)P(Y \in B), \tag{2.3}$$

$$\iff f(x, y) = f(x)f(y), \tag{2.4}$$

$$\iff \mathbb{E}[\phi(X)\psi(Y)] = \mathbb{E}[\phi(X)]\mathbb{E}[\psi(Y)]. \tag{2.5}$$

Two fundamental results are at the core of the Monte Carlo method: the law of large numbers and the central limit theorem.

The law of large numbers can be stated as follows.

Theorem 2.1 (the law of large numbers)**.** Let $X^1, X^2, X^3, \dots$ be independent and identically distributed random variables with mean μ and set $S_N = \sum_{i=1}^{N} X_i$. Then

$$\lim_{N\to\infty} \frac{1}{N} S_N = \mu \quad \text{(with probability one)}. \tag{2.6}$$

The central limit theorem describes the error rate between S_N/N and μ for large N and provides the rate of convergence. First one needs to define the convergence in distribution.

Definition 2.8 (convergence in distribution)**.** Let X and $X^1, , X^2, X^3, \dots$ be random variables. Denote $F_N(x) = P(X_N \leq x)$ and $F(x) = P(X \leq x)$ the distribution functions of X_N and X, respectively. Let C_F be the set of continuity points of F. (X_N) is said to converge in distribution to X if for all $x \in C_F$,

$$\lim_{N\to\infty} F_N(x) = F(x) \quad \text{(with probability 1)}. \tag{2.7}$$

The convergence in distribution is noted: $X_N \xrightarrow{d} X$.

The central limit theorem describes the convergence in distribution of the series S_N/N.

Theorem 2.2. Let $X^1, , X^2, X^3, \dots$ be independent and identically distributed random variables with mean μ and variance σ^2 and set $S_N = \sum_{i=1}^{N} X_i$. Then

$$\sqrt{N}\Big(\frac{1}{N} S_N - \mu\Big) \xrightarrow{d} \mathcal{N}(0, \sigma^2), \tag{2.8}$$

where $\mathcal{N}(0, \sigma^2)$ is the normal distribution with zero mean and variance σ^2.

The probability of the random variable X can depend on another random variable Y. In that case X and Y are said to be conditionally dependent. The conditional distribution of X given Y is defined as follows.

Definition 2.9. Let X and Y be two random variables. The conditional distribution of Y given X is:

$$p(y|x) = \frac{p(x,y)}{\int_Y p(x,y)dy}. \tag{2.9}$$

It can also be referred to as the transition density from X to Y.

The Bayes' formula provides a way to compute the conditional distribution of X given Y from the conditional distribution of Y given X and their probability distributions.

Definition 2.10 (Bayes' formula)**.** Let X and Y be two random variables and p the probability density function. Then

$$p(x|y) = \frac{p(y|x)\int_Y p(x,y)dy}{\int_X p(x,y)dx}. \tag{2.10}$$

2.2 Overview of the Monte Carlo method

The Monte Carlo method addresses the problem of computing the expectation of a random variable:

$$\tau = \mathbb{E}[\psi(X)] = \int_X \psi(x) f(x) dx, \tag{2.11}$$

where X is a random variable taking values in $X \in \mathbb{R}^d$, $f : X \to \mathbb{R}_+$ is a probability density on X, and $\psi : X \to \mathbb{R}$ is some function, referred to as the objective function, such that the above expectation is finite.

Let $(X^i)_{i=1}^N$ be independent identically distributed random variables with density f and τ_N be defined as:

$$\tau_N = \frac{1}{N} \sum_{i=1}^{N} \psi(X^i). \tag{2.12}$$

Then by the law of large numbers, it comes that:

$$\lim_{N \to \infty} \tau_N \to \mathbb{E}[\psi(X)]. \tag{2.13}$$

This result is at the core of the basic Monte Carlo sampler algorithm. The pseudo-code can be found at figure 2.2.

Algorithm 1 Monte Carlo algorithm

for $i = 1 \to N$ **do**
| draw $X^i \sim f$
end
set $\tau_N \leftarrow \frac{1}{N} \sum_{i=1}^{N} \psi(X^i)$

Figure 2.1: Pseudo-code for the Monte Carlo algorithm

The error of the Monte Carlo method is random. However, it is unbiased since:

$$\tau_N = \mathbb{E}[\frac{1}{N} \sum_{i=1}^{N} \psi(X^i)] = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}[\psi(X^i)] = \tau. \tag{2.14}$$

Denoting $\sigma^2(\psi) = \mathbb{V}[\psi(X)]$, one can show that the variance of τ_N is:

$$\mathbb{V}[\tau_N] = \mathbb{E}[|\tau_N - \tau|^2] = \frac{1}{N^2} \sum_{i=1}^{N} \mathbb{V}[\psi(X^i)] = \frac{1}{N} \sigma^2(\psi), \tag{2.15}$$

implying that the convergence rate of the method is $O(\frac{1}{\sqrt{N}})$. In addition, the central limit theorem implies that

$$\sqrt{N}\Big(\tau_N - \tau\Big) \xrightarrow{d} \mathcal{N}(0, \sigma^2(\psi)). \tag{2.16}$$

This provides approximate confidence bounds to the estimator τ_N:

$$I_\alpha = \Big(\tau_N \pm \lambda_{\alpha/2}\frac{\sigma(\psi)}{\sqrt{N}}\Big), \tag{2.17}$$

where $\lambda_{\alpha/2}$ denotes the p-quantile of the standard normal distribution. One can say that I_α covers τ with (approximate) probability $1 - \alpha$.

2.3 Tracking problem

A tracking or filtering problem is the problem of computing the "best estimate" for the current state of a stochastic dynamic system from noisy observations.

A stochastic dynamic system describes the evolution of 3 random variables: X_t, Y_t and Z_t. The variable Z_t is the quantity one wants to track, to estimate. It is related to a state variable X_t that is observed via noisy measurements Y_t.

The formal definition of a stochastic dynamic system is given by:

Definition 2.11 (stochastic dynamic system)**.** Let $t = 0, \ldots, T$ be the finite time horizon studied. A stochastic dynamic system is defined as follows:

$$X_{t+1}|X_t \sim q_t(x_{t+1}|x_t) \quad \text{for} \quad t = 0, \ldots, T-1, \tag{2.18}$$
$$Y_t|X_t \sim p_t(y_t|x_t) \quad \text{for} \quad t = 0, \ldots, T, \tag{2.19}$$
$$Z_t = h_t(X_t) \quad \text{for} \quad t = 0, \ldots, T, \tag{2.20}$$
$$X_0 \sim \mathcal{F}, \tag{2.21}$$

where

- $X_t \in \mathbb{R}^n$, $Y_t \in \mathbb{R}^m$, $Z_t \in \mathbb{R}^p$ are random variables,
- $q_t(\cdot|\cdot)$: is the transition density on X_t,
- $p_t(\cdot|\cdot)$ is the transition density from X_t to Y_t,
- $h_t(\cdot)$ is the objective function,
- and $\mathcal{F}$ is the initial distribution of X_t.

Remark 2.1. The pair $(X_t, Y_t)_{t\geq 0}$ forms what is called a Hidden Markov model.

The "best estimate" of z_t given the observations $(y_0, \ldots, y_t)$, denoted $y_{0:t}$, is computed as follows.

Definition 2.12 ("best estimate")**.** The "best estimate" of z_t is defined as:

$$\hat{z}_t = \mathbb{E}[h_t(X_t)|y_{0:t}] = \int_{X_t} h_t(x_t) f_t(x_t|y_{0:t}) dx_t, \tag{2.22}$$

where f_t is the transition density from Y_t to X_t. Computing $\hat{z}_t$ is referred to as the filtering problem.

Remark 2.2. An analytic expression of this quantity is, however, only available for a relatively small and restrictive choice of systems and measurements model, the most important being the case of linear systems subject to Gaussian noise.

The transition density from Y to X is unknown and in most cases intractable (c.f. remark 2.2). However, the Bayes' formula provides a way to compute it up to an unknown constant:

$$f_t(x_t|y_{0:t}) = \frac{f_t(x_t, y_{0:t})}{p_t(y_{0:t})} \tag{2.23}$$

Indeed, for a given $y_{0:t}$, $p(y_{0:t})$ is a constant. It is referred to as the normalizing constant, c_t. With this notation, $p(x_t|y_{0:t})$ can be written as:

$$f_t(x_t|y_{0:t}) = \frac{\phi_t(x_t)}{c_t}. \tag{2.24}$$

where ϕ_t is a function of only x_t for a given $y_{0:t}$. It can be showed that

$$\phi_t(x_t) = F(x_0) p_0(y_0|x_0) \prod_{k=1}^{t} p_k(y_k|x_k) q_k(x_k|x_{k-1}). \tag{2.25}$$

2.4 Sequential Monte Carlo method

The goal of a Sequential Monte Carlo method is to produce a sequence of N weighted samples $\left(X_t^i, \omega_t(X_t^i)\right)_{t\geq 0}^{i=1,\ldots,N}$ representing the target distribution $f_t(x_t|y_{0:t})$.

Each draw $X_t^i = (X_0^i, X_1^i, \ldots, X_t^i)$ is called a particle and $\omega_t^i = \omega_t(X_t^i)$ is its associated weight. The total weight of the particles Ω_t is defined as $\Omega_t = \sum_{i=1}^N \omega_t^i$.

Assuming one had generated particles (X_t^i) according to the transition density $g_t(x_{t+1}|x_t)$, then, if one computes the weight of the particles according to $\omega_t(x_t) =$

$\frac{\phi_t(x_t)}{g_t(x_t)}$, one can approximate z_t using a Monte Carlo method on the weighted particles:

$$\begin{aligned}
\hat{z}_t = \mathbb{E}_f[h_t(X_t)|y_{0:t}] &= \int_{X_t} h_t(x_t) f_t(x_t|y_{0:t}) dx_t \\
&= \frac{\int h_t(x_t) \frac{\phi_t(x_t)}{c} dx_t}{\int \frac{\phi_t(x_t)}{c} dx_t} \quad (\text{as } \int \frac{\phi_t(x_t)}{c} dx_t = 1) \\
&= \frac{\int h_t(x_t) \frac{\phi_t(x_t)}{g_t(x_t)} g_t(x_t) dx_t}{\int \frac{\phi_t(x_t)}{g_t(x_t)} g_t(x_t) dx_t} \\
&= \frac{\int h_t(x_t) \omega_t(x_t) g_t(x_t) dx_t}{\int \omega_t(x_t) g_t(x_t) dx_t} \\
&= \frac{\mathbb{E}_g[h_t(X_t)\omega_t(X_t)]}{\mathbb{E}_g[\omega_t(X_t)]} \\
&\approx \frac{\frac{1}{N}\sum_{i=1}^N h_t(X_t^i)\omega_t^i}{\frac{1}{N}\sum_{i=1}^N \omega_t^i} \\
&= \sum_{i=1}^N \frac{\omega_t^i}{\Omega_t} h_t(X_t^i). \qquad (2.26)
\end{aligned}$$

2.4.1 Sequential Importance Sampling

Let $g_t(x_t)$ be an instrumental distribution specified through transition densities by

$$g_t(x_t) = g_0(x_0) \prod_{k=0}^{t-1} g_k(x_{k+1}|X_k). \qquad (2.27)$$

Assuming that $(X_t^i)_{i=1}^N$ have been generated from $g_t(x_t)$ such that

$$\sum_{i=1}^N \frac{\omega_t^i}{\Omega_t} h_t(X_t^i) \approx \hat{z}_t, \qquad (2.28)$$

as $g_{t+1}(X_{t+1}) = g_t(x_t) g_t(x_{t+1}|x_t)$, a draw X_{t+1}^i may be generated by simulating $X_{t+1}^i \sim g_t(x_{t+1}|X_t^i)$.

Consequently, $\left({X_{t+1}^i}^i, \omega_{t+1}^i\right)$ can be generated by computing $\omega_{t+1}^i \leftarrow \frac{\phi_{t+1}(X_{t+1}^i)}{g_{t+1}(X_{t+1}^i)}$.

The computation of ω_{t+1}^i can be done recursively as:

$$\begin{aligned}\omega_{t+1}^i &= \frac{\phi_{t+1}(X_{t+1}^i)}{g_{t+1}(X_{t+1}^i)} \\ &= \frac{\phi_{t+1}(X_{t+1}^i)}{\phi_t(X_t^i)g_t(X_{t+1}^i|X_t^i)}\frac{\phi_t(X_t^i)}{g_t(X_t)} \\ &= \frac{\phi_{t+1}(X_{t+1}^i)}{\phi_t(X_t^i)g_t(X_{t+1}^i|X_t^i)}\omega_t^i. \end{aligned} \tag{2.29}$$

The approximation of $\hat{z}_t$ by the sequential importance sampling is denoted $\hat{z}_t^{\mathrm{SIS,N}}$:

$$\hat{z}_t^{\mathrm{SIS,N}} := \sum_{i=1}^{N} \frac{\omega_t^i}{\Omega_t} h_t(X_t^i) \approx \mathbb{E}_f[h_t(X_t)|y_{0:t}]. \tag{2.30}$$

The pseudo-code describing the Sequential Importance Sampling is shown in figure 2.2.

Algorithm 2 Sequential Importance Sampling algorithm

Data: $\{y_t\}_{t=0,\dots,T}$
Result: $\{\hat{z}_t^{SIS,N}\}_{t=0,1,\dots,T}$
for $i = 1 \to N$ **do**
  draw $X_0^i \sim g_0(x_0)$
  set $\omega_0^i \leftarrow \frac{\phi_0(X_0^i)}{g_0(X_0^i)}$
  set $\hat{z}_0^{\mathrm{SIS,N}} \leftarrow \sum_{i=1}^{N} \frac{\omega_i^0}{N} h_0(X_0^i)$
end
for $t = 0, \dots, T-1$ **do**
  for $i = 1 \to N$ **do**
    draw $X_{t+1}^i \sim g_t(x_{t+1}|X_t^i)$
    set $\omega_{t+1}^i \leftarrow \frac{\phi_{t+1}(X_{t+1}^i)}{\phi_t(X_t^i)g_t(X_{t+1}^i|X_t^i)}\omega_t^i$
    set $\hat{z}_{t+1}^{\mathrm{SIS,N}} \leftarrow \sum_{i=1}^{N} \frac{\omega_i^{t+1}}{N} h_{t+1}(X_{t+1}^i)$
  end
end

Figure 2.2: Pseudo-code of the Sequential Importance Sampling algorithm

Unfortunately, due to the fact that the particle weights are generated through

subsequent multiplications, one or few particles will end carrying almost all the weight. This problem is called weight degeneration.

2.4.2 Sequential Importance Sampling with Resampling

A simple idea allows to avoid the weight degeneracy problem. Having at hand a weighted sample $(X_t^i, \omega_t^i)_{i=1}^N$ approximating a probability distribution function f_t, a uniformly weighted sample can be formed by resampling with replacement new variables $(\tilde{X}_t^i)_{i=1}^N$ form $(X_t^i)_{i=1}^N$ according to the weights $(\omega_t^i)_{i=1}^N$.

The Sequential Importance Sampling with Resampling (SISR) estimate of $\hat{z}_t$ is denoted $\hat{z}_t^{\mathrm{SISR},N}$. It is defined by:

$$\hat{z}_t^{\mathrm{SISR},N} \coloneqq \sum_{i=1}^{N} \frac{\omega_t^i}{\Omega_t} h_t(X_t^i). \tag{2.31}$$

One can prove that the resampling does not induce bias to the estimator.

The full scheme for SISR is given by figure 2.3.

Algorithm 3 Sequential Importance Sampling algorithm with Resampling

Data: $\{y_t\}_{t=0,\dots,T}$
Result: $\{\hat{z}_t^{SISR,N}\}_{t=0,1,\dots,T}$
for $i = 1 \to N$ **do**
  draw $X_0^i \sim g_0(x_0)$
  set $\omega_0^i \leftarrow \frac{\phi_0(X_0^i)}{g_0(X_0^i)}$
  set $\hat{z}_0^{\text{SISR,N}} \leftarrow \sum_{i=1}^N \frac{\omega_i^0}{N} h_0(X_0^i)$
end
for $t = 0, \dots, T-1$ **do**
  for $i = 1 \to N$ **do**
    draw with replacement $(\tilde{X}_t^i)_{i=1}^N$ among $(X_t^i)_{i=1}^N$ according to the probabilty $(\omega_t^i/\Omega_t)_{i=1}^N$
    draw $X_{t+1}^i \sim g_t(x_{t+1}|(\tilde{X}_t^i)$
    set $\omega_{t+1}^i \leftarrow \frac{\phi_{t+1}(X_{t+1}^i)}{\phi_t(X_t^i) g_t(X_{t+1}^i|X_t^i)} \omega_t^i$
    set $\hat{z}_{t+1}^{\text{SISR,N}} \leftarrow \sum_{i=1}^N \frac{\omega_i^{t+1}}{N} h_{t+1}(X_{t+1}^i)$
  end
end

Figure 2.3: Pseudo-code of the Sequential Importance Sampling with Resampling algorithm

The convergence of the algorithm as the number of particles N tends to infinity for t fixed has been established [28]. Using this result, one can write a central limit theorem for the SISR estimate with N particles $\hat{z}_t^{\text{SISR},N}$:

$$\sqrt{N}\Big(\hat{z}_t^{\text{SISR},N} - \hat{z}_t\Big) \xrightarrow{d} \mathcal{N}(0, \sigma^2(\phi)). \tag{2.32}$$

2.4.3 The particle filter

The particle filter is a possible implementation of the Sequential Importance Sampling with Resampling where the particles are evolving according to the dynamic system, that is $g_t(x_{t+1}|x_t)$ is set to $q(x_{t+1}|x_t)$. This leads to a simplified equation to compute the particles weight.

Recall from equation (2.25) that

$$\phi_t(x_t) = F(x_0)p(y_0|x_0)\prod_{k=1}^{t} p(y_k|x_k)q(x_k|x_{k-1}).$$

Then it comes that

$$\frac{\phi_{t+1}(X_{t+1})}{\phi_t(x_t)} = p(y_{t+1}|x_{t+1})q(x_{t+1}|x_t). \tag{2.33}$$

Plugging this last result and $g_t(x_{t+1}|x_t) = q(x_{t+1}|x_t)$ in equation (2.29), it comes that:

$$\begin{aligned}\omega_{t+1} &= \frac{\phi_{t+1}(X_{t+1})}{\phi_t(x_t)g_t(x_{t+1}|x_t)} \\ &= \frac{p(y_{t+1}|x_{t+1})q(x_{t+1}|x_t)}{q(x_{t+1}|x_t)} = p(y_{t+1}|x_{t+1}).\end{aligned} \tag{2.34}$$

The weight of the particle X_t^i is simply the conditional probability of the measurement given the particle.

Chapter 3

Materials and methods

This chapter is organized into six sections. In section 3.1, a stochastic dynamic system with intermittent measurements is introduced and the filtering problem on this system is defined. In section 3.2, the particle filter algorithm with sparse measurements is presented. It is explained how the particle filter can deal with intermittent measurements and provides an approximate solution to the filtering problem. In section 3.3, the definition of the optimal measurement time set is given, that is the set of measurements that minimizes the expected mean squared tracking error of the filtering estimate. This quantity is intractable and an estimator of the expected mean squared tracking error is presented in section 3.4. The *apriori problem* is defined as a combinatorial optimisation problem. A natural extension to this problem is presented in section 3.5, the *online problem*. The idea is to recompute the optimal measurement time set once a measurement has been acquired. In section 3.6, five different algorithms to solve the *apriori* and *online problems* are presented.

3.1 Filtering problem with intermittent measurements

A filtering or tracking problem with intermittent measurements is the problem of computing the "best estimate" for the state of a stochastic dynamic system from sparse noisy observations. A formal definition is the following.

Definition 3.1 (dynamic system with intermittent measurements)**.** Let $t = 0, \dots, T$ be the finite time horizon studied and $\mathcal{M}\{0, \dots, T\}$ be the set of available measurement times of size N, i.e. $|\mathcal{M}| = N$ with $\mathcal{M} = \{\tau_1, \dots, \tau_N\}$ such that $\tau_i \in \{0, \dots, T\}, \quad \tau_i < \tau_j, \quad i < j$.

A dynamic system with intermittent measurements is defined as:

$$X_{t+1}|X_t \sim q_t(x_{t+1}|x_t) \quad \text{for} \quad t = 0, \dots, T-1, \tag{3.1}$$
$$Y_t|X_t \sim p_t(y_t|x_t) \quad \text{for} \quad t \in \mathcal{M}, \tag{3.2}$$
$$Z_t = h_t(X_t) \quad \text{for} \quad t = 0, \dots, T, \tag{3.3}$$
$$X_0 \sim \mathcal{F}, \tag{3.4}$$

where

- $X_t \in \mathbb{R}^n$, $Y_t \in \mathbb{R}^m$, $Z_t \in \mathbb{R}^p$ are random variables,
- $q_t(\cdot|\cdot)$ is the transition density on X_t,
- $p_t(\cdot|\cdot)$ is the transition density from X_t to Y_t,
- $h_t(\cdot)$ is the objective function,
- and $\mathcal{F}$ is the initial distribution of X_t.

Example 3.1. In a tumor tracking problem based on X-ray images, $x_t \in \mathbb{R}^6$ can be a state vector containing the tumor's position and velocity in the 3-dimensional space, $y_t \in \mathbb{R}^2$ can be the 2-dimensional projection of the target and $z_t \in \mathbb{R}^3$ the position of the mass center in the 3-dimensional space.

The "best estimate" of z_t given the measurements up to time t, $\{y_\tau\}_{\tau \in \mathcal{M}, \tau \leq t}$ is defined similarly as in definition 2.12.

Definition 3.2 ("best estimate" with intermittent measurements)**.** The "best estimate" of z_t given the measurements $\{y_\tau\}_{\tau \in \mathcal{M}, \tau \leq t}$, is defined as:

$$\hat{z}_t = \mathbb{E}[h_t(X_t)|Y_\tau, \tau \in \mathcal{M}, \tau \leq t] = \int_{X_t} h_t(x_t) f_t(x_t|y_\tau, \tau \in \mathcal{M}, \tau \leq t) dx_t, \tag{3.5}$$

where $f_t(x_t|y_\tau, \tau \in \mathcal{M}, \tau \leq t)$ is the transition density from Y_t to X_t.

The same commentary as in remark 2.2 applies: an analytic expression of this expression is only available in particular choices of systems and measurements model.

3.2 Particle filter tracking with intermittent measurements

As explained in section 2.4, the particle filter algorithm proposes an efficient method to compute an approximation of the "best estimate" $\hat{z}_t$.

The idea is to produce $n_{\text{part.}}$ weighted particles to represent the target distribution $f_t(x_t|y_\tau, \tau \in \mathcal{M}, \tau \leq t)$. The particles are generated according to the dynamic of the system. The dynamic model used by the particle filter will be referred to as the *tracking model*.

To deal with intermittent measurements, the update of the weight of the particles is skipped when no measurement is available, i.e. when $t \notin \mathcal{M}$, all weights are set to 1. Indeed, no measurement at time t is equivalent to a measurement at time t with infinite noise, thus leading to equal probability for each particle.

For a given *tracking model* and constants T, $n_{\text{part.}}$, the particle filter can be interpreted as a function $PF[\cdot]$ of $\{y_t\}_{t\in\mathcal{M}}$. The approximation of $\hat{z}_t$ returned by the particle filter is noted $\hat{z}_t(\mathcal{M})$:

$$\{y_t\}_{t\in\mathcal{M}} \mapsto PF[\{y_t\}_{t\in\mathcal{M}}] =: \{\hat{z}_t(\mathcal{M})\}_{t=0,\dots,T}. \tag{3.6}$$

The full pseudo-code of the particle filter with intermittent measurements is shown in figure 3.1.

Algorithm 4 Particle filter with intermittent measurements

Data: $\{y_t\}_{t\in\mathcal{M}}$, *tracking model*$\Big(p_t(\cdot|\cdot), q_t(\cdot|\cdot), h_t(\cdot), \mathcal{F}\Big)$
Result: $\{\hat{z}_t(\mathcal{M})\}_{t=0,1,\ldots,T}$
initialization:
for $i = 1 \to n_{part.}$ **do**
  draw $X_0^i \sim \mathcal{F}$
end
selection:
for $i = 1 \to n_{part.}$ **do**
  if $0 \in \mathcal{M}$ **then**
    set $\omega_0^i \leftarrow p_0(y_0|X_0^i)$
  else
    set $\omega_0^i \leftarrow 1$
  end
end
set $\hat{z}_0(\mathcal{M}) \leftarrow \frac{\sum_{i=1}^{n_{\text{part.}}} h_0(X_0^i)\omega_0^i}{\Omega_0}$
for $t = 0, 1, \ldots, T-1$ **do**
  mutation:
  for $i = 1 \to n_{part.}$ **do**
    draw with replacement $(\tilde{X}_t^i)_{i=1}^{n_{\text{part.}}}$ among $(X_t^i)_{i=1}^{n_{\text{part.}}}$ according to the probabilty $(\omega_t^i/\Omega_t)_{i=1}^{n_{\text{part.}}}$
    draw $X_{t+1}^i \sim q_t(x_{t+1}|\tilde{X}_t^i)$
  end
  selection:
  for $i = 1 \to n_{part.}$ **do**
    if $t \in \mathcal{M}$ **then**
      set $\omega_{t+1}^i \leftarrow p_{t+1}(y_{t+1}|X_{t+1}^i)$
    else
      set $\omega_{t+1}^i \leftarrow 1$
    end
  end
  set $\hat{z}_{t+1}(\mathcal{M}) \leftarrow \frac{\sum_{i=1}^{n_{\text{part.}}} h_{t+1}(X_{t+1}^i)\omega_{t+1}^i}{\Omega_{t+1}}$
end

Figure 3.1: Pseudo-code describing the particle filter algorithm with intermittent measurements. Essentially, a particle filter algorithm alternates between a *mutation step* used to estimate the state at the next time step from the estimate at the current step; and a *selection step* that updates the state estimation to incorporate the information acquired in the last measurement. To deal with intermittent measurements, the update of the weight of the particles is skipped when no measurement is available, i.e. when $t \notin \mathcal{M}$, all weights are set to 1. The above pseudo-code is an adaptation to intermittent measurements of the sampling importance resampling particle filter (see Algorithm 4 in [1]).

3.3 An optimality criteria for measurement time sets

As explained in section 3.2, the particle filter estimate $\hat{z}_t(\mathcal{M})$ depends on the measurement time set $\mathcal{M}$.

The error on the particle filter estimate is random (see section 2.4), thus the mean squared tracking error of the set of measurement $\mathcal{M}$ is a random variable.

Definition 3.3 (mean squared tracking error)**.** For a given measurement time set $\mathcal{M}$, one defines the random variable

$$\mathrm{MSE}(\mathcal{M}) \coloneqq \frac{1}{T+1}\sum_{t=0}^{T} \|Z_t - \hat{Z}_t(\mathcal{M})\|^2. \tag{3.7}$$

The problem to address is to find the set of N measurement times that minimizes the expected filtering mean squared tracking error over the finite time horizon $t = 0, \ldots, T$. This set is referred to as the optimal measurement time set.

Definition 3.4 (*optimal measurement time set*)**.** The optimal measurement time set is the set $\mathcal{M}$ that minimizes the expected mean squared tracking error:

$$\begin{aligned} \min_{\mathcal{M}\{0,\ldots,T\}} \mathbb{E}\left[\mathrm{MSE}(\mathcal{M})\right] &=: \mathbb{E}_{\mathrm{MSE}}[\mathcal{M}] \\ \text{subject to } |\mathcal{M}| &= N \end{aligned} \tag{3.8}$$

where $\hat{z}_t(\mathcal{M})$ is obtained from the particle filter and $\|\cdot\|$ is the Euclidean norm. As $\mathrm{MSE}(\mathcal{M})$ depends on the random variables $\{Z_t\}_{t=0,1,\ldots,T}$ and $\{\hat{Z}_t(\mathcal{M})\}_{t=0,1,\ldots,T}$, the expectation is on these random variables.

Remark 3.1. It has to be noted that $||\cdot||$ could be any other norm.

3.4 The expected mean squared tracking error estimator and the *apriori* problem

To find the optimal measurement time set (3.8), one has to compute the expectation, $\mathbb{E}_{\mathrm{MSE}}[\mathcal{M}]$ which has, in general, no analytical expression.

The Monte Carlo algorithm (see section 2.2) provides a way to approximate this expectation.

The main idea is to generate n_{draws} realizations for $\{z_t\}_{t=0,1,\dots,T}$ and $\{\hat{z}_t(\mathcal{M})\}_{t=0,1,\dots,T}$ according to their probability distribution functions, and average their mean squared tracking error. The dynamic model used by the Monte Carlo to generate $\{z_t\}_{t=0,1,\dots,T}$ and $\{\hat{z}_t(\mathcal{M})\}_{t=0,1,\dots,T}$ is referred to as the *training model*.

The probability density function of $\{\hat{z}_t(\mathcal{M})\}_{t=0,1,\dots,T}$ is not given analytically but it is still possible to draw values according to it.
Indeed, for a given measurement time set $\mathcal{M}$, one can generate $\{x_t\}_{t=0,\dots,T}$, $\{y_t\}_{t\in\mathcal{M}}$ and $\{z_t\}_{t=0,\dots,T}$.
For each simulated sequence of measurements $\{y_t\}_{t\in\mathcal{M}}$, the particle filter computes $\{\hat{z}_t(\mathcal{M})\}_{t=0,\dots,T}$ and thus one can compute the mean squared tracking error of the draw $\{z_t, \hat{z}_t(\mathcal{M})\}_{t=0,\dots,T}$:

$$\text{mse}(\mathcal{M}) = \frac{1}{T+1}\sum_{t=0}^{T} \|z_t - \hat{z}_t(\mathcal{M})\|^2. \tag{3.9}$$

The nesting of the Monte Carlo algorithm and the particle filter is represented in figure 3.2. The complexity of the Monte Carlo estimation is

$$O(\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}]) = O\Big(n_{\text{draws}} \cdot O_{PF}\Big), \tag{3.10}$$

where O_{PF} is the particle filter's complexity.

The Monte Carlo estimator of the expectation $\mathbb{E}_{\text{MSE}}[\mathcal{M}]$, denoted by $\hat{\mathbb{E}}_{\text{MSE}}[\cdot]$, is defined as follows.

Definition 3.5 (Monte Carlo estimator of the expected mean squared tracking error)**.**

$$\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}] \coloneqq \frac{1}{n_{\text{draws}}}\sum_{k=1}^{n_{\text{draws}}} \text{mse}^k(\mathcal{M}) \approx \mathbb{E}_{\text{MSE}}[\mathcal{M}], \tag{3.11}$$

where $\text{mse}^k(\mathcal{M})$ are drawn according figure 3.2 and the dynamic model is subject to equations 3.4 to 3.1.

With this notation, the problem of finding the optimal measurement time set is approximately equivalent to finding the set $\mathcal{M}$ that minimizes $\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}]$. Solving this problem before any measurement acquisition is referred to as the *apriori problem*:

Definition 3.6 (*apriori problem*)**.** The *apriori problem* is define as finding the set $\mathcal{M}$ that minimizes the approximate expected mean squared tracking error before any measurement acquisition:

$$\min_{\mathcal{M}\{0,\dots,T\}} \hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}] \text{ subject to } |\mathcal{M}| = N, \tag{3.12}$$

where the dynamic model is subject to equations 3.4 to 3.1.

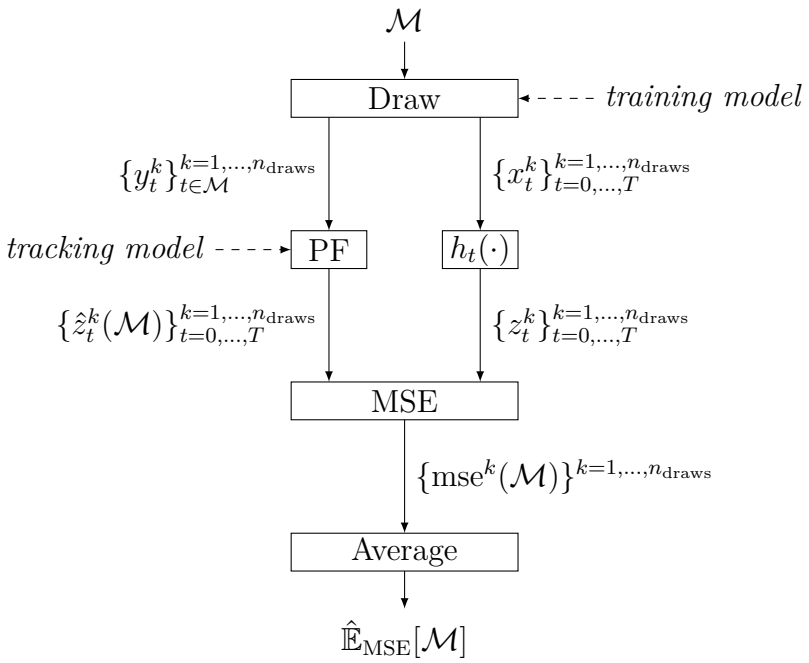

Figure 3.2: Representation of the expected mean squared tracking error estimator $\mathcal{M} \mapsto \hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}]$. The outputs generated by the "Draw" block are drawn according to the *training model*. The "$h_t(\cdot)$" block uses equation (3.3) to compute z_t^k that corresponds to the x_t^k. The "PF" block represents a particle filter that computes an estimate $\hat{z}_t^k(\mathcal{M})$ of z_t^k from previous intermittent measurements $\{y_t^k\}_{t\in\mathcal{M}}$. The model used by the particle filter is referred to as the *tracking model*. The "MSE" block computes the mean squared of the difference between the inputs. The "Average" block computes $\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}]$ according to equation (3.11).

3.5 Adaptation to *online* measurements acquisition

A natural extension of the *apriori problem* (definition 3.6) is to perform the computation of the optimal measurement time set $\mathcal{M} = \{\tau_1, \ldots, \tau_N\}$ in an *online* manner. The idea is to recompute the optimal measurement time(s) after measurements have been acquired. The full problem is to solve the *apriori problem* N times over smaller and smaller time horizons. The first *apriori problem* is solved over a time horizon 0 to T, the second will be over a time horizon 0 to $T - \tau_1$ and so on. This problem is referred to as the *online problem*.

After measurements $\{\tau_1, \ldots, \tau_j\}$ have been acquired, the dynamic system mod-

eled by 3.1 to 3.4 becomes:

$$\begin{aligned} X_{t+1}|X_t &\sim q_t(x_{t+1}|x_t)dx_{t+1} && \text{for } t = \tau_j, \dots, T-1, && (3.13)\\ Y_t|X_t &\sim p_t(y_t|x_t)dy_t && \text{for } t \in \mathcal{M}, && (3.14)\\ Z_t &= h_t(X_t) && \text{for } t = \tau_j, \dots, T, && (3.15)\\ X_{\tau_j} &\sim f_t(x_{\tau_j}|y_{\tau_1}, \dots, y_{\tau_j}), && && (3.16) \end{aligned}$$

where $f_0 = \mathcal{F}$.

After acquiring $\{\tau_1, \dots, \tau_j\}$, the *apriori problem* is to find a set M such that:

$$\min_{\mathcal{M}\{\tau_j+1,\dots,T\}} \hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}] \text{ subject to } |\mathcal{M}| = N - j, \tag{3.17}$$

where the dynamic model is subject to equations 3.13) to (3.16).

A notable difference between equations (3.4) and (3.16) is that the initial distribution $f_t(x_\tau|y_{\tau 1}, \dots, y_{\tau j})$ is not known anymore. Being able to draw according to this distribution is a requirement to run the particle filter on this model. As explained in section 2.4, the particle filter is an efficient method to approximate a probability density function and, in the same way as described in section 2.4, it can be used to approach the target distribution. Referring at figure 3.1, one can use the particle filter algorithm on the measurements $\{y_t\}_{t\in\{\tau_1,\dots,\tau_j\}}$ to generate the resampled particles $(\tilde{X}^i_{\tau j})_{i=1}^{n_{\text{part.}}}$ representing the target $f_t(x_{\tau j}|y_{\tau 1}, \dots, y_{\tau j})$. These particles can be used as initial draws to run the particle filter on the next measurement times, $t \geq \tau_j$.

The *online problem* can be defined as the following problem.

Definition 3.7 (*online problem*)**.** Let $\tau_0 = -1$. The *online problem* is defined as solving the following problem iteratively for $j = 0, \dots, N$:

- Find a solution $\mathcal{M} = \{\tau_{j+1}, \dots, \tau_N\}$ to

$$\min_{\mathcal{M}\{\tau_j+1,\dots,T\}} \hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}] \text{ subject to } |\mathcal{M}| = N-j \text{ and equations (3.13) to (3.16)}, \tag{3.18}$$

- Acquire measurement $y_{\tau_{j+1}}$.

Defining $\{y_t\}_{t=0,\dots,T}$ as the set of accessible measurements, one can interpret the *online* computation of optimal measurement as a function of $\{y_t\}_{t=0,\dots,T}$. The set of *online* acquired measurement is denoted $\mathcal{M}(y)$ as it is a function of the

measurements.

Let $O_{apriori}(T)$ be the complexity of the *apriori problem* over a time horizon T and O_{online} be the complexity of the *online problem*. As $O_{apriori}(T)$ is a decreasing function, it comes that:

$$\begin{aligned} O_{online} =& O_{apriori}(T) + \sum_{i=1}^{N-1} O_{apriori}(T - \tau_i) \\ \leq& N \cdot O_{apriori}(T). \end{aligned} \tag{3.19}$$

3.6 Five combinatorial optimization algorithms to solve the *apriori* problem

Solving the *apriori* problem (definition 3.6) or the *online problem* (definition 3.7) requires to solve a combinatorial optimization problem. It corresponds to finding the set $\mathcal{M}\{0, \ldots, T\}$ of cardinality N that minimizes $\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}]$.
An exhaustive search would require to test all admissible $\mathcal{M}$ which represents $\frac{(T+1)!}{(T+1-N)!N!}$ possibilities. It is computationally intractable for large N and T.

In this section, five heuristic optimization algorithms to find efficiently approximate solution to the *apriori problem* are presented, two direct methods (Greedy Forward and Greedy Backward) and three iterative methods (Simulated Annealing, Genetic Algorithm and Random Trial). In the following a feasible solution $\mathcal{M}_i$ of the optimization problem is referred to as an individual and a set of several individuals is called a population. The performance of these algorithms to solve the *apriori* and *online* problem will be compared in chapter 4.

3.6.1 The Greedy Forward algorithm

The greedy forward algorithm begins with an empty measurement time set. Sequentially, it adds the measurement times one at a time such that, at each iteration, the added measurement time minimizes the cost. It stops when the set contains N measurement times. [29]

The algorithm can be summarized as follows:

1. **Initialization**: The elite individual $\mathcal{M}^*$ is initialized with the empty set: $\mathcal{M}^* = \emptyset$.

2. **Forward exploration**: The population with all the neighboring bigger measurements set is created: $\mathcal{M}_i = \mathcal{M}^* \cup \{i\}, \quad \forall i \in \{0, \dots, T\} \setminus \mathcal{M}^*$.

3. **Evaluation**: For each individual in the population, the cost $\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}_i]$ is evaluated according to the objective function of the optimization problem.

4. **Selection**: The individual $\mathcal{M}_j$ with the lowest cost $\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}_j] \leq \hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}_i]$ becomes the new elite individual: $\mathcal{M}^* = \mathcal{M}_j$.

5. **Repeat**: Come back to step 2 until the measurement budget is reached: $|\mathcal{M}^*| = N$. The individual $\mathcal{M}^*$ is returned. The returned measurement time set is denoted $\mathcal{M}_{\text{GF}}$.

Selecting the i^{th} measurement time requires $T + 2 - i$ cost function evaluations. Consequently, this algorithm stops after n. calls GF cost function evaluations where n. calls GF is

$$\text{n. calls GF} \coloneqq \sum_{i=0}^{N-1} (T + 1 - i) = N(T + 2) - N(N + 1)/2. \tag{3.20}$$

The complexity of the implemented GF optimization is

$$O_{\text{GF}} \coloneqq O\Big((\text{n. calls GF}) \cdot O(\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}])\Big). \tag{3.21}$$

The forward exploration idea is illustrated on figure 3.3.

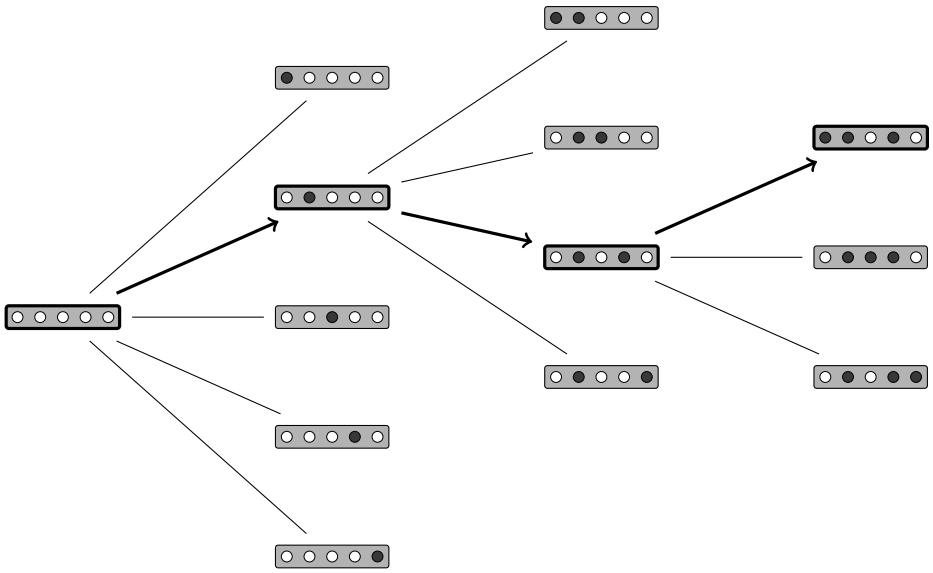

Figure 3.3: Representation of the state-space exploration in a greedy forward approach. The greedy forward algorithm starts from the empty set, evaluates the performance of the neighboring bigger states, selects the one with the best performance and iterates.

3.6.2 The Greedy Backward algorithm

The greedy backward algorithm works in the opposite direction than the greedy forward algorithm. It begins with a complete measurement time set. Sequentially, it takes out the measurement times one at a time such that, at each iteration, the dropped measurement time minimizes the cost. It stops when the set contains N measurement times [29].

The algorithm can be summarized as follows:

1. **Initialization**: The elite individual $\mathcal{M}^*$ is initialized with the full set:$\mathcal{M}^* = \{0, \ldots, T\}$.

2. **Backward exploration**: The population with all the neighboring smaller measurements set is created: $\mathcal{M}_i = \mathcal{M}^* \setminus \{i\}, \quad \forall i \in \mathcal{M}^*$.

3. **Evaluation**: For each individual in the population, the cost $\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}_i]$ is evaluated according to the objective function of the optimization problem.

4. **Selection**: The individual $\mathcal{M}_j$ with the lowest cost $\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}_j] \leq \hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}_i]$ becomes the new elite individual: $\mathcal{M}^* = \mathcal{M}_j$.

5. **Repeat**: Come back to step 2 until the measurement budget is reached: $|\mathcal{M}^*| = N$. The individual $\mathcal{M}^*$ is returned. The returned measurement time set is denoted $\mathcal{M}_{\text{GB}}$.

Removing the i^{th} measurement time requires $T + 2 - i$ cost function evaluation. Consequently, this algorithm stops after n. calls GB cost function evaluations where n. calls GB is

$$\text{n. calls GB} \coloneqq \sum_{i=N+1}^{T+1} (T + 1 - i) = \frac{(T+1)(T+2) - N(N+1)}{2}. \tag{3.22}$$

The complexity of the implemented GF optimization is

$$O_{\text{GB}} \coloneqq O\Big((\text{n. calls GB}) \cdot O(\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}])\Big). \tag{3.23}$$

The backward exploration idea is illustrated on figure 3.4.

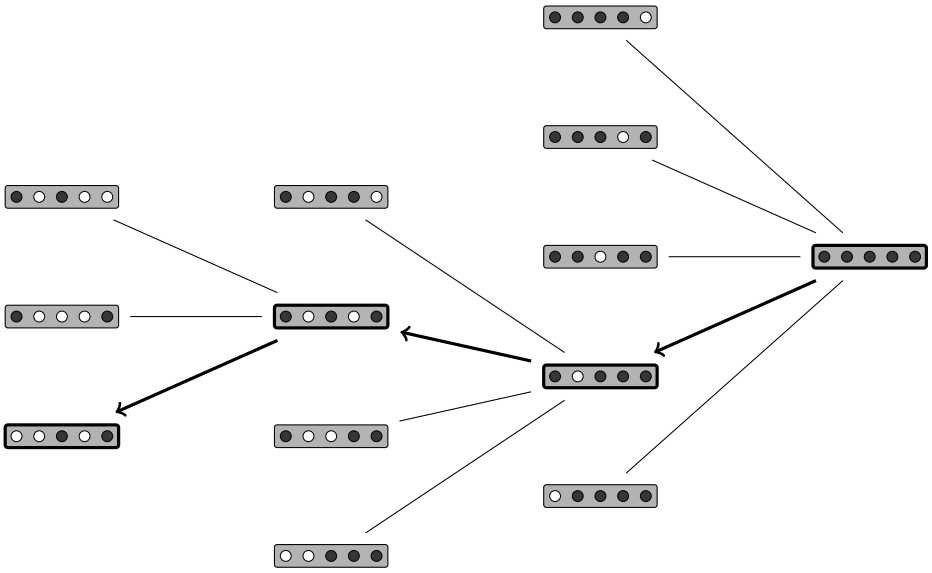

Figure 3.4: Representation of the state-space exploration in a greedy backward approach. The greedy backward algorithm starts from the full set, evaluates the performance of the neighboring smaller states, selects the one with the best performance and iterates.

3.6.3 The Simulated Annealing algorithm

Simulated annealing is one of the most commonly used algorithms to solve the problem of "black-box optimization" [30]. It works by extending a local search with

an efficient Metropolis acceptance criterion. The name simulated annealing comes from the analogy of materials physical annealing. This process brings a solid to high temperature, where its particles are free to move randomly and explore different states, before cooling it down slowly enough to reach a solid-state of minimum energy. The same idea is used with simulated annealing where the state-space points represent the possible states of the solid and the function to be minimized represents the energy of the solid.

The algorithm can be summarized as follows:

1. **Initialization**: An initial population $\mathcal{M}_i$ is uniformly sampled on the admissible set of problem. An initial temperature T° is settled.

2. **Mutation:** A mutated population individual of individuals $\tilde{\mathcal{M}}_i$ is created from a copy of the original population. For each individual $\tilde{\mathcal{M}}_i$, the measurement times are replaced by random ones according to a given probability. These random draws are done avoiding duplication, i.e., individual with repeated measurement times.

3. **Evaluation:** For each in the original population, the cost $\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}_i]$ is evaluated according to the objective function of the problem. Similarly, the cost is evaluated for the individuals of the mutated population, it gives $\hat{\mathbb{E}}_{\text{MSE}}[\tilde{\mathcal{M}}_i]$.

4. **Selection:** If $\hat{\mathbb{E}}_{\text{MSE}}[\tilde{\mathcal{M}}_i] < \hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}_i]$, then the mutated individual $\mathcal{M}_i^*$ becomes the new $\mathcal{M}_i$. Otherwise, $\mathcal{M}_i^*$ becomes the new $\mathcal{M}_i$ with probability $\exp((\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}_i] - \hat{\mathbb{E}}_{\text{MSE}}[\tilde{\mathcal{M}}_i])/T^\circ)$.

5. **Repeat**: Reduce the temperature as $T^\circ := K \cdot T^\circ$. Come back to step 2 until a stopping criterion is satisfied. The returned measurement time set is denoted $\mathcal{M}_{\text{SA}}$.

One pass of steps 2 to 5 is called an iteration. The initial temperature T° is set to 10 and the factor K is set to 0.8. With this value, the initial acceptance rate is close to 1. The stopping criterion used is on the number of generations limited to (max. iter.).

The complexity of the implemented SA optimization is

$$O_{\text{SA}} \coloneqq O\Big(2 \cdot (\text{max. iter.}) \cdot (\text{pop. size SA}) \cdot O(\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}])\Big), \tag{3.24}$$

where (pop. size SA) is the size of the population.

3.6.4 The Genetic algorithm

In the GA nomenclature, the measurement times of an individual $t \in \mathcal{M}_i$ are called its genes.

The GA [31] implements the following steps.

1. **Initialization:** An initial population of n_{indiv} individuals is uniformly sampled on the admissible set.
2. **Evaluation:** For each individual in the population, the cost $\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}_i]$ is evaluated according to the objective function of the optimization problem.
3. **Selection:** Individuals with low costs are preferably selected according to stochastic universal sampling [31].
4. **Crossover:** Selected individuals are mixed to produce new individuals called the offspring. This reconstitutes a complete population. The Count preserving crossover operator [32, 33] is used to ensure that the number of measurements stays N. Pairs of individuals are randomly formed. For each pair $(\mathcal{M}_1, \mathcal{M}_2)$, genes are picked randomly in the sets $\mathcal{M}_1 \backslash \mathcal{M}_2$ and $\mathcal{M}_2 \backslash \mathcal{M}_1$ (each gene has the same probability to be selected). The picked genes are exchanged, i.e., they pass from one individual to another.
5. **Mutation:** Each gene of each individual is replaced by a random gene according to a given probability. These random draws are done avoiding duplication, i.e., individual with repeated measurement times $\tau_i \neq \tau_j \quad \forall \tau_i, \tau_j \in \mathcal{M}$ with $i \neq j$.
6. **Repeat:** Come back to step 2 until a stopping criterion is satisfied. The returned measurement time set is denoted $\mathcal{M}_{\text{GA}}$.

One pass of steps 2 to 5 is called a generation.

The implementation of the genetic algorithm uses stochastic universal sampling and sigma scaling with unitary sigma coefficient [31]. The crossover probability is 1 and the mutation probability per gene is 0.003. The stopping criterion used is on the number of generations limited to (max. gen.).

The complexity of the implemented GA optimization is

$$O_{\text{GA}} \coloneqq O\Big((\text{max. gen.}) \cdot (\text{pop. size GA}) \cdot O(\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}])\Big), \tag{3.25}$$

where (pop. size GA) is the size of the population.

3.6.5 The Random Trial algorithm

The random trial algorithm samples measurement time sets uniformly at random and evaluates their corresponding costs. The measurement time set with the minimum cost is returned.

The algorithm can be summarized as follows:

1. **Initialization**: a measurement time set $\mathcal{M}^*$ is uniformly sampled on the admissible set. Its cost $\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}^*]$ is evaluated.
2. **Random trial**: a measurement time set $\mathcal{M}^i$ is uniformly sampled on the admissible set.
3. **Evaluation**: the cost $\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}_i]$ is evaluated according to the objective function of the problem.
4. **Selection**: If $\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}_i] < \hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}^*]$, then the newly generated individual $\mathcal{M}_i$ becomes the new $\mathcal{M}^*$.
5. **Repeat**: Come back to step 2 until reaching a sample budget. The individual $\mathcal{M}^*$ is returned. The returned measurement time set is denoted $\mathcal{M}_{\text{RT}}$.

The complexity of the implemented RT optimization is

$$O_{\text{RT}} \coloneqq O\Big((\text{sample. size.}) \cdot O(\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}])\Big). \tag{3.26}$$

3.7 A note on the *training model* and the robustness of the optimal measurement time set

The expected mean squared tracking error estimator can be interpreted as a function of the measurement time set $\mathcal{M}$ with 2 parameters: the *tracking model* and the *training model* (see figure 3.5).
These models are to be distinguished, the reason being that the robustness of the estimator, that is its property of being asymptotically unbiased, only depends on the *training model* as this model is the one used to generate the draws in the Monte Carlo method (see section 2.2). The robustness of the estimator is crucial to ensure the quality of the returned measurement time set $\mathcal{M}$ to lower the expected mean squared tracking error.

In complex practical application, models are approximations of the true dynamic state. If one has access to empirical data, it is, however, possible to circumvent this

problem. Indeed, one can generate the draws of the Monte Carlo method according to the empirical distribution.

The empirical distribution $\hat{P}_*$ associated with the data $y = (y_1, y_2, \ldots, y_n)$ gives equal weight $(1/n)$ to each of the y_i. Formally it is defined a follows.

Definition 3.8 (empirical distribution function)**.** The empirical distribution function associated with the data y is defined by

$$\begin{aligned}\hat{F}_n(z) &= \hat{P}_*(Z \leq z) \\ &= \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{y_i \leq z\}} = \text{fraction of } y_i\text{'s that are less than } z. \end{aligned} \tag{3.27}$$

Using the law of large numbers, one can show that the empirical distribution converges in distribution to the true distribution function of the data. This last result ensures that using empirical distribution as *training model* to generate the draws of the Monte Carlo estimator would make the estimator asymptotically unbiased.

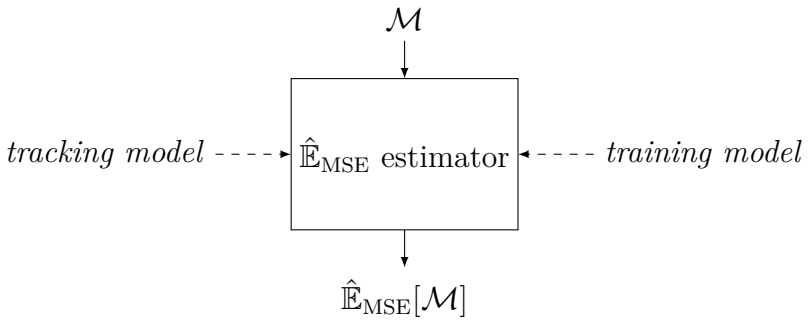

Figure 3.5: The expected mean squared tracking error estimator can be interpreted as a function of the measurement time set $\mathcal{M}$ with 2 parameters: the *tracking model* and the *training model*. These models are to be distinguished, the reason being that the robustness of the estimator only depends on the *training model*.

Chapter 4

Results

This chapter is organized into four sections. In section 4.1 the method to evaluate the tracking performance is described. This is done by introducing four performance indicators (subsection 4.1.1) and describing the method used to estimate these indicators (subsection 4.1.2). To compare the measurement time sets returned the different optimization algorithms, one has to allocate the same computational budget to each of the algorithms. This is done by setting the algorithms' parameters such that the complexity of the algorithms is equivalent (subsection 4.1.3). The parameters to compute the performance indicators are given for the measurements sets corresponding solving the *apriori* and *online* problems (subsection 4.1.4). In section 4.2, the dynamic model used for the tests is given. This model is an approximation of the dynamics of lung tumors motion. In section 4.3, the performance indicators of the measurement time sets solving the *apriori* problem are compared for the five different optimization algorithms. In section 4.4, the performance indicators of the measurement time sets solving the *online* problem are compared with the performance indicators of a measurement time set solving the *apriori* problem.

4.1 Evaluation of the performance of a measurement time set $\mathcal{M}$

4.1.1 Performance of a measurement time set and statistics on the relative gain, the performance indicators

The performance of the measurement time sets is compared to the performance of the tracking using regularly spaced measurement times.

The formal definition of the regularly spaced measurement times is given by:

Definition 4.1 (regularly spaced measurement times)**.**

$$\mathcal{M}_{\text{REG}} \coloneqq \left\{ \text{Round}\left[\frac{kT}{N-1}\right] \middle| \, k = 0, \ldots, N-1 \right\}, \tag{4.1}$$

where Round$[\cdot]$ is the rounding operator.

The performance of a measurement time set $\mathcal{M}$ is defined as the gain on the expected mean squared tracking error.

Definition 4.2 (*gain on* $\mathbb{E}_{MSE}$)**.** The gain on the expected mean squared tracking error is defined as:

$$\textit{gain on } \mathbb{E}_{MSE} \coloneqq \frac{\mathbb{E}_{\text{MSE}}[\mathcal{M}_{\text{REG}}] - \mathbb{E}_{\text{MSE}}[\mathcal{M}]}{\mathbb{E}_{\text{MSE}}[\mathcal{M}_{\text{REG}}]} \tag{4.2}$$

Using the random variable $\text{MSE}(\mathcal{M})$ (definition 3.3), one can define the relative gain as the gain in mean squared tracking error between a measurement time set $\mathcal{M}$ and the regularly spaced measurement times over the same random variables $\{X_t\}_{t=0,\ldots,T}$, $\{Y_t\}_{t=0,\ldots,T}$.

Definition 4.3 (relative gain)**.** The relative gain is the random variable

$$G(\mathcal{M}) \coloneqq \frac{\text{MSE}(\mathcal{M}_{\text{REG}}) - \text{MSE}(\mathcal{M})}{\text{MSE}(\mathcal{M}_{\text{REG}})}, \tag{4.3}$$

where the mean squared tracking error between the measurement time sets $\mathcal{M}$ and the $\mathcal{M}_{\text{REG}}$ is over the same random variables $\{X_t\}_{t=0,\ldots,T}$, $\{Y_t\}_{t=0,\ldots,T}$.

One defines the three following statistics on the relative gain:

$$\textit{expected gain} \coloneqq \mathbb{E}[G(\mathcal{M})] = \mathbb{E}\Big[\frac{\text{MSE}(\mathcal{M}_{\text{REG}}) - \text{MSE}(\mathcal{M})}{\text{MSE}(\mathcal{M}_{\text{REG}})}\Big] \tag{4.4}$$

$$\textit{median gain} \coloneqq m_G(\mathcal{M}) \text{ such that } \begin{cases} \text{P}(G(\mathcal{M}) \leq m_G(\mathcal{M})) = 0.5 \\ \text{P}(G(\mathcal{M}) \geq m_G(\mathcal{M})) = 0.5 \end{cases} \tag{4.5}$$

$$\textit{prob. of positive gain} \coloneqq \text{P}(G(\mathcal{M}) \geq 0) \tag{4.6}$$

These statistics on the relative gain together with the *gain on* $\mathbb{E}_{MSE}$ are referred to as the performance indicators.

4.1.2 Estimation of the performance indicators

To compute the *gain on* $\mathbb{E}_{MSE}$ and the *expected gain*, the expectations $\mathbb{E}_{\text{MSE}}[\mathcal{M}]$, $\mathbb{E}_{\text{MSE}}[\mathcal{M}_{\text{REG}}]$, and $\mathbb{E}[G(\mathcal{M})]$ are estimated using the Monte Carlo method.
The values of $\mathbb{E}_{\text{MSE}}[\mathcal{M}]$ and $\mathbb{E}_{\text{MSE}}[\mathcal{M}_{\text{REG}}]$ are computed according to figure 3.2. One draw $g(\mathcal{M})$ is computed according to figure 4.1. The dynamic model used to generate these draws is referred to as the *testing model* and the estimation is performed with n_{tests} draws. The particle filter used to compute $\{\hat{z}_t(\mathcal{M})\}_{t=0,\dots,T}$ for each of these draws is run with $n_{\text{part. tests}}$ particles. The margin error of a Monte Carlo value is estimated using the procedure described in subsection 2.2. The confidence rate about the margins is $(1-\alpha)$.

The performance indicators *median gain* and *prob. of positive gain* are estimated by respectively the median and the fraction of positive gains over the same n_{tests} draws. The margin error is estimated over $n_{\text{bootstrap}}$ sets using basic bootstrap confidence bounds described in appendix A.2. The confidence rate about the margin is $(1-\alpha)$.

The parameters n_{tests}, $n_{\text{part. tests}}$, $n_{\text{bootstrap}}$ and α are referred to as the *test parameters*.

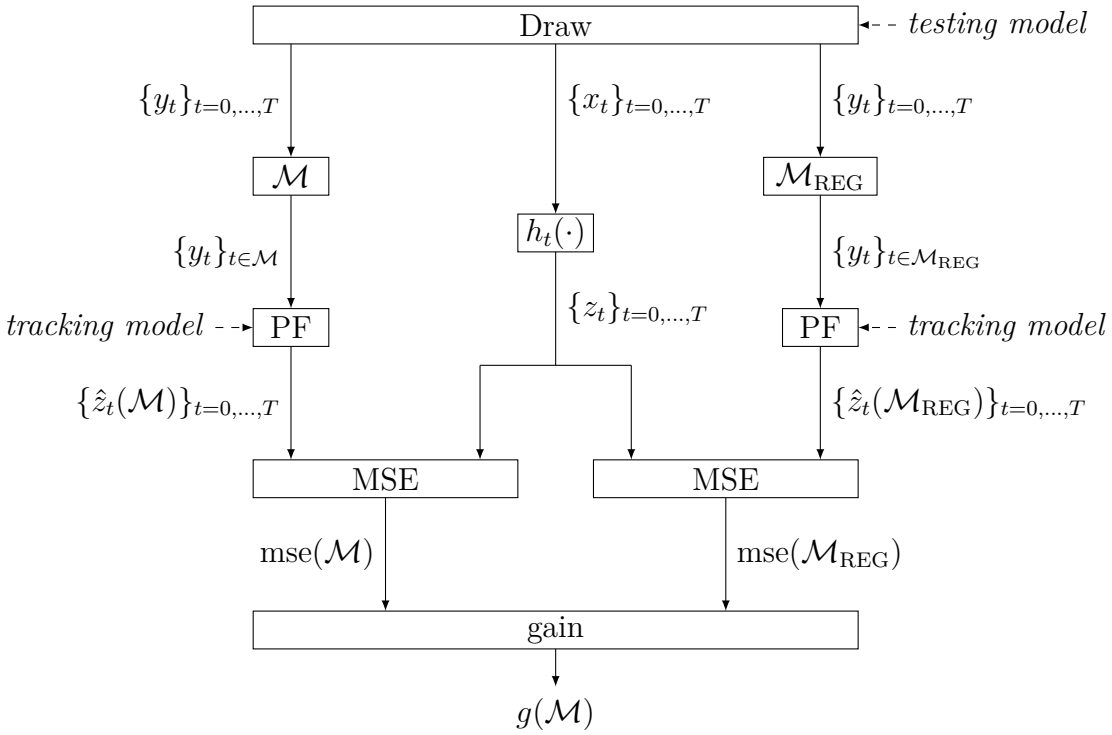

Figure 4.1: Representation of the generation of one draw $g(\mathcal{M})$. The outputs generated by the "Draw" block are drawn according to the *testing model*. The "$h_t(\cdot)$" block uses equation (3.3) to compute z_t^k that corresponds to the x_t^k. The "PF" block represents a particle filter that computes an estimate $\hat{z}_t^k(\mathcal{M})$ of z_t^k from previous intermittent measurements $\{y_t^k\}_{t\in\mathcal{M}}$. The model used by the particle filter is referred to as the *tracking model*. The "MSE" block computes the mean squared of the difference between the inputs. Note that $\text{mse}(\mathcal{M})$ and $\text{mse}(\mathcal{M}_{\text{REG}})$ are computed over the same draw $\{x_t\}_{t=0,\dots,T}$, $\{y_t\}_{t=0,\dots,T}$ and $\{z_t\}_{t=0,\dots,T}$ The gain $g(\mathcal{M})$ is then computed according to equation (4.3).

.

4.1.3 Equivalent computational budget between the different optimization algorithms

To compare the performance of the measurement time sets returned by the different optimization algorithms, one needs to allocate the same computational budget to each algorithm. This can be achieved by setting the parameters of each algorithm such that their number of calls to the particle filter algorithm is equivalent. The parameters of GA are used as a reference and are referred to as the *optimization*

parameters.

The iterative methods (SA, GA, RT) are run with expected MSE estimator, $\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}]$ using the same number of draws, n_{draws}. One evaluation of $\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}]$ for these iterative algorithms is referred to as one cost function evaluation.

Using equations (3.24), (3.25) and (3.26), the number of calls to the particle filter of the iterative optimization algorithm can be written as:

$$\textit{n. calls PF by SA} = 2 \cdot (\text{max. iter.}) \cdot (\text{pop. size SA}) \cdot n_{\text{draws}}, \tag{4.7}$$

$$\textit{n. calls PF by GA} = (\text{max. gen.}) \cdot (\text{pop. size GA}) \cdot n_{\text{draws}}, \tag{4.8}$$

$$\textit{n. calls PF by RT} = (\text{sample. size.}) \cdot n_{\text{draws}}. \tag{4.9}$$

The parameters of the SA and RT algorithms are set such that their number of calls to the particle filter is equivalent to the GA algorithm. This can be achieved by:

$$(\text{sample. size.}) = (\text{max. gen.}) \cdot (\text{pop. size GA}) \tag{4.10}$$

$$(\text{max. iter.}) = (\text{max. gen.}) \tag{4.11}$$

$$(\text{pop. size SA}) = \text{Round}[\frac{(\text{pop. size GA})}{2}] \tag{4.12}$$

Using equations (3.10), (3.21) and (3.23), the number of calls to the particle filter of the direct methods can be written as:

$$\textit{n. calls PF by GF} = (\text{n. calls GF}) \cdot n_{\text{draws GF}}, \tag{4.13}$$

$$\textit{n. calls PF by GB} = (\text{n. calls GB}) \cdot n_{\text{draws GB}}. \tag{4.14}$$

Arbitrarily, the number of particles, $n_{\text{part.}}$ is the same as for the iterative methods. Let n_{draws} refer to the number of draws of the iterative methods and recall that the value of (n. calls GF) and (n. calls GB) are constants. To get the same number of calls to the particles filter as the GA algorithm, the values of parameters $n_{\text{draw GF}}$ and $n_{\text{draw GB}}$ have to be:

$$n_{\text{draw GF}} = n_{\text{draws}} \cdot \text{Round}[\frac{(\text{max. gen.}) \cdot (\text{pop. size GA})}{(\text{n. calls GF})}], \tag{4.15}$$

$$n_{\text{draw GB}} = n_{\text{draws}} \cdot \text{Round}[\frac{(\text{max. gen.}) \cdot (\text{pop. size GA})}{(\text{n. calls GB})}]. \tag{4.16}$$

4.1.4 *Apriori* and *online* comparison

The performance of the five optimization algorithms to solve the *apriori problem* will be compared. The comparison is called the *apriori* comparison.

The best performing optimization algorithm will then be used to solve the *online problem*. The performance of the measurement time set solving the *online problem* will be compared to the one solving the *apriori problem*. This comparison is called the *online* comparison.
As the measurement time set returned by the *online* algorithm is a function of the measurements, the algorithm has to be run on each test. This increases the computation time significantly. Taking this into consideration, the test and optimization parameters for the *online* comparison have been set.

The values of the model parameters, test parameters and optimization parameters are given in table 4.1. Note that the number of particles used for the optimization is different than the number of particles for the filtering.

	parameters	*apriori* comparison	*online* comparison
model	T	30	30
	N	11	11
optimization	n. draws	1,000	200
	n. part.	200	100
	pop. size	100	30
	max. gen.	25	15
test	n. tests	100,000	500
	n. part. tests	1,000	1,000
	n. bootstrap	1,000	1,000
	alpha	0.05	0.05

Table 4.1: Values of the model parameters, test parameters and optimization parameters for the *apriori* and *online* comparison. The given optimization parameters are the parameters used to set the GA algorithm. The parameters of the other optimization algorithms are set according to equations (4.10), (4.11), (4.12), (4.15) and (4.16).

4.2 Modeling the motion of a lung tumor

To illustrate the performances of the above method, a model approximating the one-dimensional lung mobile tumor motion is built.

Mobile tumors tracking is particularly challenging for the implementation of adaptive radiotherapy systems. The movement creates uncertainty about the position of the tumor, which is compensated by margins that will radiate healthy tissues. This effect is particularly important for particle therapy that is less robust to planning errors. One solution to this problem is to guide the delivery of doses by real-time X-ray imaging. In that case, X-ray images are acquired during the treatment to track the tumor. These images feed into a model for predicting the near future of tumor movement, which is necessary to compensate for the delay between image acquisition and device activation, including the computational delay of the predictive algorithm. Unfortunately, radiographic images induce undesirable irradiation. To limit patients' exposure to harmful radiations, the number of X-ray acquisitions has to be used in a parsimonious way. This parsimonious use is described by the acronym ALARA: as low as reasonably achievable.

The model was inspired by research in lung tumor tracking [34], [35] and the respiratory motion tracking system of the CyberKnife treatment device [36], [37]. The model developed does not pretend to have any medical value. It rather aims at illustrating the performance of the added value of selecting measurement times to improve the tracking of a system dynamics under the constraint of limited measurements.

The tumor position to estimate z_t is modeled as a shifted sinusoidal signal with a time-varying amplitude a_t, a time-varying shift b_t and a constant frequency ω. Both a_t and b_t are bounded random walks. Bounds ensure values remain realistic over time. Besides, the constant oscillation frequency ω is picked uniformly at random at the beginning of the process. Each measurement y_t is a noisy version of the position z_t.

One defines the state X_t and its realization $x_t = \begin{bmatrix} a_t & b_t & \omega_t \end{bmatrix}^\top \in \mathbb{R}^3$.

$$X_{t+1}|X_t \sim \begin{bmatrix} \text{clip}(\mathcal{N}(a_t, \sigma_a), \underline{a}, \bar{a}) \\ \text{clip}(\mathcal{N}(b_t, \sigma_b), \underline{b}, \bar{b}) \\ \omega_t \end{bmatrix}, \quad \text{for} \quad t = 0, \dots, T-1, \tag{4.17}$$

$$Y_t|X_t \sim \mathcal{N}(a_t \sin(\omega_t t) + b_t, \sigma_v) \quad \text{for} \quad t \in \mathcal{M}, \tag{4.18}$$

$$Z_t = A_t \sin(\Omega_t t) + B_t \quad \text{for} \quad t = 0, \dots, T, \tag{4.19}$$

$$X_0 \sim \mathcal{U} \begin{bmatrix} [\underline{a}, \bar{a}] \\ [\underline{b}, \bar{b}] \\ [\underline{\omega}, \bar{\omega}] \end{bmatrix}. \tag{4.20}$$

The clipping function is defined as $\text{clip}(x, \underline{x}, \bar{x}) := \min(\max(x, \underline{x}), \bar{x})$. Equation (4.20) indicates that the initial state is uniformly distributed at random on the indicated domain.

To make this model more realistic, the values of $\underline{a}$, $\bar{a}$, $\underline{b}$ and $\bar{b}$ are inspired from [38] and the values of $\underline{\omega}$ and $\bar{\omega}$ are inspired from [39]. All parameters are given in Tab. 4.2.

The model is used as *tracking model* for the particle filter, *training model* for the expected mean squared tracking error estimator and *test model* for the evaluation of the performance indicators.

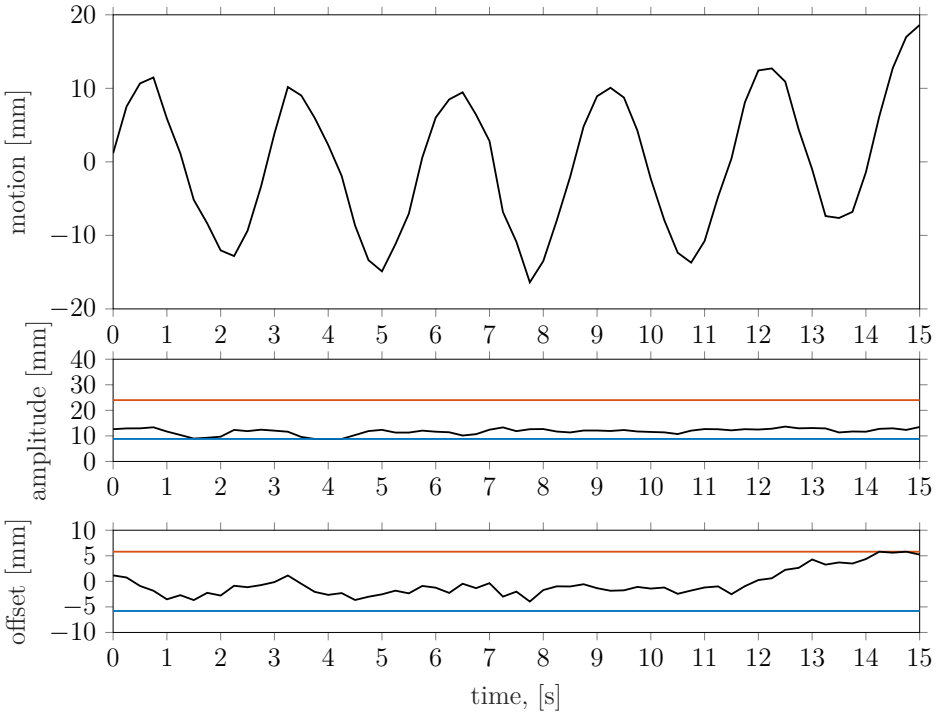

Figure 4.2: Illustration of a realization of the lung tumor dynamics over 15 sec or $T = 60$. The dynamic model used is given at equations (4.17) to 4.20). The parameters used for the model are given in table 4.2. From top to bottom, the figures represent the evolution of the tumor's motion z_t, the state variable amplitude a_t and the state variable offset b_t. The lower and upper bound of the amplitude and the offset are depicted on the figures respectively by a blue and red horizontal line.

$\underline{a} = 8.8[mm]$	$\underline{b} = -5.8[mm]$	$\underline{\omega} = 0.13[rad/ds]$
$\bar{a} = 24[mm]$	$\bar{b} = 5.8[mm]$	$\bar{\omega} = 0.21[rad/ds]$
$\sigma_a = 1[mm]$	$\sigma_b = 1[mm]$	

Table 4.2: Parameters used for the model (4.17)-(4.20). Values of $\underline{a}$, $\bar{a}$, $\underline{b}$ and $\bar{b}$ are inspired by [38]. Values of $\underline{\omega}$ and $\bar{\omega}$ are inspired from [39]. The time interval ds is the duration between two measurements acquisitions. In the following the value $ds = 0.25s$ is used. This delay corresponds to the time required between an image acquisition and the radiotherapy device activation, including the computational delay of the predictive algorithm [37]

4.3 Results for the *apriori* comparison

The performances of the solutions of the *apriori* problem found by the GF, GB, SA, GA and RT optimization algorithms are compared. The model, test, and optimization parameters for the *apriori* comparison can be found in table 4.2.

The evolution of the cost $\hat{\mathbb{E}}_{\text{MSE}}$ with respect to the number of cost function evaluations (one cost function evaluation corresponds to one evaluation of $\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}]$ by an iterative optimization algorithm) is illustrated on figure 4.3.
One can observe that all five optimization algorithms return a measurement time set with associated expected mean squared tracking error significantly lower than the regularly spaced measurements. The final expected mean squared tracking error of the solution returned by the SA, RT, GF and GB optimization algorithms all lay close around 6. The solution returned by GA algorithm performs notably better with a final minimum cost around 5.4. One can note that the difference between GA average and minimum cost decreases over generations before reaching a quasi-convergence meaning that most of the individuals are identical.

The performance indicators given by table 4.3. One can note that the solution returned by the GA algorithm performs better on the four indicators with a *gain on* $\mathbb{E}_{MSE}$ of 37.5% ($\pm$0.3%), an *expected gain* of 23.9% ($\pm$0.3%), a *median gain* of 34.2% ($\pm$0.3%) and a *prob. of positive gain* 76.8% ($\pm$0.3%). The confidence rate about the margins is 95%.

The histogram of relative gain $g(\mathcal{M}_{\text{GA}})$ over the $100,000$ draws is shown in figure 4.4. The histogram can be interpreted as an approximation of the probability density of the random variable $G(\mathcal{M}_{\text{GA}})$. The *prob. of positive gain* for the GA algorithm can be read as the fraction of the area of the histogram above 0. The vertical lines corresponding to the null, *expected* and *median* gain are plotted. The

histogram of the relative gain for the measurement time sets returned by the GF, GB, SA and RT can be found in appendix B.2 respectively on figures B.1, B.2, B.3 and B.4.

Figure 4.5 compares the estimates $\hat{z}_t(\mathcal{M}_{\text{REG}})$ and $\hat{z}_t(\mathcal{M}_{\text{GA}})$ of the exact z_t over one particular realization. One can observe better filtering performances using the measurement time set $\mathcal{M}_{\text{GA}}$. Quantitatively, the relative gain obtained using $\mathcal{M}_{\text{GA}}$ instead of $\mathcal{M}_{\text{REG}}$ is $g = 29.2\%$.

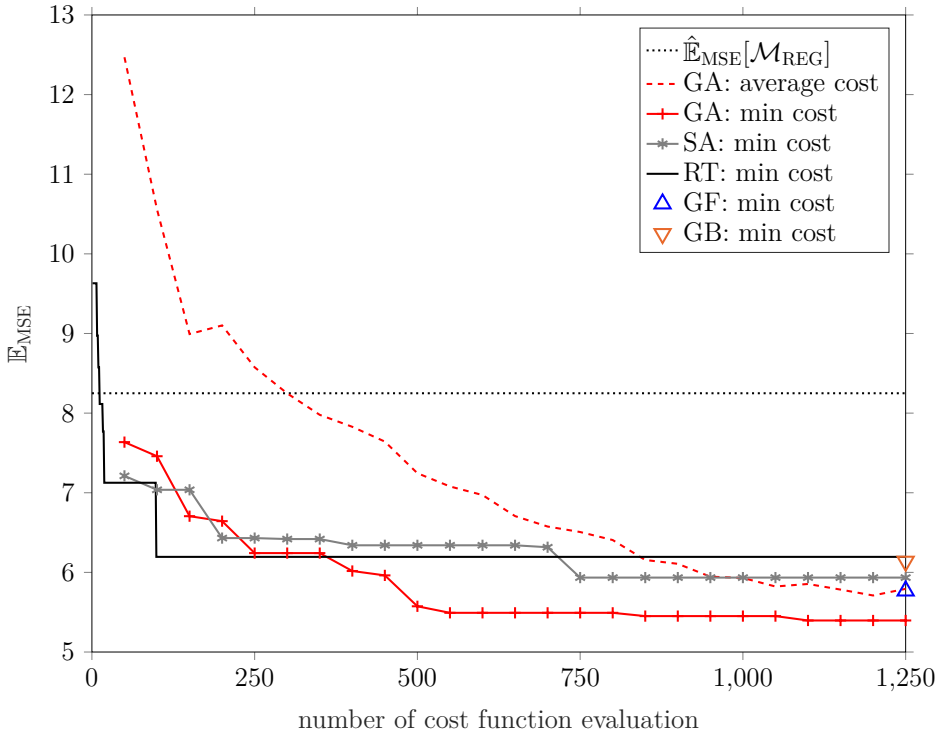

Figure 4.3: Evolution of the minimum cost $\hat{\mathbb{E}}_{\text{MSE}}$ with respect to the number of cost function evaluations for the different optimization algorithms (genetic algorithm (GA), simulated annealing (SA), random trial (RT), greedy forward (GF), greedy backward (GB)). It illustrates the quality of minimization for given computational resources. The evolution of the average cost $\hat{\mathbb{E}}_{\text{MSE}}$ of the population of the GA algorithm at each generation and the expected cost $\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}_{\text{REG}}]$ of the regularly spaced measurement time set are shown as well. One cost function evaluation corresponds to one evaluation of $\hat{\mathbb{E}}_{\text{MSE}}[\mathcal{M}]$ by the iterative optimization algorithm.

	GF	GB	SA	GA	RT
gain on $\mathbb{E}_{MSE}$ ($\pm$ 0.3%)	34.6%	34.0%	33.6%	37.5%	30.1%
expected gain ($\pm$ 0.3%)	19.9%	21.1%	20.3%	23.9%	15.4%
median gain ($\pm$ 0.3%)	32.8%	32.1%	31.0%	34.2%	27.4%
prob. of positive gain ($\pm$ 0.3%)	73.7%	75.6%	74.7%	76.8%	71.2%

Table 4.3: Performance indicators for the measurement time sets returned by the GF, GB, SA, GA and RT optimization algorithms. The indicators are the *gain on* $\mathbb{E}_{MSE}$, the *expected gain*, the *median gain* and *prob. of positive gain*. Their definition is given at subsection 4.1.1 and the method of estimation is explained at subsection 4.1.2. The values of the model, test and optimization parameters can be found in table 4.1. The margin error the performance indicators is the maximum of the margin error of this indicator for the different optimization algorithms. The confidence rate about the margins is 95%.

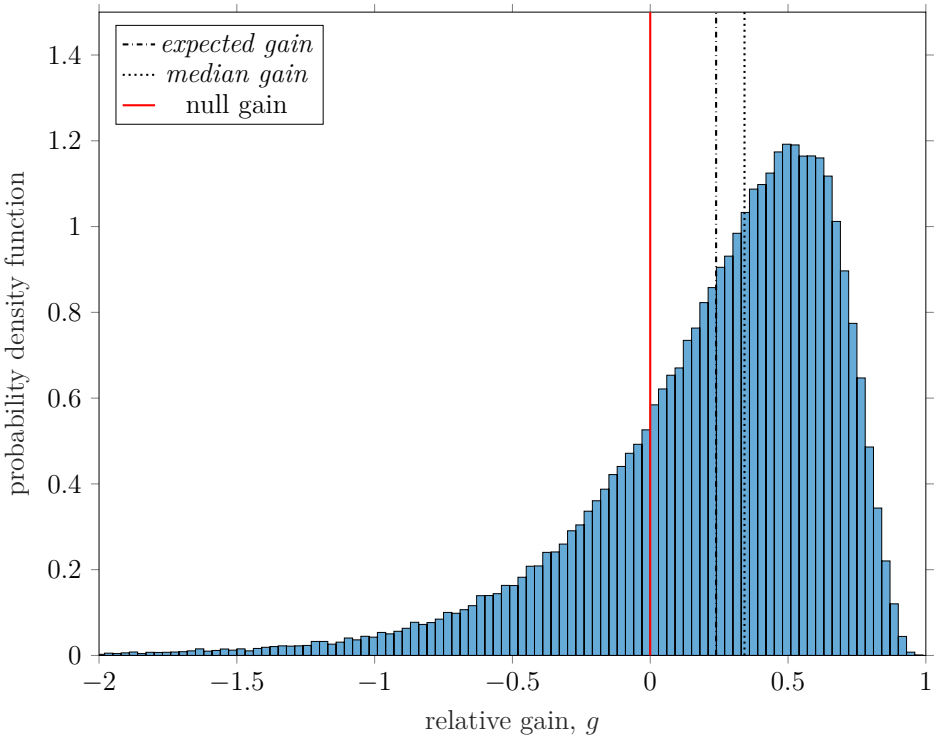

Figure 4.4: Histogram of the relative gain $G(\mathcal{M}_{\mathrm{GA}})$ obtained over 100,000 draws. The *expected gain* is 23.9%, the *median gain* is 34.2% and the *prob. of positive gain* is 76.8%. The method of estimation is explained at subsection 4.1.2. The values of the model, test and optimization parameters can be found in table 4.1.

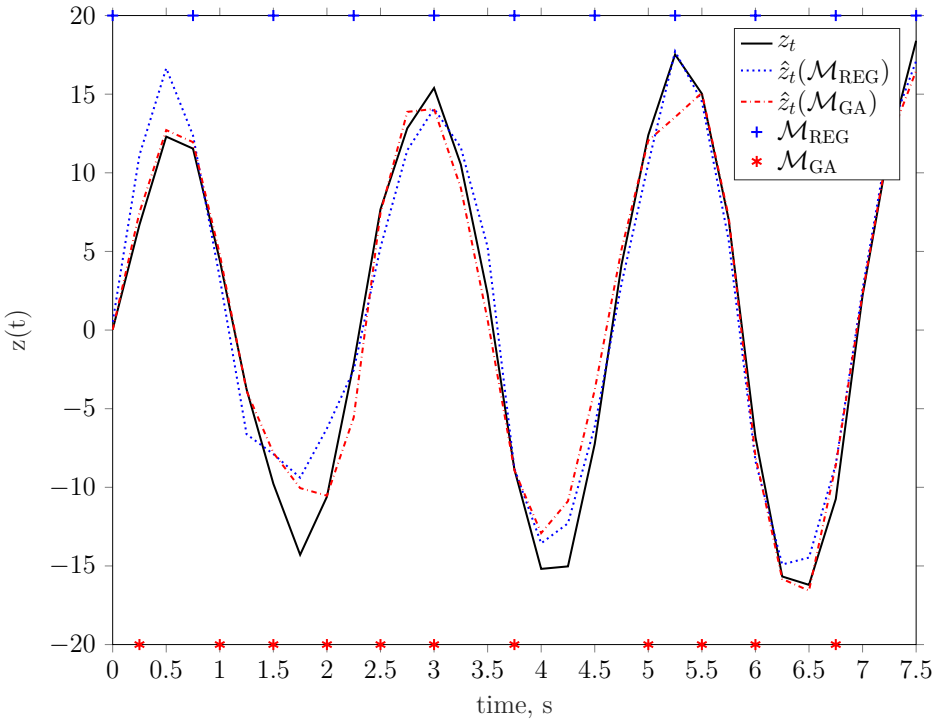

Figure 4.5: Comparison over a single draw of true value z_t with the values $\hat{z}_t(\mathcal{M})$ filtered by the particle filter using the measurement time set $\mathcal{M}_{\text{GA}}$ and $\mathcal{M}_{\text{REG}}$. The relative gain on the complete sequence is $g(\mathcal{M}_{\text{GA}}) = 29.2\%$. Results are simulated from model (4.17)-(4.20). The values of the model, test and optimization parameters can be found in table 4.1

4.4 Results for the *online* comparison

As the GA demonstrated better performances in subsection 4.3, the comparison of the *apriori* and *online* implementations is limited to comparing the solution of the *apriori* and *online* problem returned by the GA optimization algorithm. The model, test and optimization parameters for the *online* comparison can be found in table 4.2. As the *online* algorithm has to be run on each test, the number of tests, as well as the computational cost, has been lowered compared to those used for the *apriori* comparison.

The performance indicators given in table 4.4. One can note that the solution of the *online* problem returned by the GA algorithm performs better on the four indicators with a *gain on* $\mathbb{E}_{MSE}$ of 39.8% (±4.1%), an *expected gain* of 29.8% (±3.5%), a *median gain* of 43.2% (±3.7%) and a *prob. of positive gain* 83.0%

(±3.5%). The confidence rate about the margins is 95%.

The histogram of relative gain $g(\mathcal{M}_{\mathrm{GA}})$ over the 500 draws is shown in figure 4.6. The histogram can be interpreted as an approximation of the probability density of the random variable $G(\mathcal{M}_{\mathrm{GA}})$. The *prob. of positive gain* of the measurement time sets solving the *online problem* can be read as the fraction of the area of the histogram above 0. The vertical lines corresponding to the null, *expected* and *median* gain are plotted. The histogram of the relative gain of the measurement time sets solving the *apriori problem* found by the GA algorithm can be found in appendix B.2 on figure B.5.

It has to be noted that the solution of the *online* found by the GA achieves better performances than any measurement time set tested in the *apriori* comparison in subsection 4.3 with a computational cost for each *online* call of the GA algorithm at least 55 times smaller (see the value of the optimization parameters on table 4.1).

	GA *apriori*	GA *online*
gain on $\mathbb{E}_{MSE}$	29.9% (± 4.4%)	39.8% (± 4.1%)
expected gain	19.9% (± 4.2%)	29.8% (± 3.5%)
median gain	33.9% (± 4.8%)	43.2% (± 3.7%)
prob. of positive gain	74.3% (± 4.1%)	83.0% (± 3.5%)

Table 4.4: Performance indicators for the solutions to the *apriori* and *online problem* returned bu the GA algorithm. The indicators are the *gain on* $\mathbb{E}_{MSE}$, the *expected gain*, the *median gain* and *prob. of positive gain*. Their definition is given at subsection 4.1.1 and the method of estimation is explained at subsection 4.1.2. The values of the model, test and optimization parameters can be found in table 4.1. The margin error the performance indicators is the maximum of the margin error of this indicator for the different optimization algorithms. The confidence rate about the margins is 95%.

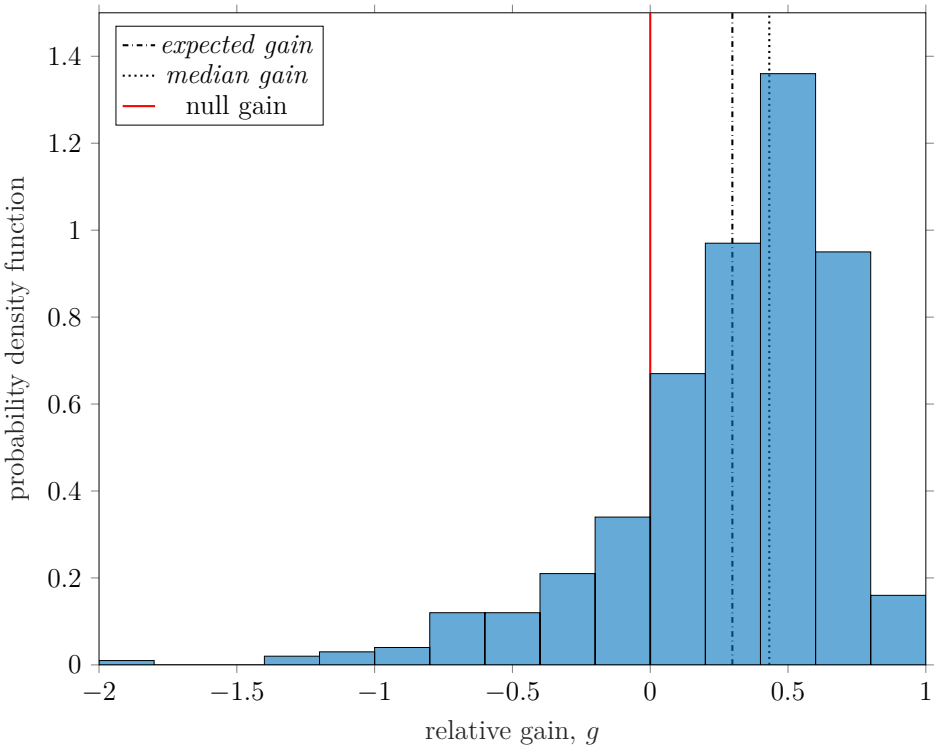

Figure 4.6: Histogram of the relative gain $G(\mathcal{M}_{\mathrm{GA}}(y))$ obtained over 500 draws. The *expected gain* is 29.8%, the *median gain* is 43.2% and the *prob. of positive gain* is 83.0%. The method of estimation is explained at subsection 4.1.2. The values of the model, test and optimization parameters can be found in table 4.1.

Chapter 5

Discussion

This chapter is organized into three sections. In section 5.1, the results obtained in chapter 4 are analyzed. In section 5.2, several ways of using medical data to tailor the models to tumor tracking are presented. Future work and possible improvements of the algorithms are discussed in section 5.3.

5.1 Interpretation of the results

In section 4.3, the performances of the measurement time sets returned by five different optimization algorithms to solve the *apriori* problem are compared to regularly spaced measurement times under four indicators: the *gain on* $\hat{\mathbb{E}}_{MSE}$, the *expected gain*, the *median gain*, and the *prob. of positive gain*. Their definition is given at subsection 4.1.1 and the method of estimation is explained at subsection 4.1.2. The values of the model, test, and optimization parameters can be found in table 4.1.
The five algorithms returned measurement time sets that improved the tracking performances over all these indicators. The genetic algorithm obtained the best results: a *gain on* $\hat{\mathbb{E}}_{MSE}$ of 37.5%, an *expected gain* of 23.9%, a *median gain* of 34.2% and a *prob. of positive gain* of 76.8%.
These results demonstrate the added value of *apriori* selecting measurement times to improve the tracking of a system dynamics under the constraint of limited measurements.

As the genetic algorithm performed the best, its performance to solve the *online* problem are studied in section 4.4. Over 500 tests, the performance of the *online* computed measurement time set is compared to the regularly spaced measurement time set, as well as to the *apriori* computed measurement time set. Overall, the *online* computed measurement time sets performed better than the

regularly spaced measurement time set with a *gain on* $\hat{\mathbb{E}}_{MSE}$ of 39.8%, an *expected gain* of 29.8%, a *median gain* of 43.2%, and a *prob. of positive gain* of 83.0%. The *online* computed measurement time sets improved largely the performances of the *apriori* computed measurement time sets: a *gain on* $\hat{\mathbb{E}}_{MSE}$ improved by 9.9%, an *expected gain* improved by 9.9%, a *median gain* improved by 9.3%, and a *prob. of positive gain* improved by 5.7%.
It has to be noted that these results were achieved with a computational cost for each *online* call of the GA algorithm at least 55 times smaller than in section 4.3. These results emphasize the added value of selecting measurement times to improve the tracking of a system dynamics under the constraint of limited measurements and demonstrate the interest of performing this selection in an *online* manner.

5.2 Tailoring the method to tumor tracking using medical data

As mentioned in section 4.2, the model used to compute the results is an approximation of the real tumor dynamic.

If one had medical data of the tumor's measurements and position, one could make the estimator of the expected mean squared tracking error robust, by using the empirical distribution function as *training model* as explained in section 3.7. The idea is illustrated on figure 5.1.
More formally, if one has D pairs of empirical tumor's measurements and positions, $\{y_t^i, z_t^i\}_{t=0,\dots,T}^{i=0,\dots,D}$, one could compute the estimation of the expected mean squared tracking error of a measurement time set, $\hat{\mathbb{E}}_{\mathrm{MSE}}[\mathcal{M}]$ directly on previously collected medical data by drawing with replacement n_{draws} pairs $\{y_t^k, z_t^k\}_{t=0,\dots,T}$ among the data. The *testing model* could also be replaced by the empirical distribution function. More about statistical inference using data is to be found in appendix A.2.

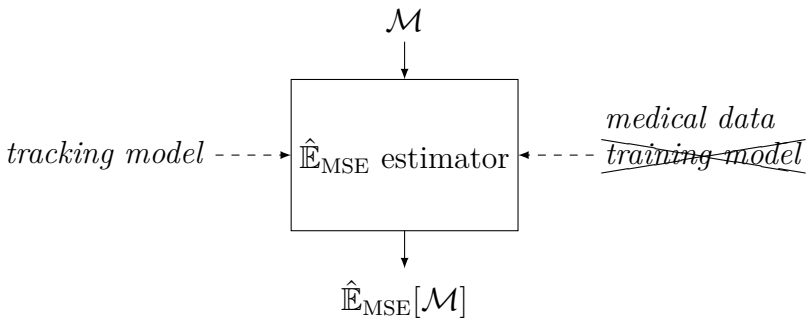

Figure 5.1: Estimation of the expected mean squared tracking error using medical data as the training model.

Medical data could also be used to improve the *tracking model* used by the particle filter. Indeed, one can use the collected data to calibrate the *tracking model*'s parameters. The method to performed this calibration is explained in appendix A.1 and [40].
Data driven model identification algorithms using machine learning algorithms could also be investigated [41], [42].

In order to use the medical images as data, one would need to detect from the images the tumor to be tracked. Over the last decade, deep neural networks have demonstrated outstanding performance in performing the task of object detection [43], [44]. This technology has been used with success to automate brain tumor detection [45]. Recently, the "Kaggle data science bowl challenge" has been submitted to the community to reward scientific work that uses deep learning for lung cancer detection [46]. This approach is currently used for cancer diagnosis and could be extended for the design of detection approach tailored for tracking.

5.3 Improving the algorithm and future work

Further work on the algorithm could focus on improving the speed of the iterative optimization algorithms used to solve the *online problem*. The *online* algorithm requires to solve N times the *apriori problem* over smaller and smaller time horizons. Two considered options to increase the speed of the computation are:

- to reuse the measurement time set found at the previous step
- to change the stopping criterion such that the algorithm would stop once the first measurement time of the set is computed (e.g. for the genetic algorithm, one could decide to stop the computation once the population has converged on the value of the first measurement time τ_1).

Other optimization algorithms such as reinforcement learning could be envisaged to solve the *apriori problem*.

Another extension of the algorithm could be to generalize the measurement time set to allow to acquire measurements from L different sources $Y^i, i = 1, \ldots, L$. The measurement Y^i would be accessible for $t \in \mathcal{M}^i$:

$$Y_t^i | X_t \sim p_t(y_t^i | x_t) dy_t^i \quad \text{for} \quad t \in \mathcal{M}^i, \tag{5.1}$$

The constraint on the measurement size would become $\sum_{i=1}^{L} |\mathcal{M}^i| = N$. In the case of mobile tumor tracking with $L = 2$, a measurement source Y^1 could be an X-ray image in the vertical plan and Y^2 could be an X-ray image in the horizontal plan.

Chapter 6

Conclusion

The problem of finding the optimal measurement time set for particle filtering over a finite time horizon is presented and formalized as a combinatorial optimization problem. A domain of application is the tracking of mobile tumors based on X-ray images. Indeed, as the radiographic images induce undesirable radiation, the number of X-ray acquisitions has to be used in a parsimonious way to limit patients' exposure to harmful radiations.
Finding the optimal measurement time set can be performed either *apriori*, before any measurement acquisition, or *online*, as the measurements are acquired. These problems are referred to respectively as the *apriori problem* and the *online problem*.

The main contribution of this work is to propose a method, nesting of a genetic algorithm, a Monte Carlo algorithm and a particle filter, to find near-optimal solution to these problems.
The performance of the algorithm to solve the *apriori* and *online problem* is measured on a simplified lung tumor model. In comparison with measurements performed at regular time intervals, the measurement time set solving the *apriori problem* reduces the expected mean squared tracking error by 37.5%. The expected tracking performance is improved by 39.8% using the measurement time set solving the *online problem* and this with a significantly smaller computational budget. Overall, the results demonstrate the added value of selecting measurement times for particle filtering.

Several ways of using medical data to tailor the method to mobile tumor tracking are presented. Medical data can, among others, be used to ensure the method robustness and to calibrate the model parameters. Future work could focus on increasing the speed of the optimization algorithm solving the *online problem* as well as extending the problem to combine measurements from different sources.

Bibliography

[1] M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188, 2002.

[2] Olivier Cappé. Ten years of hmms. 2001.

[3] Carine Hue, J-P Le Cadre, and Patrick Pérez. A particle filter to track multiple objects. In *Proceedings 2001 IEEE Workshop on Multi-Object Tracking*, pages 61–68. IEEE, 2001.

[4] Jung Uk Cho, Seung Hun Jin, Xuan Dai Pham, Jae Wook Jeon, Jong Eun Byun, and Hoon Kang. A real-time object tracking system using a particle filter. In *2006 IEEE/RSJ international conference on intelligent robots and systems*, pages 2822–2827. IEEE, 2006.

[5] Henrik Andreasson, André Treptow, and Tom Duckett. Localization for mobile robots using panoramic vision, local features and particle filter. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 3348–3353. IEEE, 2005.

[6] Fredrik Gustafsson, Fredrik Gunnarsson, Niclas Bergman, Urban Forssell, Jonas Jansson, Rickard Karlsson, and P-J Nordlund. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on signal processing*, 50(2):425–437, 2002.

[7] Fredrik Gustafsson. Particle filter theory and practice with positioning applications. *IEEE Aerospace and Electronic Systems Magazine*, 25(7):53–82, 2010.

[8] Arjun V Shenoy, Jagadeesan Prakash, KB McAuley, Vinay Prasad, and SL Shah. Practical issues in the application of the particle filter for estimation of chemical processes. *IFAC Proceedings Volumes*, 44(1):2773–2778, 2011.

[9] Mayank Shekhar Jha, Mathieu Bressel, Belkacem Ould-Bouamama, and Genevieve Dauphin-Tanguy. Particle filter based hybrid prognostics of proton exchange membrane fuel cell in bond graph framework. *Computers & Chemical Engineering*, 95:216–230, 2016.

[10] Francesco Cadini, Enrico Zio, and DIANA Avram. Monte carlo-based filtering for fatigue crack growth estimation. *Probabilistic Engineering Mechanics*, 24(3):367–373, 2009.

[11] Sebastian Thrun. Particle filters in robotics. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 511–518. Morgan Kaufmann Publishers Inc., 2002.

[12] AA Abd Rahni, E Lewis, MJ Guy, B Goswami, and K Wells. A particle filter approach to respiratory motion estimation in nuclear medicine imaging. *IEEE Transactions on Nuclear Science*, 58(5):2276–2285, 2011.

[13] Gregory C Sharp, Steve B Jiang, Shinichi Shimizu, and Hiroki Shirato. Prediction of respiratory tumour motion for real-time image-guided radiotherapy. *Physics in Medicine & Biology*, 49(3):425, 2004.

[14] Shuli Sun, Lihua Xie, and Wendong Xiao. Optimal full-order and reduced-order estimators for discrete-time systems with multiple packet dropouts. *IEEE Transactions on Signal Processing*, 56(8):4031–4038, 2008.

[15] Shuli Sun, Lihua Xie, Wendong Xiao, and Yeng Chai Soh. Optimal linear estimation for systems with multiple packet dropouts. *Automatica*, 44(5):1333–1342, 2008.

[16] Kwesi Rutledge, Sze Zheng Yong, and Necmiye Ozay. Finite horizon constrained control and bounded-error estimation in the presence of missing data. *Nonlinear Analysis: Hybrid Systems*, 36:100854, 2020.

[17] Yan Liang, Tongwen Chen, and Quan Pan. Optimal linear state estimator with multiple packet dropouts. *IEEE Transactions on Automatic Control*, 55(6):1428–1433, 2010.

[18] Wenshuo Li, Zidong Wang, Yuan Yuan, and Lei Guo. Particle filtering with applications in networked systems: a survey. *Complex & Intelligent Systems*, 2(4):293–315, 2016.

[19] Jinguang Chen, Jiancheng Li, and Lili Ma. Nonlinear filtering with multiple packet dropouts. In *2010 International Conference on Image Analysis and Signal Processing*, pages 83–88. IEEE, 2010.

[20] Chen Yang, Huajing Fang, and Bing Shi. Particle filter with markovian packet dropout and time delay. *Journal of the Franklin Institute*, 356(1):675–696, 2019.

[21] Chen Yang and Huajing Fang. Modified particle filter and gaussian filter with packet dropouts. *International Journal of Robust and Nonlinear Control*, 28(8):2961–2975, 2018.

[22] Shu-li Sun and Jian Ding. Optimal filtering for discrete-time systems with one-step random delays and inconsecutive packet dropouts. *Journal of Natural Science of Heilongjiang University*, 28(4):555–560, 2011.

[23] Antoine Aspeel, Damien Dasnoy, Raphaël M Jungers, and Benoît Macq. Optimal intermittent measurements for tumor tracking in x-ray guided radiotherapy. *Medical Imaging 2019: Image-Guided Procedures, Robotic Interventions, and Modeling*, 10951:109510C, 2019.

[24] A. Sano and M. Terao. Measurement optimization in optimal process control. *Automatica*, 6(5):705–714, 1970.

[25] Alexandre Aksenov, Pierre-Olivier Amblard, Olivier Michel, and Christian Jutten. Optimal measurement times for a small number of measures of a brownian motion over a finite period. *arXiv preprint arXiv:1902.06126*, 2019.

[26] Jimmy Olsson. Lecture notes in computer intensive methods in mathematical statistics, June 2019.

[27] Allan Gut. *An intermediate course in probability*. springer, 2009.

[28] Xiao-Li Hu, Thomas B Schon, and Lennart Ljung. A basic convergence result for particle filtering. *IEEE Transactions on Signal Processing*, 56(4):1337–1348, 2008.

[29] Verleysen Michel Lee John. Machine learning : regression, dimensionality reduction and data visualization, December 2019.

[30] Daniel Delahaye, Supatcha Chaimatanan, and Marcel Mongeau. Simulated annealing: From basics to applications. In *Handbook of Metaheuristics*, pages 1–35. Springer, 2019.

[31] Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[32] AJ Umbarkar and PD Sheth. Crossover operators in genetic algorithms: A review. *ICTACT journal on soft computing*, 6(1), 2015.

[33] Stephen J Hartley and Aaron H Konstam. Using genetic algorithms to generate steiner triple systems. In *Proceedings of the 1993 ACM Conference on Computer Science*, pages 366–371. ACM, 1993.

[34] Marcus Isaksson, Joakim Jalden, and Martin J Murphy. On using an adaptive neural network to predict lung tumor motion during respiration for radiotherapy applications. *Medical physics*, 32(12):3801–3809, 2005.

[35] Martin J Murphy, Marcus Isaakson, and Joakim Jalden. Adaptive filtering to predict lung tumor motion during free breathing. In *CARS 2002 Computer Assisted Radiology and Surgery*, pages 539–544. Springer, 2002.

[36] Harold C Urschel, John J Kresl, James D Luketich, Lech Papiez, Raymond A Schulz, and Robert D Timmerman. *Treating tumors that move with respiration*. Springer, 2007.

[37] Mischa Hoogeman, Jean-Briac Prévost, Joost Nuyttens, Johan Pöll, Peter Levendag, and Ben Heijmen. Clinical accuracy of the respiratory tumor tracking system of the cyberknife: assessment by analysis of log files. *International Journal of Radiation Oncology* Biology* Physics*, 74(1):297–303, 2009.

[38] Jennifer Dhont, Jef Vandemeulebroucke, Manuela Burghelea, Kenneth Poels, Tom Depuydt, Robbe Van Den Begin, Cyril Jaudet, Christine Collen, Benedikt Engels, Truus Reynders, et al. The long-and short-term variability of breathing induced tumor motion in lung and liver over the course of a radiotherapy treatment. *Radiotherapy and Oncology*, 126(2):339–346, 2018.

[39] Ruth Wuyts, Elke Vlemincx, Katleen Bogaerts, Ilse Van Diest, and Omer Van den Bergh. Sigh rate and respiratory variability during normal breathing and the role of negative affectivity. *International Journal of Psychophysiology*, 82(2):175–179, 2011.

[40] Christopher F Mooney, Christopher L Mooney, Christopher Z Mooney, Robert D Duval, and Robert Duvall. *Bootstrapping: A nonparametric approach to statistical inference*. Number 95. sage, 1993.

[41] Vincent A Akpan and George D Hassapis. Nonlinear model identification and adaptive model predictive control using neural networks. *ISA transactions*, 50(2):177–194, 2011.

[42] Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.

[43] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[44] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *Advances in neural information processing systems*, pages 2553–2561, 2013.

[45] Hao Dong, Guang Yang, Fangde Liu, Yuanhan Mo, and Yike Guo. Automatic brain tumor detection and segmentation using u-net based fully convolutional networks. In *annual conference on medical image understanding and analysis*, pages 506–517. Springer, 2017.

[46] Kingsley Kuan, Mathieu Ravaut, Gaurav Manek, Huiling Chen, Jie Lin, Babar Nazir, Cen Chen, Tse Chiang Howe, Zeng Zeng, and Vijay Chandrasekhar. Deep learning for lung cancer detection: tackling the kaggle data science bowl 2017 challenge. *arXiv preprint arXiv:1705.09435*, 2017.

[47] Nikolas Kantas, Arnaud Doucet, Sumeetpal S Singh, Jan Maciejowski, Nicolas Chopin, et al. On particle methods for parameter estimation in state-space models. *Statistical science*, 30(3):328–351, 2015.

[48] Anthony Christopher Davison and David Victor Hinkley. *Bootstrap methods and their application*, volume 1. Cambridge university press, 1997.

Appendices

Appendix A

Theoretical annex

This chapter presents complementary theory about the methods and the algorithms used in chapter 3 and 4. The theory presented is inspired by the excellent course given by Jimmy Olsson at the KTH (Kungliga Tekniska högskolan) [26] and the course reference book [27].

A.1 Calibrating the model's parameters

Let ζ be the unknown parameters of the dynamic model. It can be calibrated by finding parameters that maximize the normalized log-likelihood function:

$$\zeta \to l_m(\zeta, \{y_t\}_{t=0,\dots,T}) = \frac{1}{T} \ln L(\zeta, \{y_t\}_{t=0,\dots,T}), \tag{A.1}$$

where the likelihood $L(\zeta, \{y_t\}_{t=0,\dots,T}) = p_\zeta(\{y_t\}_{t=0,\dots,T}) = c_t(\zeta)$ is the normalizing constant of the smoothing distribution [47].

An exact expression of the normalizing constant is in most cases intractable. It can, however, be approximated using a particle filter.

Let $\omega_k(\zeta)$ be the total weight of the particles at time k (c.f. section 2.4) for the particle filter ran with the parameter ζ, the normalizing constant at time t, $c_t(\zeta)$ can be estimated as:

$$c_t^{\mathrm{SISR},N}(\zeta) = \frac{1}{N^{t+1}} \prod_{k=0}^{t} \Omega_k(\zeta). \tag{A.2}$$

One can prove that the estimator is unbiased: $\mathbb{E}[c_t^{\mathrm{SISR},N}(\zeta)] = c_t(\zeta)$.

To calibrate the model, one can then run the particle filter with different values of the parameters ζ^i, estimate the $c_t^{\mathrm{SISR},N}(\zeta^i)$ and select the parameter ζ^i that leads to the maximal normalized log-likelihood value.

A.2 Statistical inference in data using Bootstrap

In statistics, bootstrap techniques are methods of statistical inference based on multiple replication of data from the data set studied, according to resampling techniques [40], [48].

The data y is viewed as a realization of some random variable Y with distribution P_* belonging to some family P of probability distributions where P is some general, possibly non-parametric family of distributions.

In this setting one wants to make inference about some property $\tau = \tau(P_*)$ of the distribution $P_* \in P$ that generated the data y.
The interest variable τ is estimated using some statistic on the data $t(y)$. It has to be noted that the estimate $t(y)$ is a realization of the random variable $t(Y)$.

In the same way, the error $\Delta(y) = t(y) - \tau$ is a realization of the random variable $\Delta(Y) = t(Y) - \tau$.

To assess the uncertainty of the estimator one needs to analyze the cumulative distribution function F_Δ of $\Delta(Y)$. A confidence bound $I_\alpha = (L(y), U(y))$ for τ on level α should satisfy:

$$\begin{aligned} 1 - \alpha &= P_*\Big(L(Y) \leq \tau \leq U(Y)\Big) \\ &= P_*\Big(t(Y) - L(Y) \geq t(Y) - \tau \geq t(Y) - U(Y)\Big) \\ &= P_*\Big(t(Y) - L(Y) \geq \Delta(Y) \geq t(Y) - U(Y)\Big). \end{aligned} \tag{A.3}$$

Thus with:

$$t(y) - L(y) = F_\Delta^{-1}(1 - \alpha/2) \tag{A.4}$$

$$t(y) - U(y) = F_\Delta^{-1}(\alpha/2), \tag{A.5}$$

and it comes that

$$I_\alpha = \Big(t(y) - F_\Delta^{-1}(1 - \alpha/2), t(y) - F_\Delta^{-1}(\alpha/2)\Big). \tag{A.6}$$

Consequently, the error cumulative distribution function F_Δ is needed to make qualitative statements about the estimator. However, $F_\Delta(z)$ is in general unknown.

The bootstrap algorithm deals with this problem by replacing P_* with a data-driven approximation and analyzing the variation of $\Delta(Y)$ using a Monte Carlo simulation from the empirical distribution $\hat{P}_*$.

The empirical distribution function associated with the data y is defined by:

Definition A.1 (empirical distribution)**.** The empirical distribution $\hat{P}_*$ gives equal weight to each y_i:

$$\begin{aligned}\hat{F}_n(z) &= \hat{P}_*(Z \leq z) \\ &= \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{y_i \leq z\}} = \text{fraction of } y_i\text{'s that are less than } z. \end{aligned} \tag{A.7}$$

By the law of large numbers, it holds that $\hat{F}_n(z) \xrightarrow{d} F(z)$ almost surely.

The bootstrap algorithm is the following:

1. Simulate B new data sets Y_b^*, $b \in 1, 2, \ldots, B$, where each Y_b^* has the size of y, from $\hat{P}_*$. Each Y_b^* is obtained by drawing, with replacement, n times among the y_i's.

2. Compute the values $\{t(Y_b^*)\}_{b=1,\ldots,B}$, of the estimator and the bootstrapped errors $\Delta_b^* = t(Y_b^*) - \hat{\tau}$ for $b = 1, \ldots, B$.

As the bootstrapped errors $(\Delta_b^*)_{b=1}^B$ are approximately distributed according to F_Δ, one may use the approximation:

$$F_\Delta^{-1}(p) \approx \Delta^*_{(\lceil Bp \rceil)}, p \in (0, 1), \tag{A.8}$$

where $(\Delta^*_{(1)}, \ldots, \Delta^*_{(B)})$ are the ordered errors.

Using this approximation in equation (A.6) gives the basic bootstrap confidence bound:

$$I_\alpha = \Big(\hat{\tau} - \Delta^*_{(\lceil B(1-\alpha/2) \rceil)}, \hat{\tau} - \Delta^*_{(\lceil B\alpha/2 \rceil)}\Big). \tag{A.9}$$

Appendix B

Additional figures and table

B.1 Additional figures

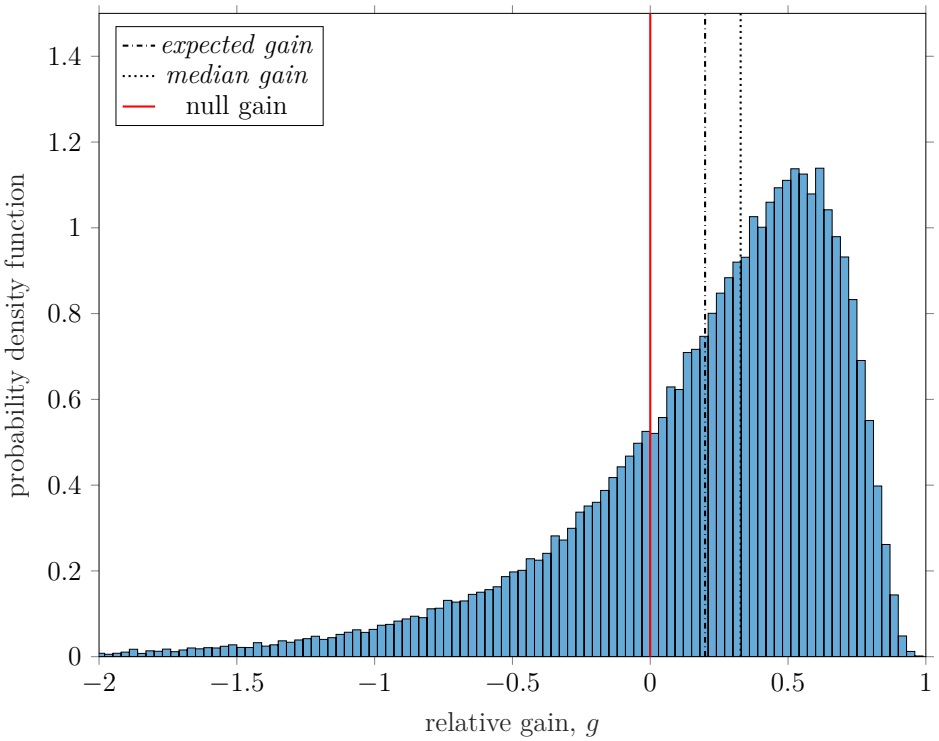

Figure B.1: Histogram of the relative gain $G(\mathcal{M}_{\text{GF}})$ obtained over $100,000$ draws. The *expected gain* is 19.9%, the *median gain* is 32.8% and the *prob. of positive gain* is 73.7%. The method of estimation is explained at subsection 4.1.2. The values of the model, test and optimization parameters can be found in table 4.1.

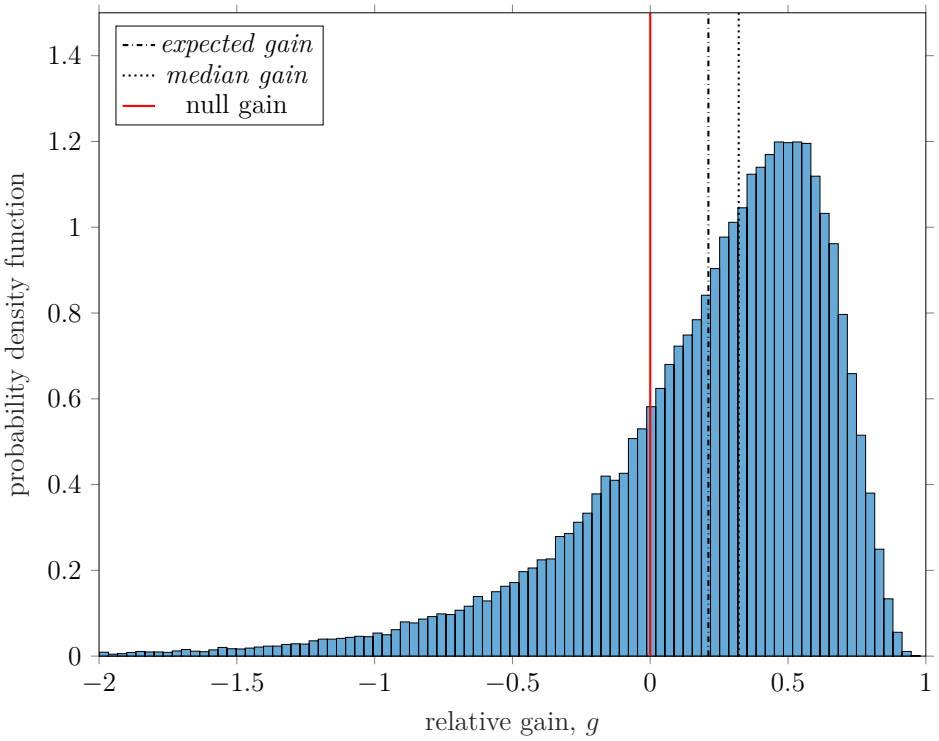

Figure B.2: Histogram of the relative gain $G(\mathcal{M}_{\text{GB}})$ obtained over $100,000$ draws. The *expected gain* is 21.1%, the *median gain* is 32.1% and the *prob. of positive gain* is 75.6%. The method of estimation is explained at subsection 4.1.2. The values of the model, test and optimization parameters can be found in table 4.1.

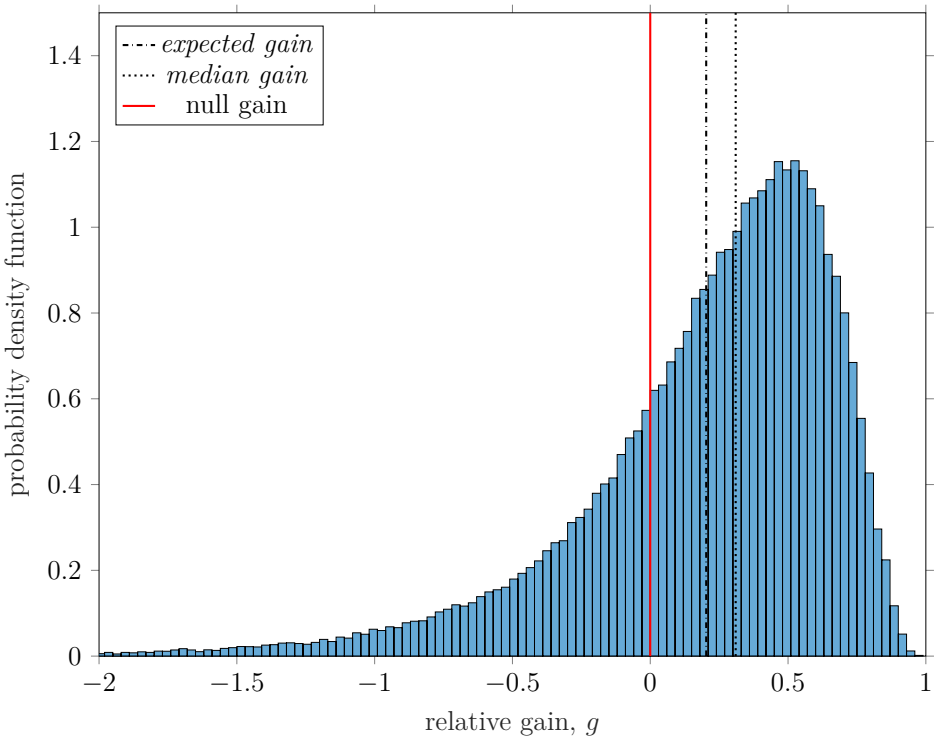

Figure B.3: Histogram of the relative gain $G(\mathcal{M}_{\text{SA}})$ obtained over $100,000$ draws. The *expected gain* is 20.3%, the *median gain* is 31.0% and the *prob. of positive gain* is 74.7%. The method of estimation is explained at subsection 4.1.2. The values of the model, test and optimization parameters can be found in table 4.1.

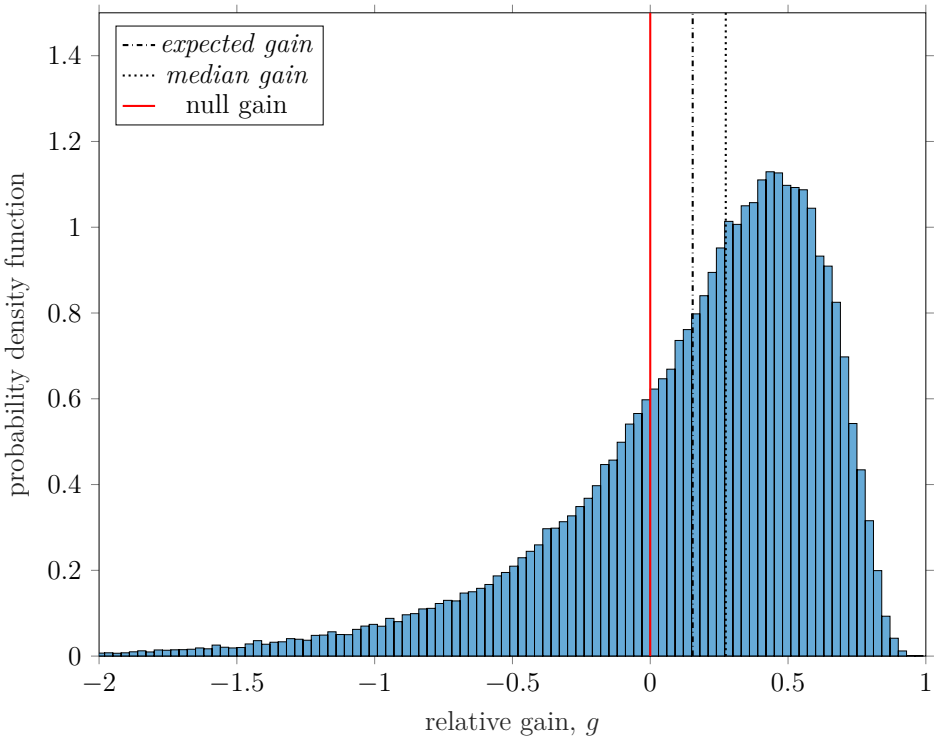

Figure B.4: Histogram of the relative gain $G(\mathcal{M}_{\mathrm{RT}})$ obtained over $100,000$ draws. The *expected gain* is 15.4%, the *median gain* is 27.4% and the *prob. of positive gain* is 71.2%. The method of estimation is explained at subsection 4.1.2. The values of the model, test and optimization parameters can be found in table 4.1.

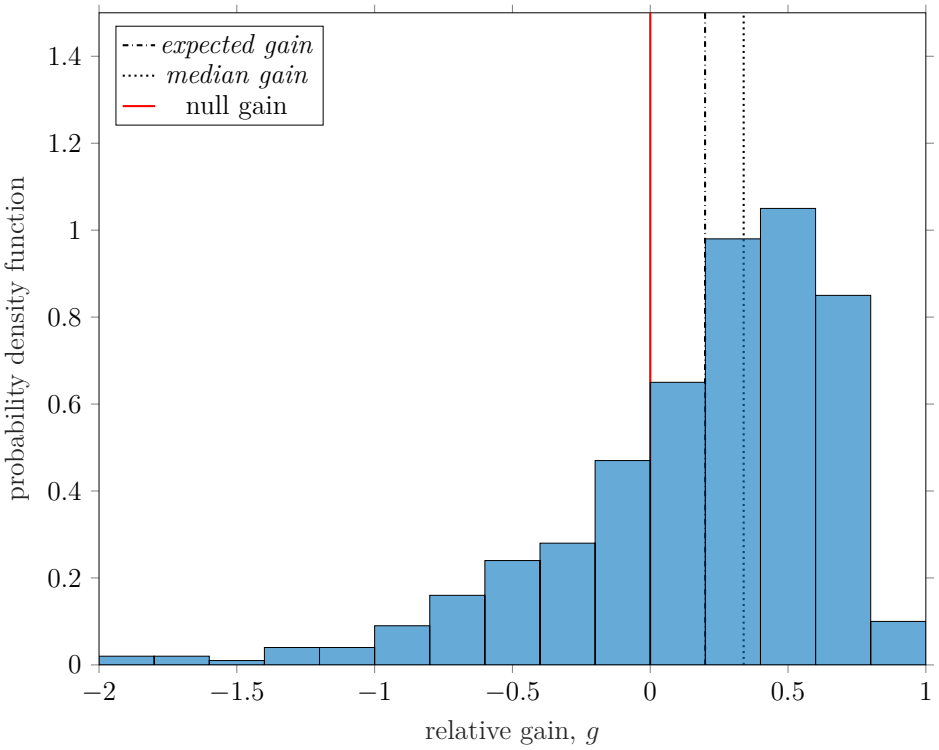

Figure B.5: Histogram of the relative gain $G(\mathcal{M}_{\mathrm{GA}})$ obtained over 500 draws. The *expected gain* is 19.9%, the *median gain* is 33.9% and the *prob. of positive gain* is 74.3%. The method of estimation is explained at subsection 4.1.2. The values of the model, test and optimization parameters can be found in table 4.1.

B.2 Additional table

parameters GF	n. calls GF	286
	n. draws GF	8,741
	n. part	200
parameters GB	n. calls GB	430
	n. draws GB	5,841
	n. part.	200
parameters SA	n. draws	1,000
	n. part.	200
	pop. size SA	50
	max. iter.	25
parameters GA	n. draws	1,000
	n. part.	200
	pop. size GA	100
	max. gen.	25
parameters RT	n. draws	1,000
	n. part.	200
	sample size	25

Table B.1: Table of the parameters value used to allow the different optimization algorithms the same computational budget in the *apriori comparison*.

Appendix C

Matlab code

A Matlab (MathWorks, Natick, Massachusetts, USA) implementation of all the presented algorithms and the code that generate all figures are available on GitHub at
github.com/AmauryGouverneur/Optimal_Measurement_Times_For_Particle_Filtering.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**