Conduite d'expériences par apprentissage actif pour l'identification de systèmes dynamiques biologiques : application à l'estimation de paramètres d'équations différentielles ordinaires

Adel Mezine

NNT : 2016SACLE030

Université Paris-Saclay
École doctorale 580 : Sciences et Technologies de l'Information et de la Communication (STIC)

Thèse de doctorat

Préparé à l'université d'Évry Val d'Essonne

Conduite d'expériences par apprentissage actif pour l'identification de systèmes dynamiques biologiques

Application à l'estimation de paramètres d'équations différentielles ordinaires

Auteur : **Adel Mezine**

Thèse présentée en vue d'obtenir le grade de

Docteur en Informatique de l'Université Paris-Saclay

Soutenue le 11 Octobre 2016, à Evry

Composition du Jury de :

Grégory Batt	Chargé de Recherche, Inria Saclay	Rapporteur
Pierre Geurts	Professeur, Université de Liège	Rapporteur
Jean-Christophe Janodet	Professeur, Université Évry Val d'Essonne	Président du jury
Nathalie Leblanc-Fournier	Maître de Conférences, Université de Clermont-Ferrand	Examinatrice
Florence d'Alché-Buc	Professeur, Télécom ParisTech	Directrice
Véronique Letort	Maître de Conférences, École Centrale Supelec	Co-encadrante

Titre: Conduite d'expériences par apprentissage actif pour l'identification de systèmes dynamiques biologiques

Résumé

Ces dernières années, les progrès continuels des techniques de criblage et de séquençage à haut débit ont nourri la biologie des systèmes, ouvrant la voie à l'identification de systèmes dynamiques biologiques tels que des réseaux de régulation génique. Cependant, l'insuffisance et la mauvaise qualité des données expérimentales se traduisent trop souvent par des estimations incertaines des paramètres d'intérêt des systèmes étudiés : ces incertitudes peuvent être levées en produisant de nouvelles données dans des conditions expérimentales variées, ce qui implique un coût potentiellement élevé. Dans cette thèse, nous proposons un nouvel algorithme d'apprentissage actif, destiné à recommander de manière séquentielle les expériences les plus utiles à l'identification de systèmes dynamiques biologiques modélisés par des équations différentielles. Le problème est formulé sous la forme d'un jeu à un joueur : le joueur se voit attribuer un budget dédié aux expérimentations, et un coût spécifique est affecté à chaque expérience ; à chaque tour, il est amené à choisir une, voire plusieurs expériences réalisées sur le système étudié dans le but de maximiser la qualité de l'estimation, une fois le budget épuisé. Notre approche, intitulée « Experimental DEsign for Network inference » (EDEN), s'appuie sur la classe d'algorithme UCT (Upper Confidence bounds for Trees search) qui allie la souplesse de la recherche arborescente de Monte-Carlo à l'efficacité des algorithmes de bandits multi-bras pour parcourir l'ensemble des séquences d'expériences possibles en privilégiant surtout celles qui sont les plus prometteuses. EDEN présente le grand avantage d'anticiper les expériences suivantes en sélectionnant à chaque tour des expériences sachant qu'elles seront suivies par un certain nombre d'autres expériences. Illustrée sur deux cas d'étude, le réseau de signalisation JAK/STAT et un des réseaux de régulation génique proposé dans la compétition internationale DREAM7, EDEN, entièrement automatique, obtient de très bonnes performances pour un budget limité et un large choix d'expériences (perturbations, mesures).

Mots-clés : Apprentissage actif, planification séquentielle d'expériences, réseaux de régulation génétique, estimation de paramètres, recherche arborescente Monte-Carlo.

Title: Design of experiments by active learning for the identification of dynamical biological systems

Abstract

Continuous progress in screening and high-throughput sequencing techniques in recent years paves the way for the identification of dynamic biological systems such as gene regulatory networks. However, the scarcity of the experimental data often leads to an uncertain estimation of parameters of interest. These uncertainties can be solved by generating new data in different experimental conditions, which induces additional costs. This thesis proposes a general active learning approach to develop tools of sequential experimental design for the identification of dynamical biological systems. The problem is formulated as a one-player game : the player has a budget dedicated for his experiments, each experiment has a different cost ; at every turn, he chooses one or more experiments to be performed on the system with the ultimate aim of maximizing the quality of the estimate, until the available budget is exhausted. The proposed approach called *Experimental DEsign for Network inference* (EDEN), is based on UCT (Upper Confident bounds for Trees) algorithm which combines Monte-Carlo tree search algorithms with multi-arm bandits to perform an effective exploration of the possible sequences of experiments. A strong point of the approach is anticipation : an experiment is selected at each round, knowing that this round will be followed by a number of other experiments, according to the available budget. This generic approach is rolled out in parameter estimation in nonlinear ordinary differential equations using partial observations. EDEN is applied on two problems : signaling network and gene regulatory network identification. Compared to the competitors, it exhibits very good results on a DREAM7 challenge where a limited budget and a wide range of experiments (perturbations, measurements) are available.

Keywords : Active learning, sequential experimental design, gene regulatory networks, parameter estimation, Monte-Carlo tree search

« *La façon de remercier dépend de ce qu'on reçoit* »

Pierre Dac

J'adresse mes plus sincères remerciements envers Florence et Véronique, qui m'ont offert l'opportunité de collaborer avec elles et qui m'ont apporté un soutien infaillible tout ont long de cette thèse.

Je remercie chaleureusement l'ensemble des membres du jury, et l'ensemble du public venu assister à ma soutenance de thèse.

En particuliers, je remercie Pierre Geurts et Gregory Batts qui m'ont fait l'honneur rapporter ma thèse. Je remercie également Nathalie Leblanc-Fournier pour ses remarques en tant qu'examinatrice et Jean-Christophe Janodet pour avoir présidé mon jury de thèse.

Merci aussi à Olivier Teytaud, Michèle Sebag et Jean-Christophe Janodet de m'avoir accorder le privilège de faire partie des membres de mes deux comités de thèse.

Je tient à exprimer toute ma reconnaissance à Michèle Sebag et Artémis Llamosi pour leur collaboration à mes travaux. Nous n'aurions pu atteindre nos objectifs sans l'expertise indispensable de Michèle, et à l'esprit critique d'Artémis. C'est pourquoi je leur adresse un grand merci.

Merci l'ensemble des membres de mon laboratoire d'accueil IBISC, avec qui j'ai partagé de très bons moments, notamment : Néhémy, Romain, Céline, Vincent, Fred, Moussab, Murielle.

Je remercie notamment les membres de l'équipe pédagogique avec j'ai pris plaisir à interagir : Guillaume Hutzler, Judith Benzaki, Philippe Declercq et Jean-Luc Perraudin.

Merci également à l'ensemble des membres de Telecom ParisTech qui m'a accueilli pendant la deuxième moitié de ma thèse. J'ai également partagé de très d'agréables moments avec Claire, Aurélien, Nicolas et Albert.

Enfin je remercie Ha pour m'avoir supporté pendant cette thèse et pour m'avoir apporté de précieux conseils tant sur le plan professionnel que personnel.

Enfin je remercie tous ceux que j'aurai oubliés, et qui ont pu contribuer de près ou de loin au bon déroulement de cette thèse.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Symboles

$\mathcal{X}, \mathcal{Y}, \mathcal{Z}$	ensembles ou espaces vectoriels
X, Y, Z	variables aléatoires réelles
$\mathbf{X}, \mathbf{Y}, \mathbf{Z}$	réalisations de variables aléatoires matricielles
$\mathbf{x}, \mathbf{y}, \mathbf{z}$	réalisations de variables aléatoires vectorielles
x, y, z	réalisations de variables aléatoires scalaires
$\inf_{x \in \mathcal{X}} f(x)$	borne inférieure de f sur $\mathcal{X}$
$\sup_{x \in \mathcal{X}} f(x)$	borne supérieure de f sur $\mathcal{X}$
$\det \mathbf{M}$	déterminant d'une matrice carrée $\mathbf{M}$
$\mathrm{Tr}(\mathbf{M})$	trace d'une matrice carrée $\mathbf{M}$
$\Lambda(\mathbf{M})$	spectre d'une matrice carrée $\mathbf{M}$
$\mathrm{diag}(\mathbf{v})$	matrice diagonale formée des éléments du vecteur $\mathbf{v}$
$\mathbf{M}^{-1}$	inverse d'une matrice carrée $\mathbf{M}$
$\mathbf{M}^\mathsf{T}$, $\mathbf{v}^\mathsf{T}$	transposée d'une matrice $\mathbf{M}$ quelconque ou d'un vecteur $\mathbf{v}$
$\hat{\mathbf{x}}$	estimé de $\mathbf{x}$
$\dot{x}$	dérivée première de x par rapport au temps
$\lvert x \rvert$	valeur absolue d'un scalaire, ou module d'un nombre complexe
$\lvert \mathcal{X} \rvert$	cardinal d'un ensemble
$\lvert\lvert \mathbf{v} \rvert\rvert$	norme du vecteur $\mathbf{v}$
$\lvert\lvert \mathbf{v} \rvert\rvert_2$	norme euclidienne du vecteur $\mathbf{v}$
$\mathbf{x}_{0:K}$	suite des $\mathbf{x}_i$, pour i variant de 0 à K
$\mathcal{A}$	l'espace des actions
$\mathcal{N}(\mu, \sigma^2)$	loi normale d'espérance μ et de variance σ^2
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	loi normale multivariée d'espérance $\boldsymbol{\mu}$ et de covariance $\boldsymbol{\Sigma}$
$\mathbf{Y}$	jeu de données
$\mathbf{Y}_e$	jeu de données associé à une expérience e
$\mathbf{Y}_{\mathbf{s}_n}$	jeu de données associé à une séquence d'expériences de longueur n
$\mathcal{E}$	l'espace des expériences

$\mathcal{E}_n$	l'espace des expériences à la n-ème itération de EDEN
$\mathcal{H}$	l'espace des hypothèses
$\boldsymbol{\mathcal{I}}$	la matrice d'information de Fisher
$\mathcal{M}$	un modèle
$\mathcal{M}_e$	le modèle associé à l'expérience e
$\mathcal{M}_{\mathbf{s}_n}$	le modèle fusionné associé à une séquence d'expériences de longueur n
$\mathcal{P}(\mathcal{E})$	ensemble des combinaisons d'expériences
$\mathcal{P}_B(\mathcal{E})$	ensemble des combinaisons d'expériences d'un coût inférieur à B
$\mathcal{S}$	espace d'états
$\mathcal{U}(a, b)$	loi uniforme sur l'intervalle $[a, b]$
$\mathcal{U}$	ensemble d'exemples non étiquetés
$\mathcal{L}$	ensemble d'exemples étiquetés
$\mathcal{T}$	un estimateur
$\mathcal{V}(\mathbf{Y})$	espace des versions associé à un ensemble d'observations $\mathbf{Y}$
χ^2	statistique du *khi-carré*
B	budget
F	fonction de répartition de la loi normale
F_n	fonction de répartition empirique d'un échantillon de taille n
L	horizon d'un MDP
K	nombre de pas de temps
$K_{i,j}$	constante de dissociation
I	bras ou action choisi d'un bandit
N	nombre de parcours d'arbre
P	loi de probabilité
R	fonction de récompense d'un MDP
$R_{\text{réel}}$	risque réel
$\hat{R}$	risque empirique
S	échantillon d'une variable aléatoire
T	fonction de transition d'un MDP
$U(c, \boldsymbol{\theta})$	utilité générale d'une combinaison expériences c sachant $\boldsymbol{\theta}$
$V_\pi(s)$	fonction de valeur associée à une stratégie π et un état s
a	action ou bras d'un problème bandit
b	coût d'une expérience
d	nombre de gènes
e	expérience
f	fonction d'évolution
g^+	fonction d'activation de Hill

g^-	fonction d'inhibition de Hill
h	fonction d'observation ou fonction de décision
$h_{i,j}$	constante de Hill
ℓ	fonction de perte
m	dimensions du vecteur d'observation
p	densité de P
u	fonction d'utilité
$\tau(\mathbf{y}_{1:n})$	estimation du paramètre $\boldsymbol{\theta}$ à partir de $\mathbf{y}_1 \dots \mathbf{y}_n$
p_i	concentration en protéine associée au gène i
r_i	concentration en ARN messagers transcrits à partir du gène i
$\mathbf{c}$	une combinaison d'expériences
$\mathbf{x}$	vecteur d'états latents
$\mathbf{y}$	vecteur d'états observables
$\mathbf{s}$	séquence d'expériences
$\mathbf{u}$	vecteur des variables de contrôle
$\mathbf{Y}_{1:n}$	série temporelle multivariée de longueur n
α	niveau de confiance dans le test de Kolmogorov-Smirnov
β	budget intermédiaire
ϵ	réalisation d'un bruit
$\boldsymbol{\epsilon}$	vecteur des réalisations de bruit
γ_i	taux de transcription en ARN messager associé à l'expression du gène i
μ	espérance d'une distribution de récompense
$\hat{\mu}_\Theta$	espérance d'une distribution *a priori* sur les paramètres
σ	écart-type
σ^2	variance
$\boldsymbol{\theta}$	vecteur des paramètres
π	stratégie de choix d'action
ψ	stratégie de recommandation
ρ_i	taux de traduction des protéines associé à l'expression du gène i
ξ	plan d'expériences
$\mathcal{K}$	gain de Kalman
$\boldsymbol{\Sigma}$	matrice de covariance
$\hat{\boldsymbol{\Sigma}}^\Theta$	estimé de la matrice de covariance des paramètres $\boldsymbol{\theta}$
$\hat{\boldsymbol{\Sigma}}_k^{\mathcal{Y}}$	matrice de covariance à l'instant t_k du vecteur d'observations
Θ	espace des hypothèses de vecteurs de paramètres $\boldsymbol{\theta}$

Introduction

> « *Les sciences ne tentent pas d'expliquer, elles essaient à peine d'interpréter, elles font principalement des modèles. Par modèle on entend une construction mathématique qui, par l'addition de quelques interprétations verbales, décrit les phénomènes observés.* »
>
> John von Neumann

Motivation

Dans la plupart des disciplines scientifiques, les modèles mathématiques sont devenus des outils incontournables pour la compréhension des phénomènes complexes qui régissent le comportement des systèmes physiques. La construction de tels modèles requiert un nombre suffisant de données expérimentales (Sontag, 2002) qui doivent être pertinentes au regard de notre connaissance *a priori*. Si certains domaines d'application bénéficient d'une abondance de données, avec les fameuses *Big Data*[1], il subsiste encore des domaines comme la biologie des systèmes, pour lesquels les données s'acquièrent péniblement en raison du coût des expériences à mettre en place et des ressources qu'elles nécessitent. C'est en particulier le cas pour des applications qui requièrent la prise en compte de l'aspect dynamique du système étudié comme la chronobiologie médicale, la biosynthèse de composants actifs ou l'identification de réseaux macromoléculaires en biologie des systèmes. La chronobiologie médicale met à profit les rythmes biologiques caractérisant la régulation des fonctions biologiques afin d'améliorer l'efficacité de traitements. Elle est considérée comme l'une des pistes prometteuses dans le cas de certains cancers, pour réduire les effets secondaires en optimisant de manière fine les temps d'administration du traitement. En biologie de synthèse, l'optimisation du processus de production d'un composant actif comme un médicament doit s'appuyer sur un nombre d'expériences qu'il faut choisir. En biologie des

1. En français, *données volumineuses*.

systèmes, l'identification des régulations macromoléculaires à l'œuvre dans la cellule (Neil D. Lawrence, 2010) est nécessaire à la compréhension de la réponse cellulaire à un signal d'entrée (hormone, médicament, stress,...) et pourra donner lieu à terme à du ciblage thérapeutique. Cette identification s'appuie sur l'observation de l'expression génique et des concentrations de protéines au cours du temps et dans différentes conditions. Tous ces exemples impliquant des systèmes dynamiques, les données doivent pouvoir être acquises à différents intervalles de temps afin d'en suivre les évolutions, ce qui fait augmenter de manière drastique le coût des expériences associées à la fois en termes de budget, de temps et de moyens matériels et humains.

Dans ce contexte, privilégier la qualité des expériences à leur quantité paraît indispensable.

La planification d'expériences (Pronzato, 2006) apparaît comme une réponse appropriée aux nouveaux défis auxquels sont confrontés les biologistes. Les approches usuelles de la planification d'expériences s'appuient essentiellement sur des méta-modèles de substitution de type linéaire ou non-linéaire, qui ne prennent pas en considération la nature des phénomènes observés. La planification d'expériences orientée modèle (MBDOE [2]) emploie au contraire des modèles phénoménologiques, lesquels intègrent au sein de leur structure la connaissance *a priori* des phénomènes concernés. L'élaboration de tels modèles est bien plus ardue ; en revanche les bénéfices qu'on peut en tirer sont considérables puisqu'ils permettent d'améliorer notre compréhension des fonctions biologiques et donc d'envisager des applications telles que l'identification de réseaux de régulation génique, l'optimisation de la biosynthèse de composants actifs, la planification d'essais cliniques ciblés.

Contributions

Cette thèse est consacrée au problème de la recommandation automatique d'expériences pour l'identification de systèmes dynamiques partiellement observables dont la dynamique est modélisée par un système d'équations différentielles ordinaires. On prend essentiellement pour cadre d'application les systèmes dynamiques biologiques que sont les réseaux de régulation génique, sur lesquels on peut appliquer un ensemble de perturbations et mesurer leurs effets sur l'évolution de l'*expression génétique* et de la concentration des protéines au cours du temps. Sachant que chaque expérience (association des mesures à d'éventuelles perturbations) se voit attribuer un coût fixe déterminé à l'avance, nous proposons de résoudre ce problème d'identification en prenant en considération le budget alloué à l'expérimentation, ce dernier n'étant, à notre connaissance, jamais pris en compte dans la littérature.

2. Model-Based Design Of Experiment

Nous adoptons une approche séquentielle s'inscrivant dans le cadre des plans d'expériences bayésiens (BED [3]). L'un des points forts de la méthode que nous proposons est sa capacité à recommander une expérience à réaliser en prenant en compte le fait que d'autres expériences la suivront. Il ne s'agit donc pas seulement de maximiser l'information apportée par une expérience de manière indépendante mais bien d'anticiper également sur l'information qui pourra être recueillie lorsque les résultats des expériences suivantes, dans la limite du budget disponible, viendront s'y ajouter.

Pour y parvenir, nous nous appuyons sur la famille d'algorithmes d'arbres de recherche de Monte-Carlo (MCTS [4]), une technique employée avec succès dans le domaine des jeux comme le Go, à l'origine du programme *AlphaGo* décrit par Silver et al. (2016), qui est la première intelligence artificielle à même de vaincre un champion du monde de Go en conditions normales (sans handicap). Ces algorithmes permettent d'évaluer de manière efficace des stratégies à long terme en privilégiant les séquences de mouvements les plus prometteuses. Nous proposons une version de MCTS adaptée à notre problème, en nous plaçant dans la configuration d'un jeu à un joueur dans lequel les actions du joueur correspondent à des réalisations d'expériences, l'utilité d'une série d'expériences étant évaluée par son intérêt pour la tâche d'estimation des paramètres et des états cachés, et le jeu se terminant dès lors que le budget est totalement épuisé.

Cette approche globale est transcrite sous la forme d'un méta-algorithme appelé EDEN (Experimental DEsign for Network inference (Llamosi et al., 2014)), englobant les phases d'estimations, d'expérimentations, et de recommandations d'expériences. Les résultats publiés (Mezine et al., 2015) montrent que l'approche que nous avons développée permet de rivaliser avec les performances humaines sur des problèmes réalistes assez difficile comme celui de la compétition DREAM7 (Meyer et al., 2014).

Organisation du document

Le manuscrit est organisé en trois grandes parties. La partie I introduit le contexte de cette thèse en décrivant brièvement les éléments théoriques et méthodologiques sur lesquels s'appuient nos travaux ainsi que l'état de l'art associé.

Plus précisément, la bonne compréhension des travaux que nous avons menés dans cette thèse et de leurs applications potentielles nécessite tout d'abord quelques connaissances dans des domaines assez variés, issues de la biologie des systèmes (chapitre 1), de la théorie des systèmes dynamiques (chapitre 2), de l'apprentissage et de la théorie des jeux (chapitre 3). Nous commençons donc par introduire en première partie certaines de ces notions fondamentales afin de faciliter la lecture des parties suivantes. Ainsi, on présentera

3. Bayesian Experimental Design
4. Monte Carlo Tree Search

les modalités de l'expression génique au sein de la cellule vivante et sa représentation sous forme de réseau de régulation génique. L'évolution temporelle des quantités associées à cette expression génique nous conduit à considérer le problème sous l'angle de l'identification paramétrique de système dynamique (biologique). Pour cela, on introduira les cadres de l'apprentissage actif et des problèmes de décision séquentielle. Ensuite on présentera un état de l'art des méthodes de planification d'expériences optimales (chapitre 4) pour l'estimation paramétrique de modèles dynamiques. Nous prêterons une attention particulière à la notion de plans d'expérience séquentiels bayésiens, qui constitue le point de départ de l'approche que nous décrivons par la suite.

La deuxième partie est dédié à notre principale contribution : le développement d'une méthodologie complète, efficace, et surtout la plus automatisée possible, pour la conduite d'expériences par apprentissage actif pour l'estimation de paramètres. Dans cette partie, le chapitre 5 décrit formellement le problème visé, en présentant de manière formelle le problème d'optimisation sous-jacent et en le plaçant dans le cadre d'une approche jeu. Le chapitre 6 présente de manière détaillée notre solution au problème, formulé à l'aide d'un méta-algorithme dont on décrira les différentes composantes.

Dans la troisième partie, nous présentons les résultats numériques obtenus avec notre approche sur deux problèmes concrets. Le premier problème, imaginé à partir d'un exemple bien connu de la littérature (JAK/STAT), nous sert à illustrer les différentes étapes de la mise en œuvre de notre méthodologie. Le second problème est tiré de la compétition internationale DREAM7[5] qui était notre objectif applicatif principal tout au long de cette thèse. Nos performances sont comparées à celles des autres participants dans des conditions similaires à celles du challenge.

L'ouvrage s'achève sur un chapitre de conclusions et sur les perspectives d'évolution à moyen et long terme.

Remarque sur les notations : afin de faciliter la lecture, nous avons utilisé la convention de noter en gras les variables de dimension > 1.

5. Dialogue for Reverse Engineering Assessements and Methods

Première partie

Contexte et état de l'art

Chapitre 1

Biologie des systèmes

Sommaire

« On ne peut pas faire une théorie scientifique d'un individu, puisque chacun est unique, mais on peut faire une théorie scientifique des conditions universelles d'existence des individus. »

Alain Prochiantz

Dans cette thèse, on s'intéresse tout particulièrement au problème d'identification de systèmes dynamiques biologiques. Cette section présente quelques éléments de biologie des systèmes afin d'introduire la notion de *réseau de régulation génique* et sa représentation en tant que système dynamique biologique.

1.1 La cellule vivante

Les cellules sont les unités constitutives de tout être vivant. Elles contiennent un support de l'information génétique, l'ADN (Acide DésoxyriboNucléique), et un cytoplasme, solution aqueuse comprenant diverses biomolécules. Elles se divisent en deux grandes catégories, les *procaryotes* (figure 1.1) et les *eucaryotes*[1] (figure 1.2), qui se distinguent selon la présence ou non d'un noyau délimité par une enveloppe nucléaire. Chez les procaryotes, parmi lesquels les bactéries, cette enveloppe est absente et l'ADN circule librement à l'intérieur du cytoplasme. Chez les eucaryotes, l'ADN est contenu dans la membrane nucléaire qui enveloppe le noyau. Les eucaryotes constituent une grande partie des organismes vivants parmi lesquels plusieurs catégories d'organismes uni-cellulaires et multi-cellulaires : les animaux, les champignons, les plantes et les protozoaires.

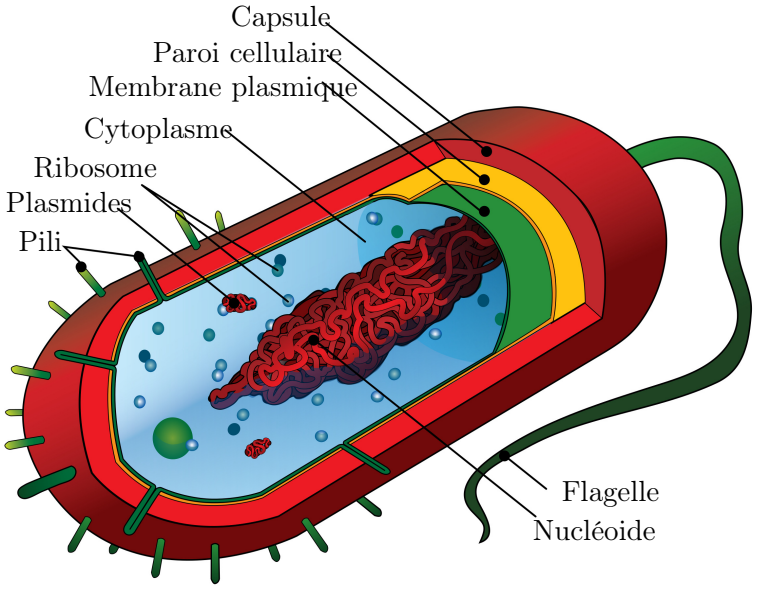

FIGURE 1.1 – Structure d'une cellule procaryote typique

1. le terme eucaryote provient du grec *Eukaryota* « qui signifie littéralement "ceux qui possèdent un véritable noyau", »(*Wikipédia*)

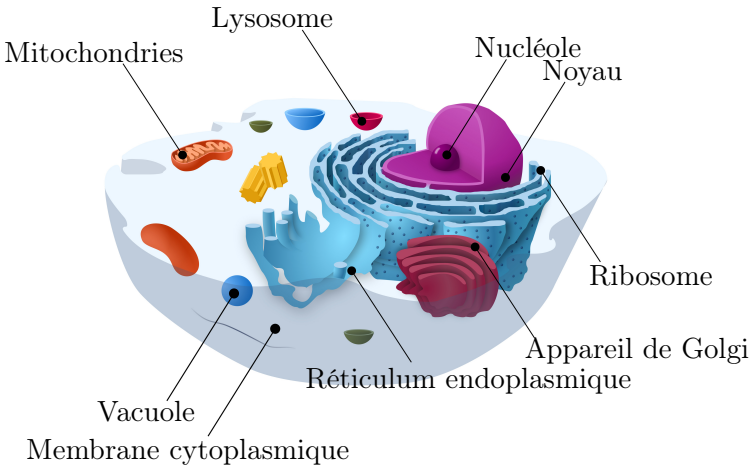

FIGURE 1.2 – Structure d'une cellule eucaryote de type animal

1.2 L'expression génique

L'ADN (figure 1.3) est une macromolécule, présente dans chaque cellule et structurée en chromosomes, qui sert de vecteur à l'*information génétique* nécessaire au fonctionnement de la cellule. Les unités de base de cette information génétique sont les gènes, segments (ou locus) d'ADN. Chez un individu chaque cellule contient cette même information. C'est l'interaction avec l'environnement qui va faire qu'un gène s'*exprime*, induisant un comportement différent des cellules.

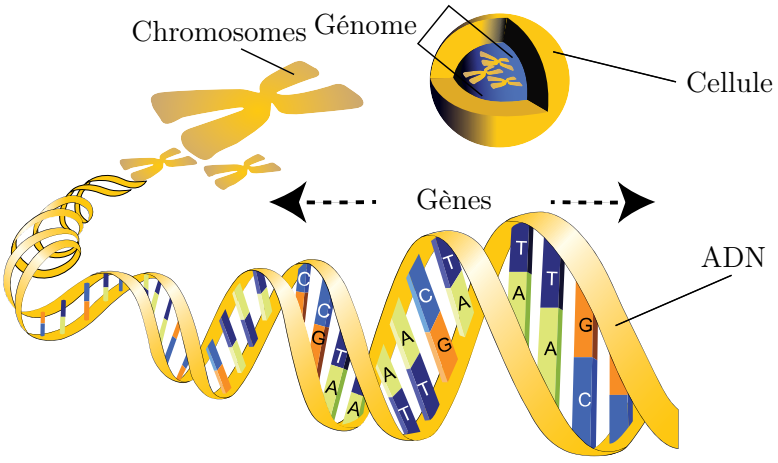

FIGURE 1.3 – Schéma de l'ADN.

Le dogme central de la biologie moléculaire (Crick, 1970) énonce le principe selon lequel l'information génétique est conservée et utilisée : l'ADN est transcrit en ARN, pouvant ou non être traduit en assemblage d'acides aminés constituant la protéine (figure 1.4).

L'ensemble de ces opérations constitue l'*expression génique*, qui fait intervenir deux mécanismes fondamentaux, la *transcription* et la *traduction*.

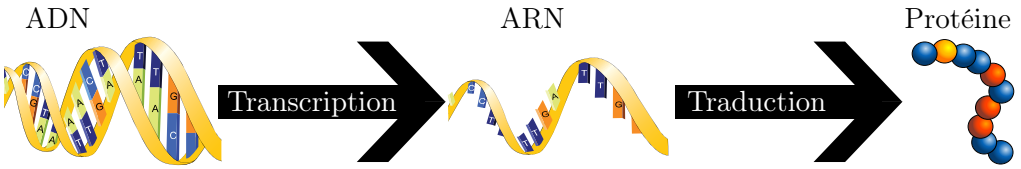

FIGURE 1.4 – Le dogme centrale de la biologie moléculaire

La transcription peut produire deux types d'ARN (Acide Ribonucléique) : les ARN *codants* et *non-codants*. Aussi appelés ARN messagers (ARNm), les ARN codants sont destinés à être traduits en protéines par les ribosomes. Quant aux ARN non-codants, ils ne sont pas traduits en protéines mais peuvent remplir d'autres fonctions. Il existe différents types d'ARN, par exemple les ARN de transports (ARNt figure 1.5) qui interviennent dans le processus de traduction, ou les ARN ribosomiques (ARNr) qui, associés aux protéines, forment des ribosomes. On a longtemps sous-estimé l'importance des ARN non-condants mais de récentes études visent à identifier et prédire leurs structures (Fariza, 2014). On va néanmoins s'intéresser ici exclusivement au rôle joué par les ARN codants.

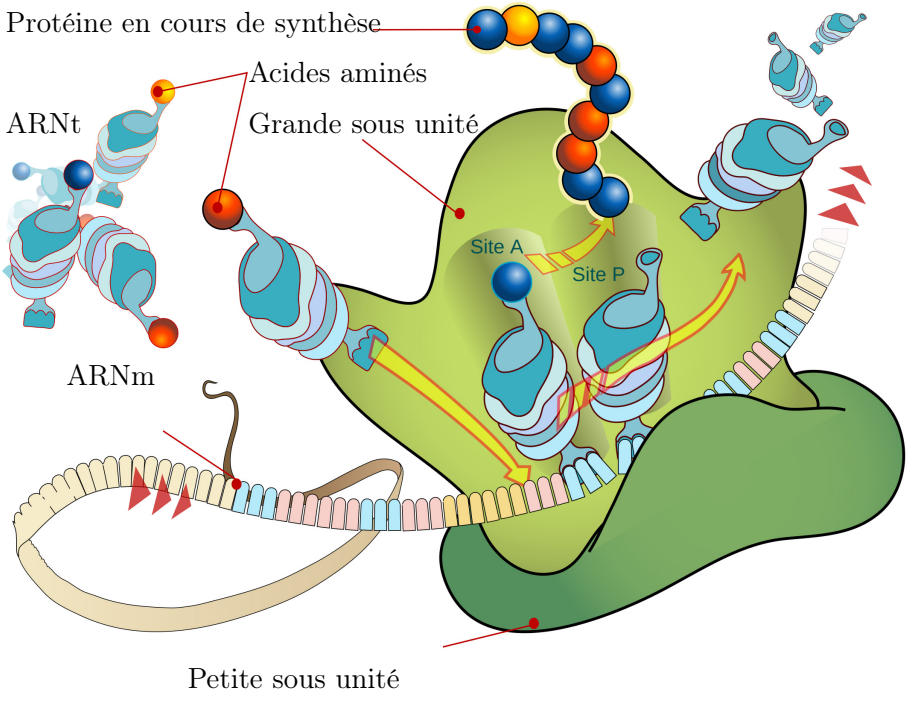

FIGURE 1.5 – Schéma du mécanisme de traduction de l'ARN messager en protéines.

1.3 Réseau de régulation génique

Les protéines peuvent jouer différents rôles au sein de la cellule. Elles peuvent jouer le rôle de récepteur dans la signalisation cellulaire, servir à la construction des ribosomes, ou agir comme catalyseurs (enzymes) permettant ainsi d'accélérer certaines réactions chimiques. Les protéines jouent également un rôle important dans le mécanisme de régulation de l'expression génique. En effet, la protéine peut agir comme *facteur de transcription*. Cela signifie que la protéine peut influer positivement (activateur) ou négativement (inhibiteur) sur la transcription d'un ou plusieurs gènes. Les facteurs de transcription agissent en se fixant sur les séquences régulatrices des gènes (promoteur et opéron) à transcrire. L'ensemble des interactions existantes entre les différents gènes et leurs produits forme ce que l'on appelle un réseau de régulation génique (Brazhnik et al., 2002). On dira qu'un gène i inhibe (ou régule négativement) un autre gène j si le produit de l'expression du gène i empêche le gène j de s'exprimer. Et on dira qu'un gène i active (ou régule positivement) un autre gène j si au contraire l'expression du gène i favorise celle du gène j. En formant un graphe orienté $G = (V, A)$ dont les sommets V représentent les différents gènes et les arcs A représentent les régulations pouvant intervenir, on peut construire un modèle qualitatif représentatif du réseau de régulation.

$$G = \begin{pmatrix} 0 & 0 & 1 \\ -1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

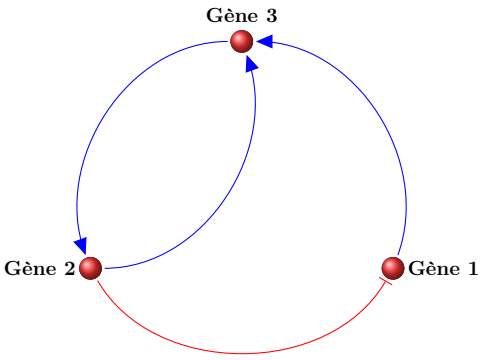

FIGURE 1.6 – Exemple de réseau de régulation génique. Les flèches bleues (resp. rouges) représentent les activations (resp. inhibitions). La matrice d'incidence G fournit une représentation mathématique du graphe.

Ce mode de représentation (figure 1.6) est une simplification qui ne montre que les influences des gènes entre eux. En réalité les régulations entre les gènes ne se font pas

nécessairement de manière aussi directe (figure 1.7). Ces régulations peuvent faire intervenir des protéines, des complexes protéiques (une protéine en transporte une autre) ou des métabolites (sucre, nucléotide, acides-aminé ou lipide).

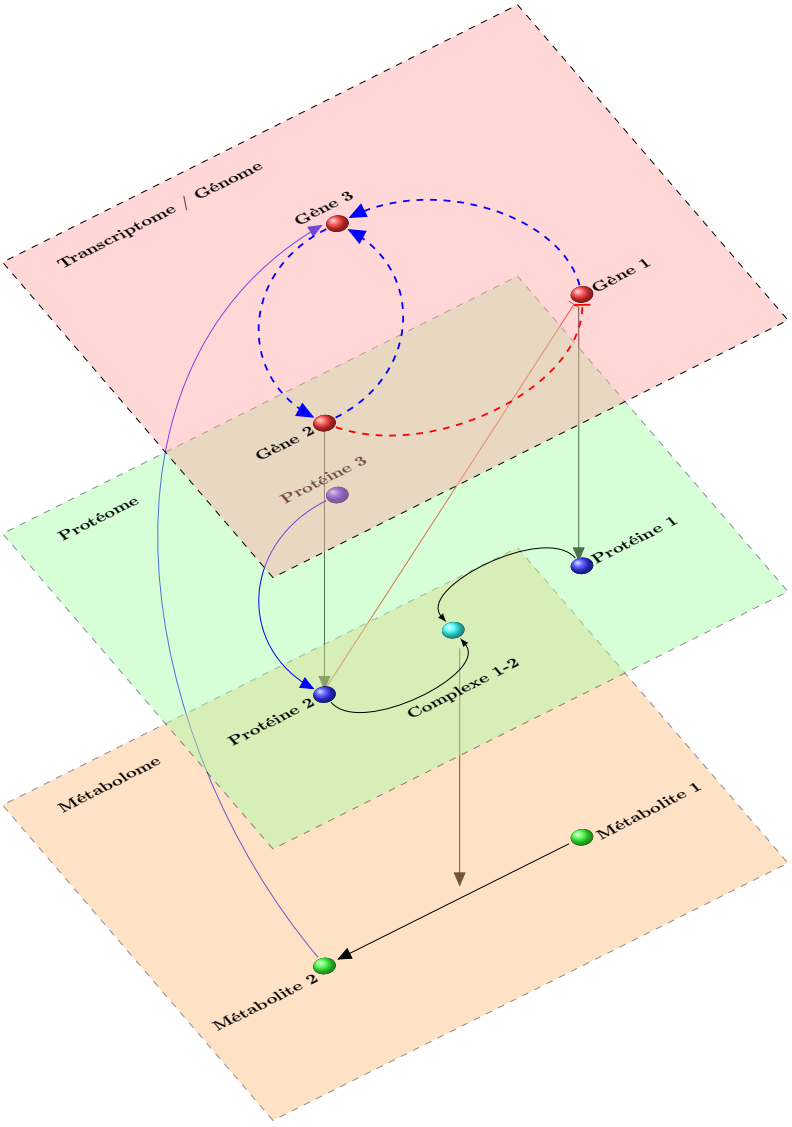

FIGURE 1.7 – Exemple de réseau de régulation biologique. Les flèches en pointillés représentent les régulations indirectes.

L'inférence de réseau de régulation génique consiste à identifier la structure du réseau d'influence entre les gènes intervenant dans un processus de régulation. Ristevski (2013) présente un état de l'art des méthodes d'inférence les plus couramment utilisées. Une fois la structure connue, on peut alors s'intéresser à la dynamique de variation des concentrations d'ARNm et de protéines. Divers formalismes, continus ou discrets, ont été utilisés pour modéliser ces dynamiques et ont donné lieu à une littérature très riche (voir par exemple, la revue de De Jong (2002)). Dans cette thèse, nous avons utilisé celui des équations différentielles ordinaires introduit en tout premier lieu par Chen et al. (1999).

1.4 Modélisation dynamique de la régulation génique

1.4.1 Composantes d'un système dynamique biologique

Un système dynamique est un système qui évolue au cours du temps. Il peut être décrit par les composantes suivantes :

- des variables d'états $\mathbf{x}$ qui représentent les valeurs prises par certaines grandeurs caractéristiques du système,
- des variables de contrôle $\mathbf{u}$ qui représentent les stimuli externes par lesquels le système est influencé : on distingue parmi eux ceux que l'observateur peut manipuler pour contrôler le comportement du système,
- une règle d'évolution f qui décrit les relations entre les variables du systèmes permettant de déterminer l'évolution du système au cours du temps,
- des fonctions d'observation donnant accès à des variables observables $\mathbf{y}$,
- des paramètres $\boldsymbol{\theta}$ qui sont des constantes pouvant intervenir dans les fonctions d'évolution ou d'observation.

On distingue les systèmes dynamiques à temps discret qui peuvent être décrits par un système d'équations de transition et les systèmes dynamique à temps continu qui peuvent être décrits à l'aide d'un système d'équations différentielles ordinaires.

1.4.2 Modélisation des réseaux de régulation génique

Les premiers modèles de réseaux de régulation se sont appuyés sur un formalisme logique, celui des réseaux de René Thomas. Comme le présente Hidde DeJong (De Jong, 2002), ce formalisme discret permet de mettre en lumière la logique des régulations à l'œuvre, mais ne permet pas une modélisation fine du comportement du système qu'on peut souhaiter par exemple, en prévision. Les modèles quantitatifs offrent cette possibilité. Bien qu'ils impliquent des choix difficiles en terme de modélisation, ils ont suscité beaucoup d'intérêt notamment parce que les méthodes d'estimation de leurs paramètres, elles aussi numériques, étaient nombreuses. Parmi ces modèles, les systèmes d'équations différentielles ordinaires (EDO) sont les plus utilisés.

1.4.3 Équations différentielles ordinaires

La plupart des modèles EDO suivent la construction proposée par Chen et ses collègues (Chen et al., 1999) :

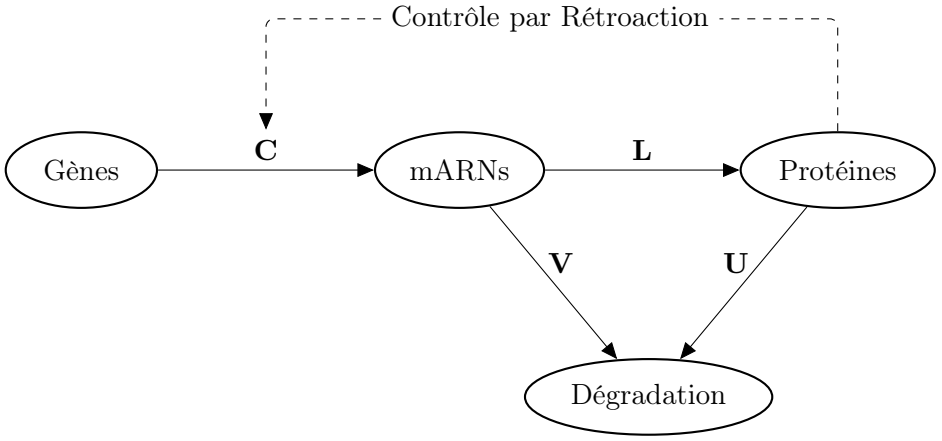

FIGURE 1.8 – Système dynamique simplifié d'un réseau de régulation génique (Chen et al., 1999)

Un réseau de régulation génique associé à d gènes est vu comme un système dynamique dont les états sont les concentrations des ARN messagers associés aux d gènes et les concentrations des d protéines[2], produits de ces d gènes. On note $\mathbf{x}_r(t)$ le vecteur d'état de taille d des concentrations en ARN messager mesurées à l'instant t, et $\mathbf{x}_p(t)$ le vecteur d'état de taille d des concentrations en protéines mesurées à l'instant t. Le vecteur $\mathbf{x}(t) = [\mathbf{x}_r(t), \mathbf{x}_p(t)] = [r_1, \ldots, r_d, p_1, \ldots, p_d]$ de taille $2d$ décrit l'état complet du système. Chen et al. (1999) proposent le modèle suivant :

$$\begin{aligned} \frac{d\mathbf{x}_r}{dt} &= f(\mathbf{x}_p) - \mathbf{V} \cdot \mathbf{x}_r, \\ \frac{d\mathbf{x}_p}{dt} &= \mathbf{L} \cdot \mathbf{x}_r - \mathbf{U} \cdot \mathbf{x}_p, \end{aligned} \tag{1.1}$$

où $\mathbf{L}$ est une matrice diagonale contenant les taux de traduction de chaque protéine, $\mathbf{V}$ est une matrice diagonale contenant les taux de dégradation des ARN messagers, $\mathbf{U}$ est une matrice diagonale contenant les taux de dégradation des protéines. Les premiers travaux proposaient la fonction f de forme linéaire en $\mathbf{x}_p$ (soit $f(\mathbf{x}_p) = \mathbf{C}\mathbf{x}_p$), peu réaliste, puis différents modèles non linéaires ont été étudiés comme le modèle inertiel de Perrin et al. (2003). C'est certainement le modèle des équations non linéaires de Michaelis-Menten avec des cinétiques de Hill, pensé et largement employé dans le cadre de la modélisation des réactions enzymatiques, qui a été le plus utilisé jusqu'ici.

Les équations de Michaelis-Menten ont été proposées pour modéliser les cinétiques enzymatiques dans des réactions biochimiques (détaillées en annexe B.1). Cependant, elles sont aussi employées pour modéliser les réseaux de régulation génique de la manière suivante : on garde les équations différentielles linéaires proposées par Chen et al. (2010)

2. en faisant l'hypothèse restrictive qu'à un gène correspond une seule protéine issue de sa transcription puis traduction.

décrivant l'évolution des concentrations de protéines mais on fait apparaître des non linéarités au niveau de l'action des régulateurs. Considérons le cas où l'expression du gène i est « activée » par le facteur de transcription j. L'équation différentielle suivante :

$$\frac{dr_i}{dt} = \dot{r_i} = \gamma_i \cdot g^+(p_j; K_{ji}, h) - k_r r_i, \tag{1.2}$$

est composée à droite de deux termes. Le premier fait intervenir l'image par la fonction de Hill g^+, définie ci-après, de la concentration p_j de la protéine j et le facteur γ_i, vitesse de transcription du gène i mesuré en $nmol.s^{-1}$. L'autre terme, linéaire, $-k_r r_i$ traduit la dégradation des ARN messagers avec un taux de dégradation k_r qu'on choisit en général identique pour les différents ARN messagers d'un même organisme.

Les fonctions de Hill permettent de traduire l'action « inductrice » d'un facteur de transcription j, présent en concentration $p_j > 0$, sur un gène cible i. Elles se traduisent, respectivement pour l'activation (g^+) et l'inhibition (g^-), par les expressions suivantes :

$$g^+_{j,i}(p_j) = \frac{p_j^{h_{j,i}}}{p_j^{h_{j,i}} + K_{j,i}^{h_{j,i}}} \tag{1.3}$$

$$g^-_{j,i}(p_j) = \frac{K_{j,i}^{h_{j,i}}}{p_j^{h_{j,i}} + K_{j,i}^{h_{j,i}}} = 1 - g^+_{j,i}(p_j) \tag{1.4}$$

Ces fonctions prennent leurs valeurs dans l'intervalle $[0, 1]$. Le paramètre h, sans unité, est défini comme le degré de coopération entre les gènes i et j. Plus h est grand, plus l'activation s'effectue brutalement. Les paramètres notés K sont les constantes de dissociation. Lorsque la concentration p_j atteint la valeur K, le gène régulé est à moitié activé (c'est-à-dire que la concentration de son ARNm vaut $1/2$). On présente les allures de ces courbes pour différentes valeurs de h en figure 1.9.

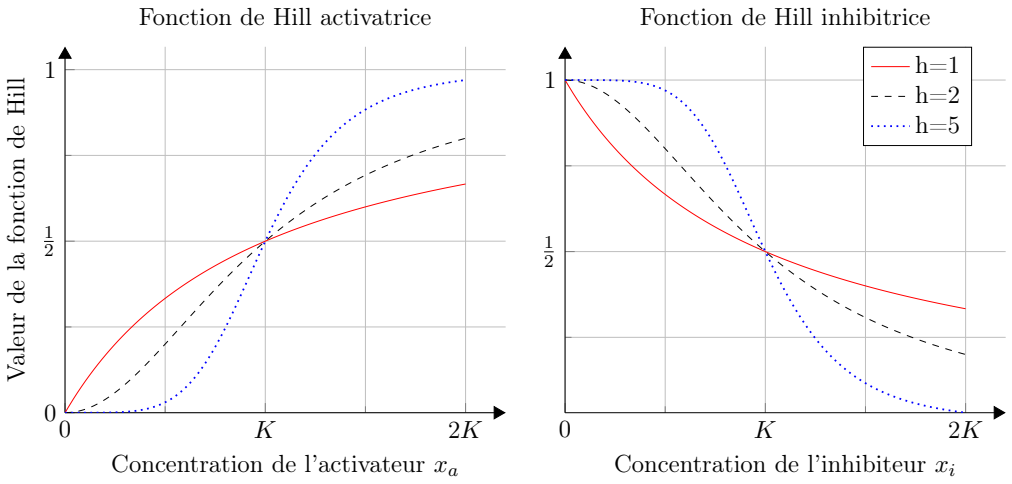

FIGURE 1.9 – Allures des fonctions de Hill. À gauche, le comportement de la fonction de Hill d'activation en fonction de la concentration de l'activateur. À droite, le comportement de la fonction de Hill d'inhibition en fonction de la concentration de l'inhibiteur.

Un réseau de régulation transcriptionnelle de d gènes subit les influences croisées des autres gènes via les facteurs de transcription. Dans la suite de cette thèse, nous choisissons un modèle où les effets des régulations activatrices s'additionnent pour augmenter l'activité du gène ciblé et les effets des régulations inhibitrices inhibitrices se multiplient. On obtient ainsi la forme générale suivante pour l'équation d'évolution des ARN messagers liés à un gène cible i : $\forall i = 1, \ldots, d$,

$$\dot{r_i}(t) = \sum_j \gamma_i \cdot g^+(p_j(t); K_{ji}, h). \prod_j \gamma_i \cdot g^-(p_j(t); K_{ji}, h) - k_r \cdot r_i(t), \tag{1.5}$$

avec l'hypothèse qu'un facteur de transcription j ne peut être simultanément activateur et inhibiteur du même gène.

Un exemple d'un tel modèle sera détaillé en section 7.3 décrivant l'application de notre approche à l'un des problèmes du challenge DREAM. Dans ce cadre d'application, différentes techniques expérimentales seront également évoquées, que ce soit celles permettant la mesure de certaines variables ou bien la perturbation du réseau de régulation. Leur description fait l'objet du paragraphe suivant, qui permet également de mieux appréhender l'enjeu du problème de design expérimental que nous cherchons à résoudre en mettant en lumière les contraintes expérimentales liées à l'acquisition de ce types de données, rares et coûteuses.

1.4 Synthèse

Dans ce chapitre, nous avons introduit certaines notions issues de la biologie des systèmes, en nous focalisant en particulier sur la définition de l'expression génique et de réseau de régulation génique. À travers les formalismes de Michaelis-Menten et de Hill, nous avons montré comment la dynamique des réseaux d'interactions biologiques peut être modélisée sous la forme d'un système d'équations différentielles ordinaires.
Dans la suite, nous introduirons quelques concepts théoriques fondamentaux pour l'analyse de ces modèles, notamment à travers la notion d'*identifiabilité*. Nous présenterons également les outils d'identification paramétrique de la littérature qui nous seront utiles dans la suite de ce manuscrit.

Chapitre 2

Identification paramétrique des systèmes dynamiques

Sommaire

« Tous les modèles sont faux, mais certains sont utiles »

George Box

2.1 Positionnement du problème d'estimation paramétrique des EDO à partir d'observations partielles

Dans le cas de l'identification des réseaux de régulation génique, il n'est pas possible d'acquérir simultanément des mesures de l'expression génique et des mesures de concentrations protéiques. L'estimation des paramètres doit donc être réalisée à partir d'observations partielles. Le cadre classique de l'estimation des EDO par moindres carrés ordinaires ne s'applique pas. En revanche, le problème d'estimation peut être décrit dans le cadre de l'estimation des modèles à espace d'état (Ghahramani, 1998).

2.1.1 Modèle à espace d'état

Un modèle à espace d'état est un modèle à état caché continu. Certaines variables d'état du système sont cachées et représentées dans le modèle par un vecteur $\mathbf{x}$ appelé état latent. Il est généralement décrit par deux types d'équations : une équation d'évolution de l'état caché du système représenté par l'équation 2.1 et une équation d'observation décrit par l'équation 2.2. Soit $t_1, \ldots, t_K$ une suite ordonnée d'instants d'observation distincts.

$$\mathbf{x}(t_{k+1}) = F(\mathbf{x}(t_k)) + \mathbf{w}_k, \tag{2.1}$$

où F est une fonction de $\mathbb{R}^d$ vers $\mathbb{R}^d$ et ϵ_k pour $k = 1, ..., K$ est une suite de réalisations de variables aléatoires représentant le bruit intrinsèque supposé additif.

$$\mathbf{y}(t_k) = h(\mathbf{x}(t_k), t_k) + \epsilon_k, \forall t_1 < t_2, \ldots, t_K \ . \tag{2.2}$$

où ϵ_k pour $k = 1, ..., K$ est une suite de réalisations de variables aléatoires représentant le bruit d'observation supposé additif.

2.1.2 Modèle à espace d'état pour EDO

Soit le système d'équations différentielles ordinaires défini par l'équation suivante :

$$\begin{aligned} \dot{\mathbf{x}}(\mathbf{t}) &= f(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\theta}, t) \\ \mathbf{x}(t_0) &= \mathbf{x}_0 \ . \end{aligned} \tag{2.3}$$

La fonction f est une fonction déterministe qui capture les relations entre les variables du systèmes $\mathbf{x}$, les paramètres $\boldsymbol{\theta}$ et les variables de contrôle $\mathbf{u}$ du système. Le vecteur $\mathbf{x}_0$ représente l'état initial du système à un instant t_0 donné.
Pour le modèle général d'observation présenté en 2.2, le modèle d'EDO peut être présenté

sous une forme complètement discrétisée pour le temps comme l'ont proposé Quach et al. (2007). Les deux équations 2.3 et 2.2 ci-dessus s'écrivent alors :

$$\begin{aligned} \mathbf{x}(t_0) &= \mathbf{x}_0 \\ \mathbf{x}(t_{k+1}) &= \mathbf{x}(t_0) + \int_{t_0}^{t_{k+1}} f(\mathbf{x}(\tau), \mathbf{u}(\tau), \boldsymbol{\theta}, \tau) d\tau \\ \mathbf{y}(t_k) &= h(\mathbf{x}(t_k), t_k) + \epsilon_k, \forall t_1 < t_2, \ldots, t_K. \end{aligned} \tag{2.4}$$

L'intégration permet de considérer des points de temps qui peuvent être irrégulièrement espacés.

Le problème général de l'identification d'un système dynamique partiellement observé, modélisé par des équations différentielles ordinaires (EDO), consiste à estimer à la fois les paramètres et les états cachés à partir d'observations partielles du système.

2.2 Méthodes d'estimation de paramètres d'un modèle à espace d'état à partir d'observations partielles

C'est dans le cadre de modèles quantitatifs à temps discrets qu'ont d'abord été développées les méthodes d'estimation des paramètres et des états cachés à partir d'observations partielles. Avant de décrire un algorithme d'estimation d'un modèle non linéaire tel que décrit plus haut, nous rappelons les principe de filtrage, de prédiction et de lissage utilisés pour estimer les états cachés d'un modèle linéaire à espace d'état (Kalman, 1960).

- le **filtrage**. Connaissant les $k+1$ observations $\mathbf{y}_{0:k}$, il s'agit d'estimer la densité de probabilité $p(\mathbf{x}_k|\mathbf{y}_{0:k})$ (à supposer qu'elle existe). Autrement dit on estime l'état présent à partir des observations passées.
- la **prédiction**. On cherche à prédire la probabilité $p(\mathbf{x}_{k+1}|\mathbf{y}_{0:k})$, à partir des observations précédentes $\mathbf{y}_{0:k}$. C'est-à-dire qu'on estime l'état futur à partir des observations passées.
- le **lissage**. On considère cette fois un ensemble de $K(\geq k)$ observations, on veut estimer l'état caché à l'instant t_k, $p(\mathbf{x}_k|\mathbf{y}_{0:K})$. Dans ce cas il s'agit d'estimer l'état du système à l'instant t_k à partir de l'ensemble des observations antérieures $\mathbf{y}_{0:k-1}$ et ultérieures $\mathbf{y}_{k:K}$.

2.2.1 Filtre de Kalman

À l'instant t_k, on note : $\mathbf{x}(t_k) = \mathbf{x}_k$. Les paramètres du modèle sont les matrices $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{H}$ ainsi que les paramètres des deux bruits $\mathbf{w}_k$ et $\mathbf{v}_k$. Le modèle est défini par les deux

équations suivantes :

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A} \cdot \mathbf{x}_k + \mathbf{B} \cdot \mathbf{u}_k + \mathbf{w}_k \\ \mathbf{y}_k &= \mathbf{H} \cdot \mathbf{x}_k + \mathbf{v}_k \end{aligned} \tag{2.5}$$

Notons que les bruits $\mathbf{w}_k$ et $\mathbf{v}_k$ sont supposés gaussiens. La figure 2.1 présente le modèle graphique de ce système linéaire.

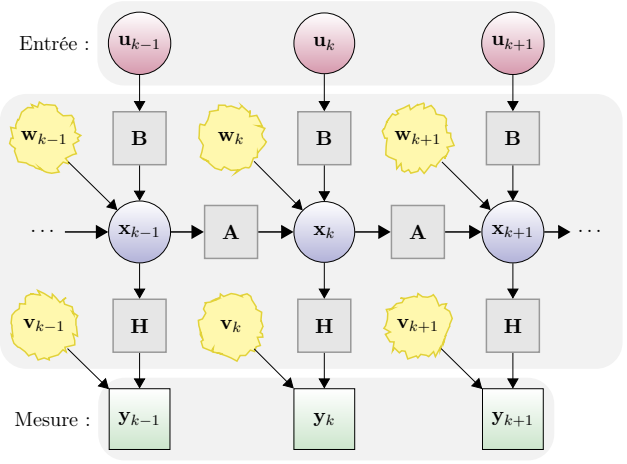

FIGURE 2.1 – Modèle linéaire du filtre de Kalman. *Source* : TEXample [1]

La tâche de filtrage consiste, étant donné un ensemble $\mathbf{y}_{0:k}$ de k+1 observations, à retrouver l'information cachée $\mathbf{x}_k$.

a. Cas linéaire gaussien

L'algorithme de filtrage de Kalman consiste en deux étapes alternées qu'on nomme **prédiction** et **mise à jour**.

Dans le filtre de Kalman, le bruit $\mathbf{w}_k$ est supposé gaussien et donc la variable aléatoire $\mathbf{x}_k$ suit aussi une loi gaussienne. L'état du modèle à chaque instant t_k est alors entièrement caractérisé par les deux premiers moments, l'espérance $\mathbf{x}_k$ et la covariance $\mathbf{\Sigma}_k$.

a..1 Prédiction L'état du système à l'instant k est propagé sur une itération à l'aide du modèle d'évolution pour prédire l'état à l'instant suivant. La mise à jour de matrice de covariance est donnée quant à elle par une forme close.

$$\begin{aligned} \hat{\mathbf{x}}_{k+1|k} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k \\ \hat{\mathbf{\Sigma}}_{k+1|k} &= \mathbf{A}\mathbf{\Sigma}_k\mathbf{A}^\top + \mathbf{Q}_k \end{aligned} \tag{2.6}$$

1. Auteur : Burkart Lingner. Version modifiée.

a..2 Mise à jour Cette phase sert à corriger la prédiction de l'état du système, à partir des observations.

$$\begin{aligned} \boldsymbol{\mathcal{K}}_k &= \hat{\boldsymbol{\Sigma}}_{k+1|k}\mathbf{H}^\mathsf{T}(\mathbf{H}\hat{\boldsymbol{\Sigma}}_k\mathbf{H}^\mathsf{T} + \mathbf{R}_k)^{-1} \\ \mathbf{x}_{k+1} &= \hat{\mathbf{x}}_{k+1|k} + \boldsymbol{\mathcal{K}}(\mathbf{y}_k - \mathbf{H}\hat{\mathbf{x}}_{k+1|k}) \\ \boldsymbol{\Sigma}_{k+1} &= (\mathbf{I} - \boldsymbol{\mathcal{K}}_k\mathbf{H})\hat{\boldsymbol{\Sigma}}_{k+1|k} \end{aligned} \tag{2.7}$$

Le filtre de Kalman peut être vu comme un estimateur récursif de l'erreur au sens des moindres carrés.

b. Cas non-linéaire

b..1 Filtre de Kalman étendu. Le filtre de Kalman classique n'étant pas conçu pour traiter des problèmes non linéaires, de nombreuses variantes ont été proposées pour étendre son champ d'application. Anderson et al. (1979), Anderson and Moore (2012) proposent un filtre de Kalman étendu (EKF - **E**xtended **K**alman **F**ilter) qui consiste à linéariser par approximation de Taylor les composantes non-linéaires (f ou h) puis à appliquer le filtre de Kalman classique sur le modèle linéarisé. Les fonctions du modèle n'ont plus besoin d'être linéaires ; elles doivent cependant être différentiables. Étant données f et h deux fonctions linéaires différentiables, leurs jacobiennes associées sont données par les équations (2.8) :

$$\begin{aligned} \mathbf{A}_k &= \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k-1|k-1},\mathbf{u}_k} \\ \mathbf{H}_k &= \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k|k-1}} \end{aligned} \tag{2.8}$$

Malheureusement, dans la pratique, la non-linéarité de certains modèles rendent trop grossière l'approximation faite par EKF, puisqu'elle peut conduire à un phénomène de divergence. C'est la raison pour laquelle d'autres solutions alternatives ont été proposées.

b..2 Filtre de Kalman sans parfum. Le filtre de Kalman sans parfum (UKF - Unscented Kalman Filter), est une alternative à EKF pour appliquer le filtre de Kalman sur des problèmes fortement non-linéaire. Au lieu de linéariser le modèle, Julier and Uhlmann (1997) proposent de linéariser la distribution gaussienne à l'aide d'un nombre fini de points auxquels on affecte des poids. Cet ensemble de points pondérés, aussi appelés *sigma-points* (2.9), est construit de telle sorte que les propriétés statistiques (espérance

et variance) de la distribution d'origine soient conservés. Étant donné la variable aléatoire $X \in \mathbb{R}^L$, d'espérance $\overline{\mathbf{x}}$ et de covariance $\mathbf{\Sigma}_{\mathcal{X}}$, UKF crée un échantillon de $2L+1$ sigma-points distribués selon l'équation (2.9)

$$\begin{aligned} \mathcal{X}_0 &= \overline{\mathbf{x}} \\ \mathcal{X}_i &= \overline{\mathbf{x}} + \left(\sqrt{(\lambda+L)\mathbf{\Sigma}_{\mathcal{X}}}\right)_i, \forall i = 1 \dots L \\ \mathcal{X}_i &= \overline{\mathbf{x}} + \left(\sqrt{(\lambda+L)\mathbf{\Sigma}_{\mathcal{X}}}\right)_{i-L}, \forall i = L+1 \dots 2L \\ \text{avec } \lambda &= \alpha^2(L+\kappa) - L \end{aligned} \tag{2.9}$$

où le terme $\left(\sqrt{(\lambda+L)\mathbf{\Sigma}_{\mathcal{X}}}\right)_i$ correspond à la i^e colonne de la matrice $\sqrt{(\lambda+L)\mathbf{\Sigma}_{\mathcal{X}}}$, les paramètres λ et κ sont des facteurs d'ajustement d'échelle, tandis que le paramètre α est utilisé pour contrôler la dispersion des sigma-points autour de l'espérance $\overline{\mathbf{x}}$.

Chaque sigma-point $\mathcal{X}_i$ se voit attribuer un poids $\omega_i^{(E)}$ déterminer par les équations suivantes :

$$\begin{aligned} \omega_0^{(E)} &= \frac{\lambda}{L+\lambda} \\ \omega_0^{(C)} &= \frac{\lambda}{L+\lambda} + (1 - \alpha^2 + \beta) \\ \omega_i^{(E)} &= \omega_i^{(C)} = \frac{1}{2(L+\lambda)}, \forall i = 1 \dots 2L \end{aligned} \tag{2.10}$$

Chaque sigma-point $\mathcal{X}_i$ est *propagé* à travers la fonction f non-linéaire pour construire les images $\mathcal{Y}_i$. On obtient ainsi :

$$\mathcal{Y}_i = f(\mathcal{X}_i), \forall i = 0, \dots, 2L \tag{2.11}$$

c. Estimation des paramètres

Jusqu'à présent, nous avons décrit une méthode qui consiste essentiellement à estimer les états cachés. Or nous sommes également intéressés par l'estimation des paramètres du modèle. Pour prendre en compte ces paramètres, l'astuce consiste à considérer le vecteur des paramètres $\boldsymbol{\theta}$ comme faisant partie des états cachés. Pour cela, on traduit l'invariance des paramètres par l'équation d'évolution suivante, dans le cas modèle à temps discret :

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1}, \forall k \tag{2.12}$$

Où bien dans le cas d'un modèle d'évolution à temps continu :

$$\dot{\boldsymbol{\theta}}(t) = 0, \forall t \tag{2.13}$$

On se ramène ensuite au cas standard en définissant artificiellement une *variable augmentée* $\mathbf{x}^a = [\boldsymbol{\theta}, \mathbf{x}]$, incorporant les paramètres au vecteur des états cachés. Cette approche a par exemple été employée dans par Quach et al. (2007) et d'Alché Buc and Brunel (2009), pour des problèmes d'une nature semblable à ceux que nous traiterons dans la partie suivante.

L'algorithme 1 décrit la procédure UKF sous sa forme dédiée à l'estimation des paramètres.

Algorithm 1: UKF : Filtre de Kalman sans parfum pour l'estimation de paramètres (Wan et al., 2000)

`/* Initialisation */`

$\hat{\mathbf{x}}_0 = \mathbb{E}[\mathbf{x}_0]$;

$\boldsymbol{\Sigma}_0 = \mathbb{E}\left[(\mathbf{x}_0 - \hat{\mathbf{x}}_0).(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T\right]$;

$\hat{\mathbf{x}}_0^a = \mathbb{E}[\mathbf{x}_0] = \left[\hat{\mathbf{x}}_0^T, \boldsymbol{\theta}_0\right]^T$;

$$\boldsymbol{\Sigma}_0^a = \mathbb{E}\left[(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a).(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a)^T\right] = \begin{pmatrix} \boldsymbol{\Sigma}_0^{\mathbf{x}} & 0 \\ 0 & \boldsymbol{\Sigma}_0^{\boldsymbol{\theta}} \end{pmatrix};$$

foreach $t_k \in T$ **do**

$\mathcal{X}_{k-1}^a = \left[\hat{\mathbf{x}}_0^a, \hat{\mathbf{x}}_0^a \pm \sqrt{(\lambda + L)\boldsymbol{\Sigma}_{k-1}^a}\right]$; `// Calcul des sigma points`

`/* Étape de prédiction */`

$\mathcal{X}_{k|k-1}^x = f(\mathcal{X}_{k-1}^x, \mathbf{u}, t_k, \mathcal{X}_{k-1}^{\boldsymbol{\theta}})$;

$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} \omega_i^{(E)} \mathcal{X}_{i,k|k-1}^x$;

$\boldsymbol{\Sigma}_k^- = \sum_{i=0}^{2L} \omega_i^{(C)} \left[\mathcal{X}_{i,k|k-1}^x - \hat{\mathbf{x}}_k^-\right]\left[\mathcal{X}_{i,k|k-1}^x - \hat{\mathbf{x}}_k^-\right]^T$;

$\mathcal{Y}_{y|y-1} = h(\mathcal{X}_{k|k-1}^x)$;

$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} \omega_i^{(E)} \mathcal{Y}_{i,y|y-1}$;

`/* Étape de mise à jour */`

$\boldsymbol{\Sigma}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k} = \sum_{i=0}^{2L} \omega_i^{(C)} \left[\mathcal{Y}_{i,y|y-1} - \hat{\mathbf{y}}_k^-\right]\left[\mathcal{Y}_{i,y|y-1} - \hat{\mathbf{y}}_k^-\right]^T + R_k$;

$\boldsymbol{\Sigma}_{\mathbf{x}_k\mathbf{y}_k} = \sum_{i=0}^{2L} \omega_i^{(C)} \left[\mathcal{X}_{i,y|y-1} - \hat{\mathbf{x}}_k^-\right]\left[\mathcal{Y}_{i,y|y-1} - \hat{\mathbf{y}}_k^-\right]^T$;

$\boldsymbol{\mathcal{K}} = \boldsymbol{\Sigma}_{\mathbf{x}_k\mathbf{y}_k}\boldsymbol{\Sigma}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k}^-$; `// Calcul du gain de Kalman`

$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \boldsymbol{\mathcal{K}}(\hat{\mathbf{y}}_k - \hat{\mathbf{y}}_k^-)$;

$\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}_k^- - \boldsymbol{\mathcal{K}}\boldsymbol{\Sigma}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k}\boldsymbol{\mathcal{K}}^{\mathsf{T}}$;

end

2.2.2 Algorithmes d'optimisation

L'estimation des paramètres peut également être résolue par le biais d'un algorithme générique d'optimisation. Il suffit de définir une fonction objective appropriée, et d'appliquer

un algorithme d'optimisation adapté. Par exemple, en appliquant la statistique du $\mathcal{X}^2$ à la sortie du modèle à espace d'état donné (équation 2.4) sur l'ensemble de données $\mathbf{Y} = \mathbf{y}_{0:K}$ on peut construire la fonction objectif.

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}\in\mathbb{R}^d}{\operatorname{argmin}} \sum_{k=0}^{K} \left(h(\mathbf{x}_k(\boldsymbol{\theta}), t_k) - \mathbf{y}_k\right)^\mathsf{T} \boldsymbol{\Sigma}_k^{-}1 \left(h(\mathbf{x}_k(\boldsymbol{\theta}), t_k) - \mathbf{y}_k\right) \tag{2.14}$$

Nous allons présenter deux méthodes d'optimisation locale et globale adaptées à notre type de problème.

a. Nelder-Mead

L'algorithme Nelder-Mead est une heuristique d'optimisation locale non-linéaire, présentée pour la première fois par Nelder and Mead (1965). L'algorithme travaille à partir d'un simplexe sur lequel sont appliquées plusieurs transformations : *contraction*, *expansion* et *réduction* (voir l'exemple de la figure 2.2 pour $d = 2$).

Plus précisément, on peut décomposer l'algorithme en un certain nombre d'opérations décrites ci-dessous :

0. Étant donnée d la dimension de l'espace de recherche, former le simplexe constitué des $d+1$ points $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_{d+1}$.
1. Évaluer la fonction objectif en ces points, puis les réindexer de manière à vérifier $f(\boldsymbol{\theta}_1) \leq f(\boldsymbol{\theta}_2) \leq \cdots \leq f(\boldsymbol{\theta}_{d+1})$.
2. Calculer $\boldsymbol{\theta}_m$ le centre de gravité de tous les points excepté $\boldsymbol{\theta}_h = \boldsymbol{\theta}_{d+1}$.
3. Calculer et évaluer $\boldsymbol{\theta}_r = \boldsymbol{\theta}_m + \alpha(\boldsymbol{\theta}_m - \boldsymbol{\theta}_r)$ une *réflexion* de $\boldsymbol{\theta}_h$ par rapport à $\boldsymbol{\theta}_m$.
   - 3.a. Si $f(\boldsymbol{\theta}_r) < f(\boldsymbol{\theta}_d)$, calculer le point d'*expansion* $\boldsymbol{\theta}_e = \boldsymbol{\theta}_m + \gamma(\boldsymbol{\theta}_r - \boldsymbol{\theta}_m)$. Si $f(\boldsymbol{\theta}_e) < f(\boldsymbol{\theta}_r)$, $\boldsymbol{\theta}_h$ est remplacé par $\boldsymbol{\theta}_e$, sinon $\boldsymbol{\theta}_h$ est remplacer par $\boldsymbol{\theta}_r$. Retourner à l'étape 1.
   - 3.b. Si $f(\boldsymbol{\theta}_d) < f(\boldsymbol{\theta}_r)$, calculer le point de *contraction* $\boldsymbol{\theta}_c = \boldsymbol{\theta}_m + \beta(\boldsymbol{\theta}_{d+1} - \boldsymbol{\theta}_m)$. Si $f(\boldsymbol{\theta}_c) < f(\boldsymbol{\theta}_d)$, remplacer $\boldsymbol{\theta}_h$ par $\boldsymbol{\theta}_c$, et retourner à l'étape 1, sinon poursuivre avec l'étape 4.
4. Appliquer une *réduction* du simplexe en réalisant une homothétie de rapport $\frac{1}{2}$ et de centre $\boldsymbol{\theta}_1$ à tous les points autres que $\boldsymbol{\theta}_1$. Retourner à l'étape 1.

La figure 2.3 synthétise l'ensemble de ces opérations sous la forme d'un organigramme.

Nelder-Mead présente l'avantage d'être une méthode de type *derivative-free*, c'est-à-dire que l'information du gradient de la fonction objectif n'est pas nécessaire à la résolution. C'est une propriété intéressante pour les problèmes que nous côtoyons, dans lesquels le

gradient n'est pas nécessairement calculable ou s'avère trop complexe à calculer.

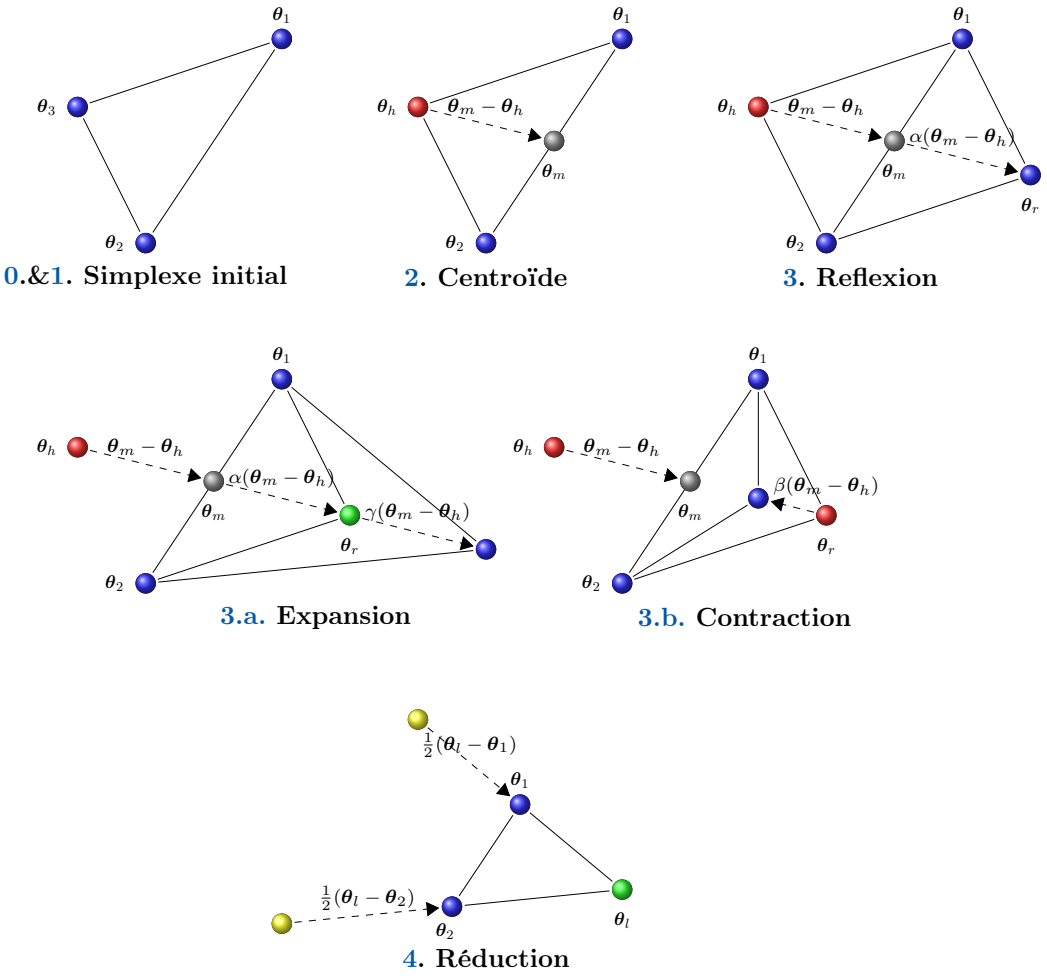

FIGURE 2.2 – Transformations appliquées par Nelder-Mead au simplexe. Les nœuds de couleur bleue représentent les points associés au simplexe courant. Le nœud gris représente le centroïde des points du simplexe excluant la pire solution. Les nœuds rouges et verts correspondent respectivement à la pire et à la meilleure solution courante. Les nœuds jaunes sont ceux affectés par l'homothétie

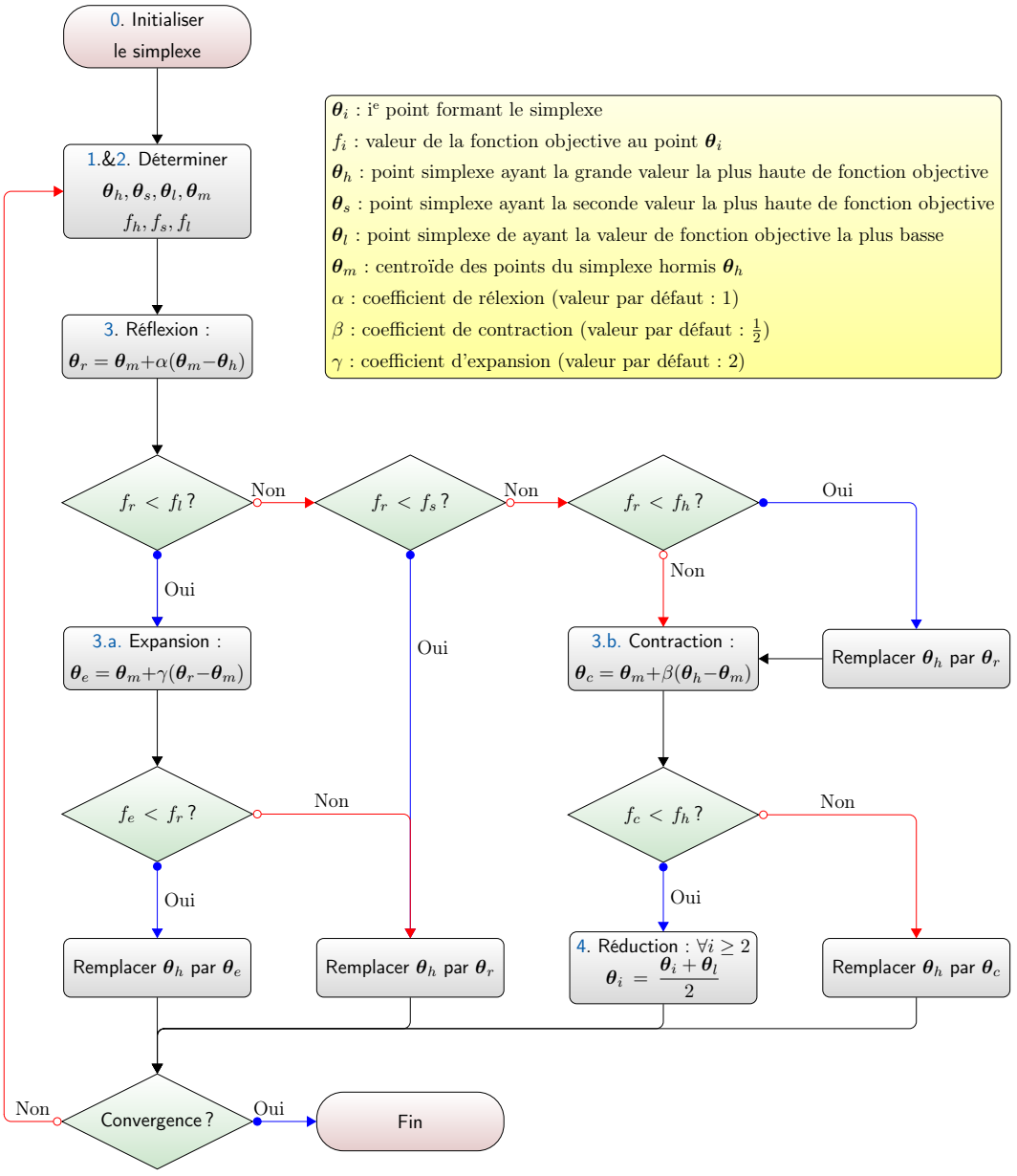

FIGURE 2.3 – Organigramme de l'algorithme Nelder-Mead (Luersen and Le Riche, 2004)

À lui seul, l'algorithme Nelder-Mead peut s'avérer inefficace sur des problèmes non-convexes de grande dimension, surtout en présence de nombreux *minima* locaux. Bien que Luersen and Le Riche (2004) aient proposé une approche intéressante visant à « globaliser » l'algorithme Nelder-Mead, nous avons privilégié un algorithme évolutionnaire appelé *recherche dispersée* (Glover, 1977), qui peut être combiné à un algorithme d'optimisation locale comme Nelder-Mead et qui obtient des performances remarquables sur un ensemble de *benchmarks* (Villaverde et al., 2015) portant sur des thématiques proches de celles que nous abordons.

b. Recherche dispersée

La recherche dispersée (Scatter Search, SS) est une méta-heuristique d'optimisation globale introduite par Glover (1977) dans le cadre de la programmation en nombres entiers (PLNE). L'algorithme original consiste à maintenir une population de bonnes solutions diversifiées (voir la figure 2.4), servant d'ensemble de référence pour guider les futures explorations. L'algorithme repose sur 3 opérations répétées jusqu'à ce qu'une condition d'arrêt soit satisfaite :

1. Construire une population initiale de vecteurs-solutions, et en extraire un ensemble de référence à partir des meilleurs éléments.
2. Générer de nouveaux points en combinant linéairement des sous-ensembles de solutions de l'ensemble de référence, de sorte que ces nouveaux points soient situés à l'intérieur et à l'extérieur des régions convexes délimitées par les points de référence.
3. Constituer une population à partir des meilleures solutions produites à l'étape 2, qui servira de nouveau point de départ à l'étape 1.

Bien qu'ayant été conçus à l'origine pour la PLNE, l'algorithme SS offre un cadre suffisamment générale pour permettre des adaptations pour d'autres applications (Laguna and Martí, 2005, Laguna and Marti, 2012). En ce qui nous concerne, Rodriguez-Fernandez et al. (2006) ont proposé SSm [2], un algorithme basé sur SS qui s'adresse directement au problème d'estimation de paramètres des systèmes dynamiques non-linéaire rencontrés en biologie des systèmes. À l'heure actuelle la version la plus aboutie est eSS [3] proposée par Egea et al. (2010) et qui obtient de meilleur résultats grâce à une méthode de combinaison des sous-ensembles s'appuyant sur la *recomposition de chemin* (Resende et al., 2010).

2. Scatter Search method
3. enhanced Scatter Search

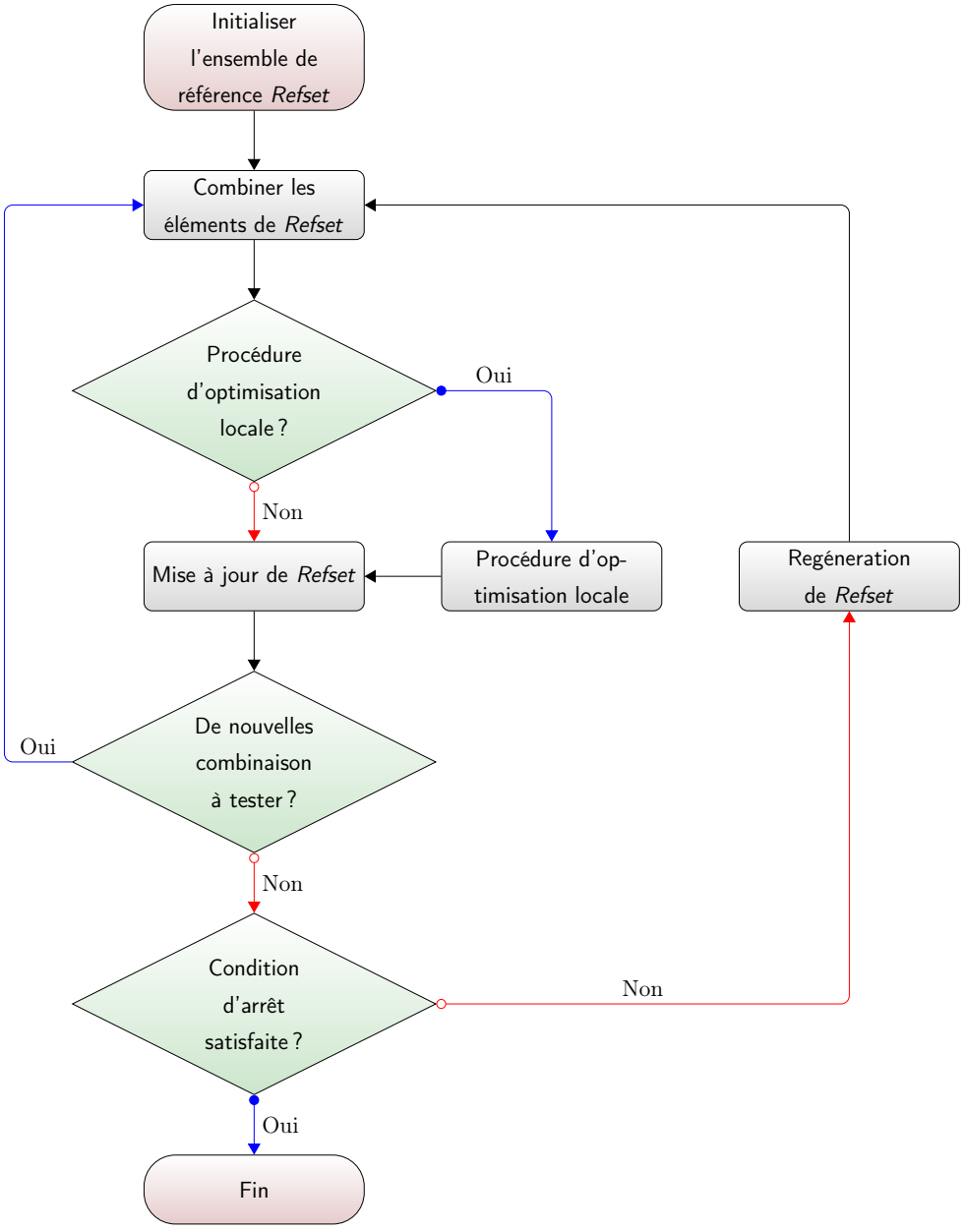

FIGURE 2.4 – Organigramme de l'algorithme de recherche dispersée (Rodriguez-Fernandez et al., 2006)

2.3 Introduction aux notions d'identifiabilité de modèle paramétrique

2.3.1 Identifiabilité structurelle

La problématique de l'identifiabilité des paramètres d'un modèle dont la structure est connue vise à déterminer, dans le cas théorique où l'on dispose d'une infinité d'observations non-bruitées du modèle, l'existence et l'unicité d'un vecteur de paramètres solution au problème d'identification. C'est donc un pré-requis nécessaire à l'identification de paramètres d'un modèle. Walter and Pronzato (1997) définissent un ensemble de notions permettant de caractériser l'identifiabilité des paramètres sont présentées. On admettra ici les définitions suivantes.
Étant donné f, un modèle de paramètre $\boldsymbol{\theta}$ qui caractérise la réponse du système à identifier.

Définition 2.1. Un paramètre $\boldsymbol{\theta}_i$ est dit *structurellement globalement identifiable* si pour presque tout $\boldsymbol{\theta}$, on a :

$$\forall \hat{\boldsymbol{\theta}} \in \Theta,\ f(\boldsymbol{\theta}) = f(\hat{\boldsymbol{\theta}}) \Longrightarrow \boldsymbol{\theta}_i = \hat{\boldsymbol{\theta}}_i$$

Définition 2.2. Un paramètre $\boldsymbol{\theta}_i$ est dit *structurellement localement identifiable* si pour presque tout $\boldsymbol{\theta}$, il existe un voisinage de $\boldsymbol{\theta}$ $v(\boldsymbol{\theta})$ tel que :

$$\forall \hat{\boldsymbol{\theta}} \in v(\boldsymbol{\theta}),\ f(\boldsymbol{\theta}) = f(\hat{\boldsymbol{\theta}}) \Longrightarrow \boldsymbol{\theta}_i = \hat{\boldsymbol{\theta}}_i$$

Définition 2.3. Un paramètre $\boldsymbol{\theta}_i$ est dit *structurellement non identifiable* si pour presque tout $\boldsymbol{\theta}$, il n'existe pas de voisinage de $\boldsymbol{\theta}$ $v(\boldsymbol{\theta})$ tel que l'on ait :

$$\forall \hat{\boldsymbol{\theta}} \in v(\boldsymbol{\theta}),\ f(\boldsymbol{\theta}) = f(\hat{\boldsymbol{\theta}}) \Longrightarrow \boldsymbol{\theta}_i = \hat{\boldsymbol{\theta}}_i$$

Finalement on dira qu'un modèle est identifiable si tous ses paramètres le sont, et *partiellement identifiable* si seule une partie des composantes du vecteur $\boldsymbol{\theta}$ est identifiable.

Il n'existe pas à notre connaissance de méthode générique permettant d'étudier l'identifiabilité structurelle de modèles dynamiques non linéaires : le choix de la méthode à employer dépend de la forme des équations du modèle. Les méthodes les plus classiquement utilisées s'appuient sur des développements en séries de Taylor des sorties, sur les séries génératrices, sur la méthode des similarités (*similarity transformation approach* (Vajda et al., 1989, Hengl et al., 2007)) ou encore sur des méthodes issues de l'algèbre différentielle. Celles-ci sont décrites et comparées par Chis et al. (2011).

2.3.2 Non-identifiabilité pratique

Lorsqu'il a été déterminé qu'un paramètre ou un ensemble de paramètres est structurellement identifiable (au moins localement), se pose alors la question de son identifiabilité pratique. En effet, l'identifiabilité structurelle suppose que les mesures sont parfaites : d'une résolution aussi fine que l'on veut, acquises pendant une durée potentiellement infinie et sans aucun bruit perturbateur. En réalité il n'en est bien sûr pas ainsi : un paramètre peut montrer une *non-identifiabilité pratique* causée par des considérations pratiques comme la présence de bruits d'observation ou un nombre très limité de points de données. Comme elle dépend de la quantité et de la qualité des données, une définition formelle de l'identifiabilité pratique est plus difficile à poser que dans le cas de l'identifiabilité structurelle qui ne dépend que des équations du modèle.

On adopte ici la définition proposée par Raue et al. (2009) à l'aide des intervalles de confiance et on présente sa méthode de détection des non-identifiabilités fondée sur la vraisemblance profilée, dont la figure 2.5 propose une illustration. On se place pour cela dans le cadre d'erreurs distribuées selon une loi gaussienne centrée : la vraisemblance (plus exactement son opposée) peut être assimilée à la fonction objectif définie par le critère des moindres carrés, dont un minimum est atteint en l'estimée ponctuelle $\hat{\boldsymbol{\theta}}$. Les intervalles de confiance sous l'hypothèse d'échantillons de petite taille (i.e. non asymptotiques), qui sont la norme en biologie, peuvent alors s'écrire :

$$\{\boldsymbol{\theta},\ \chi^2(\boldsymbol{\theta}) - \chi^2(\hat{\boldsymbol{\theta}}) < q^{\chi^2}(\alpha, df)\}$$

où $df = \#\boldsymbol{\theta}$ pour les intervalles de confiance joints (sur tous les paramètres simultanément) et $q^{\chi^2}(\alpha, df)$ est le quantile de la distribution du χ^2 à df degrés de libertés pour un niveau de risque α choisi. La méthode de Raue et al. (2009) s'appuie sur la vraisemblance profilée qui se calcule, pour chaque paramètre $\boldsymbol{\theta}_i$ indépendamment, par :

$$\chi^2_{PL}(\boldsymbol{\theta}_i) = \min_{\boldsymbol{\theta}_{-i}}(\chi^2(\boldsymbol{\theta}))$$

c'est-à-dire que pour chaque valeur de $\boldsymbol{\theta}_i$ fixée, on minimise le critère χ^2 selon toutes les autres composantes du vecteur paramètre. Cette fonction est utilisée pour explorer de manière efficace l'espace des paramètres à la recherche de zones de non-identifiabilités structurelles ou pratiques.

Les non-identifiabilités structurelles (figure 2.5 (a)&(b)) se manifestent par des paramétrisations redondantes, c'est-à-dire par des relations fonctionnelles entre des sous-ensembles de paramètres sur lesquelles la fonction objectif de l'estimation prend des valeurs constantes. Dans ce cas, l'intervalle de confiance d'un paramètre non-identifiable est non borné des deux côtés $[-\infty, \infty]$. En deux dimensions, cela signifie que la surface de réponse de la vraisemblance prend une forme de « vallée plate », s'étendant indéfiniment le long de la relation fonctionnelle correspondante. La vraisemblance profilée est également constante.

Inversement, un paramètre pratiquement identifiable sera caractérisé par un optimum (au moins local) de la vraisemblance ainsi que de la vraisemblance profilée. Son intervalle de confiance sera de la forme $[\sigma^-, \sigma^+]$ où σ^- et σ^+ prennent des valeurs finies (figure 2.5 (e)&(f)).

Enfin, un paramètre structurellement identifiable mais non pratiquement identifiable sera caractérisé par un intervalle de confiance non borné d'un côté au moins, c'est-à-dire de la forme $[\sigma^-, +\infty]$ ou $[-\infty, \sigma^+]$, bien que sa fonction de vraisemblance admette un unique optimum (figure 2.5 (c)&(d)). Le fait qu'on ait existence d'un unique optimum signifie que si l'on disposait d'observations parfaites, il nous serait possible d'identifier le paramètre. Ce serait possible également si l'on acceptait un niveau de risque très élevé ; mais ce n'est généralement pas pertinent, surtout si l'on considère que la non-identifiabilité pratique provient de données en nombre limité ou de mauvaise qualité : on aura alors plutôt tendance au contraire à augmenter le seuil $q^{\chi^2}(\alpha, df)$ ce qui renforce encore le problème de non-identifiabilité.

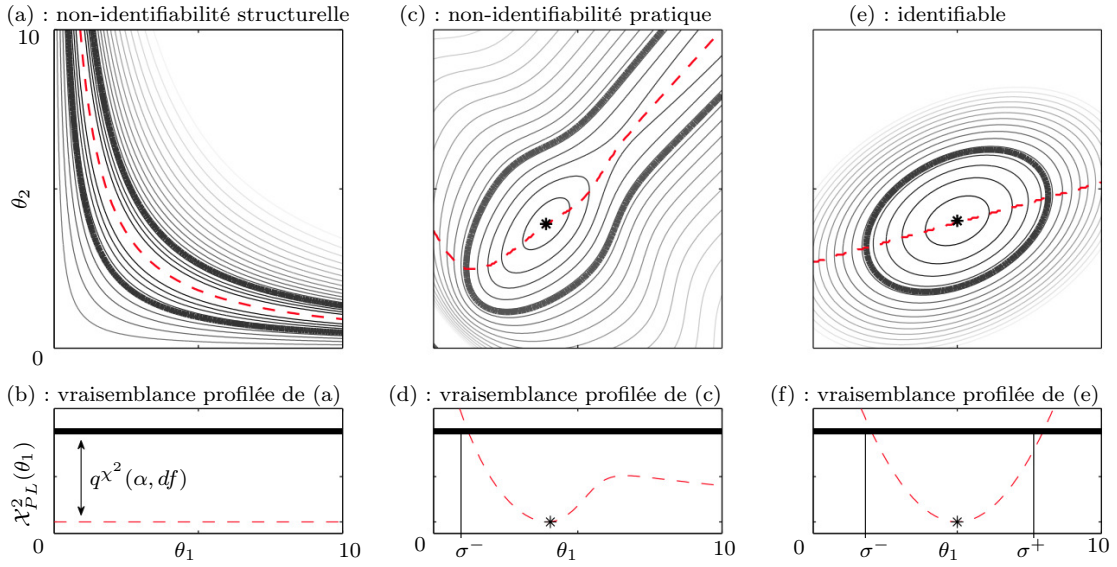

FIGURE 2.5 – Détection d'identifiabilités par la méthode de la vraisemblance profilée (Raue et al., 2011). (a), (c), (e) : courbes d'iso-vraisemblances dans un espace à 2 paramètres. (b), (d), (f) : vraisemblance profilée pour le paramètre $\boldsymbol{\theta}_1$ avec en trait gras le seuil permettant de construire l'intervalle de confiance associé à l'optimum lorsqu'il existe

2.3.3 Traitements possibles de ces problèmes d'identifiabilité

La non-identifiabilité structurelle est intrinsèquement liée à la structure du modèle (étant données les variables observées) : les solutions à adopter face à ce problème passent alors par une remise en question de la structure du modèle ou bien par sa re-paramétrisation (certaines méthodes, comme celle de Hengl et al. (2007), permettent d'identifier les sous-ensembles de paramètres qui sont simultanément identifiables), ou encore par le choix arbitraire de fixer certains paramètres à leurs valeurs nominales. On peut enfin également ajouter de nouvelles variables observables lorsque c'est possible.

En revanche, si un modèle a été prouvé être structurellement identifiable mais que des problèmes d'identifiabilité pratique se posent, on tentera généralement de les lever par l'acquisition de points de données supplémentaires bien choisis parmi les observables du modèles ou obtenus avec une meilleure résolution.

Dans chacun de ces deux cas, l'une des options prioritaires à envisager passe par l'acquisition de données supplémentaires, qu'il s'agit alors de choisir judicieusement afin qu'elles permettent de lever au mieux les problèmes d'identifiabilité détectés.

Chapitre 3

Apprentissage automatique

Sommaire

« *L'apprentissage n'épuise jamais l'esprit.* »

Leonardo da Vinci

Ce chapitre présente quelques notions d'apprentissage automatique. On commence par une introduction à l'apprentissage supervisé avant d'en venir à deux problématiques essentielles pour cette thèse, l'apprentissage actif et l'apprentissage par renforcement.

L'apprentissage automatique ou apprentissage artificiel (Cornuéjols and Miclet, 2011, Murphy, 2012, Mohri et al., 2012, Michalski et al., 2013) est un pan très important de l'intelligence artificielle. Dans sa définition première, cette discipline s'intéresse au développement de programmes capables d'augmenter leurs performances sur une tâche donnée en interagissant avec l'environnement. L'apprentissage hors ligne constitue le cadre le plus simple de l'apprentissage artificiel. Il s'agit d'estimer à partir d'un ensemble de données fourni avant apprentissage les paramètres d'un modèle. Lorsqu'il s'agit d'un modèle prédictif et que les données comprennent des exemples de paires entrées-sorties, on parle d'*apprentissage supervisé*. Lorsque les données d'entrée ne sont pas associées à des données de sortie, on parle d'*apprentissage non supervisé*. Dans l'apprentissage en ligne, l'environnement fournit régulièrement des données comme par exemple, dans le cas de l'observation d'une série temporelle. Par ailleurs, on distingue l'apprentissage actif de l'apprentissage passif. En apprentissage actif, l'algorithme d'apprentissage interagit avec l'environnement pour demander des données supplémentaires alors que l'apprentissage passif n'utilise que les données fournies. L'apprentissage par renforcement que nous introduisons dans 3.3 est la forme la plus générale d'apprentissage actif en ligne.

3.1 Apprentissage supervisé (« passif »)

Soient un espace d'entrée $\mathcal{X}$ et un espace de sortie $\mathcal{Y}$. Soit un échantillon aléatoire d'apprentissage, c'est-à dire comprenant des données i.i.d. [1] $\mathcal{L} = \langle (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \dots (\mathbf{x}_n, y_n) \rangle$ constituées de couples *vecteur d'attributs-sortie* telles que pour tout i, $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. Le but de l'apprentissage supervisé est de construire à l'aide des données $\mathcal{L}$, une fonction h appartenant à une famille de fonctions $\mathcal{H} = \{h : \mathcal{X} \mapsto \mathcal{Y}\}$ qui à tout $\mathbf{x} \in \mathcal{X}$ associe la sortie $y = h(\mathbf{x})$. On dit qu'un problème d'apprentissage est un problème de régression si l'espace de sortie est continu (typiquement $\mathcal{Y} \in \mathbb{R}^p$) et un problème de classification si l'espace de sortie est fini. Dans le cas de la classification supervisée, la fonction h est appelée *fonction de décision*. Pour évaluer les performances d'un tel *apprenant*, on définit une fonction de perte locale

$$\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$$

qui mesure pour tout triplet $(h, \mathbf{x}, y) \in \mathcal{H} \times \mathcal{X} \times \mathcal{Y}$ à quel point la prédiction $h(\mathbf{x})$ est éloignée de la vraie sortie y. Cette fonction de perte peut par exemple être une fonction de

1. indépendantes et identiquement distribuées

type « charnière » ($\ell(h, \mathbf{x}, y) = \max(1 - y \cdot h(\mathbf{x}), 0)$) classiquement utilisée en classification ou des moindres carrés en régression ($\ell(h, \mathbf{x}, y) = (y - h(\mathbf{x}))^2$).

Définition 3.1. Vrai risque ou erreur en généralisation
Étant donnée $P_{\mathcal{X}\mathcal{Y}}$ la loi de probabilité jointe sur $\mathcal{X} \times \mathcal{Y}$, le vrai risque $R(h)$ évalue la performance d'une fonction h par l'espérance de la fonction de perte associée :

$$R(h) = \mathbb{E}_{(X,Y)\sim P_{\mathcal{X}\mathcal{Y}}}\left[\ell(h(X), Y)\right] = \int_{\mathcal{X}\times\mathcal{Y}} \ell(h, \mathbf{x}, y) P_{\mathcal{X}\mathcal{Y}} d\mathbf{x} dy \tag{3.1}$$

Résoudre un problème d'apprentissage supervisé consiste à identifier la fonction $\hat{h}$ minimisant le vrai risque sur tout l'espace $\mathcal{H}$. Cependant, en pratique, la distribution $P_{\mathcal{X}\mathcal{Y}}$ est inconnue. Une première idée consiste à remplacer le vrai risque par le risque empirique en s'appuyant sur les exemples de l'ensemble d'apprentissage.

Définition 3.2. Risque Empirique
La performance d'un apprenant h sur un échantillon $\mathcal{L} = \langle(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \dots (\mathbf{x}_n, y_n)\rangle$ est donnée par le risque empirique :

$$\hat{R}(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h, \mathbf{x}_i, y_i)) \tag{3.2}$$

Cependant, la théorie statistique de l'apprentissage nous dit que le *principe de minimisation du risque empirique* n'est pertinent que si on dispose d'un très grand ensemble de données. Trop spécialiser un modèle sur un ensemble de données d'apprentissage réduit conduit au sur-apprentissage. Pour un nombre de données limité, on préfère la minimisation du risque empirique pénalisé par un terme de contrôle de la complexité du modèle, comme l'indique la formulation suivante :

$$\hat{h} = \arg\min_{h\in\mathcal{H}} \hat{R}(h) + \lambda\Omega(h),$$

où $\Omega : \mathcal{H} \to \mathbb{R}^+$ est une fonction de pénalité dépendant de h et reflétant sa complexité. Dans le cadre probabiliste des modèles à espace d'états, la minimisation de l'opposé de la log-vraisemblance pénalisée par un *a priori* correspond à cette formulation.

3.2 Apprentissage actif

Le cadre de l'apprentissage actif est un cas particulier d'apprentissage supervisé où l'apprenant a la possibilité d'influencer de manière plus ou moins importante le choix des

données d'apprentissage. Ce cadre est pertinent dans des situations où les données sont abondantes mais où le coût pour étiqueter l'information sur ces données est élevé. Dans ces conditions, l'apprenant a plutôt intérêt à sélectionner les données les plus informatives, ce qui rend l'ensemble d'apprentissage non i.i.d. L'apprentissage actif est connu pour fournir de meilleures performances que l'apprentissage passif, c'est-à-dire qu'à performances en généralisation égales, l'apprentissage actif nécessite moins d'exemples que l'apprentissage passif, et réciproquement (Cohn et al., 1994). La thèse de Steve Hanneke (Hanneke, 2009) étudie la vitesse de convergence, en terme d'erreur, de quelques algorithmes d'apprentissage actif. Steve Hanneke démontre notamment que sous certaines conditions de bruit bien défini, l'apprentissage actif améliore systématiquement sur la vitesse de convergence asymptotique par rapport à un apprentissage passif.

3.2.1 Exemple jouet

Afin l'illustrer le bénéfice potentiel de l'apprentissage actif par rapport à l'apprentissage passif, on peut considérer le problème jouet suivant. Supposons que l'on veuille apprendre une fonction de décision qui prédit si un réel x est membre de l'intervalle $[w, \infty[$, où $w \in \mathbb{R}$ est une certaine valeur seuil (figure 3.1). L'espace des hypothèses est représenté par l'ensemble des fonctions indicatrices $\mathcal{H} = \{h_w(x) = \delta_w(x) \mid w \in \mathbb{R}\}$. Un oracle est en mesure de déterminer sans commettre d'erreur si un exemple x appartient à la classe *positive* ($x \geqslant w$) ou *négative* ($x < w$).

$$h_w(x) = \delta_w(x) = \begin{cases} 1 \text{ si } x \geqslant w \\ 0 \text{ si } x < w \end{cases}$$

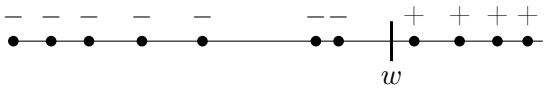

FIGURE 3.1 – Apprentissage d'un seuil sur la droite réelle (Cornuéjols and Miclet, 2011)

Une façon d'aborder ce problème du point de vue de l'apprentissage passif consiste à tirer aléatoirement selon une loi uniforme dans un intervalle réel donné (*a priori*). Ensuite on soumet à l'oracle l'ensemble de ces valeurs pour obtenir leur étiquette (+ ou −). En nommant w^- le plus grand exemple négatif et w^+ le plus petit exemple positif de l'ensemble d'apprentissage, la stratégie passive consiste à choisir comme seuil le milieu de ces deux exemples $\hat{w} = \frac{w^- + w^+}{2}$. D'après la théorie d'apprentissage PAC [2] introduite par Valiant (1984), s'il existe une hypothèse θ permettant de classer parfaitement n'importe quel échantillon issu de la distribution de données sous-jacente, alors il suffit d'un échantillon aléatoire de taille $\mathcal{O}(\frac{1}{\epsilon})$ pour obtenir un taux d'erreur d'au plus ϵ.

2. Probably Approximately Correct

À présent si on aborde ce problème par un apprentissage actif, la démarche se fait de manière séquentielle. Au lieu de demander à l'oracle toutes les étiquettes à la fois, on formule des requêtes une à une en tirant sur l'intervalle $]w^-, w^+[$ courant. Dans ce cas, la taille de l'échantillon nécessaire pour obtenir un taux erreur d'apprentissage d'au plus ϵ devient $\mathcal{O}\left(\log(\frac{1}{\epsilon})\right)$.

3.2.2 Les approches constructives et sélectives

On distingue deux types d'approches selon le type de formulation des requêtes : les *approches constructives* et les *approches sélectives*.

Dans une approche constructive, l'apprenant construit lui-même les exemples d'apprentissage au lieu de les choisir dans un ensemble prédéfini comme dans l'approche sélective. Pour construire ces exemples, il demande à l'oracle d'étiqueter n'importe quel attribut $x \in \mathcal{X}$ qui peut lui sembler utile à l'amélioration de ses performances. Cela peut poser des problèmes lorsque les attributs doivent respecter certaines contraintes pour pouvoir être étiquetés : on risque alors de construire des exemples non réalisables en pratique ou impossibles à mesurer par un appareil de mesure ou encore difficiles à interpréter par un expert humain, comme le démontrent Baum and Lang (1992) dans le cadre de la reconnaissance manuscrite par réseaux de neurones. Malgré ces inconvénients, cette approche est très prometteuse comme le montre l'application du robot scientifique ADAM (King et al., 2004, 2009a,b) qui emploie ce type d'approche pour planifier des expérimentations biologiques de manière autonome. Pour les raisons évoquées ci-dessus, on va néanmoins s'intéresser dans la suite exclusivement à la seconde approche, c'est-à-dire l'approche sélective.

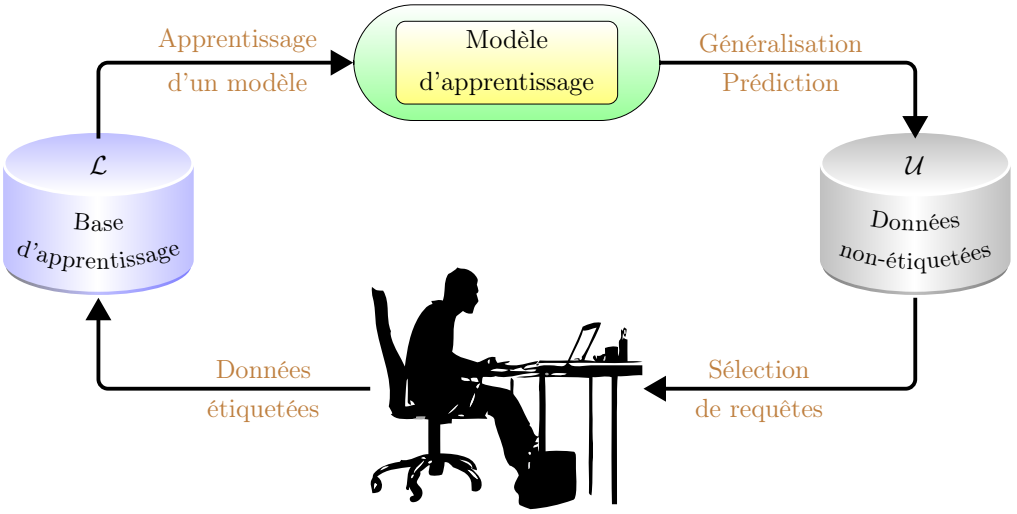

FIGURE 3.2 – Approche sélective en apprentissage actif selon le protocole *pool-based* (Settles, 2010)

Dans une approche sélective, les exemples sont choisis parmi un échantillon non étiqueté, soit disponible sous la forme d'une base de données (*pool-based*, figure 3.2), soit sous la forme d'un flux de données (*stream-based*). La méthodologie générale pour la sélection de requête consiste à demander l'étiquette des exemples pouvant apporter une information « utile », pour une définition d'un critère d'utilité approprié. Cornuéjols and Miclet (2011) présente un algorithme générique d'apprentissage actif par sélection (voir algorithme 2).

Algorithm 2: Algorithme générique d'échantillonnage actif (Muslea, 2002)

Data:

- Donnée étiquetées : $\mathcal{L} \subset \mathcal{X}$;
- Donnée non-étiquetées : $\mathcal{U} \subset \mathcal{X}$;

Input:

- Taille maximale d'échantillon : n ;
- Fonction $U : \mathcal{X} \times \mathcal{H} \to \mathbb{R}$, évaluant l'utilité d'un exemple pour l'apprentissage d'une fonction de décision ;

while $|\mathcal{L}| \leq n$ **do**

A. Apprentissage supervisé : explorer $\mathcal{H}$ à l'aide de $\mathcal{L}$ (voire $\mathcal{U}$)

B. Rechercher l'exemple $\mathbf{x}^* = \text{argmax}_{\mathbf{x} \in \mathcal{X}} U(x, \mathcal{H})$

C. Retirer l'exemple $\mathbf{x}^*$ de $\mathcal{U}$ et demander l'étiquette $f(\mathbf{x}^*)$ à l'oracle

D. Ajouter l'exemple $(\mathbf{x}^*, f(\mathbf{x}^*))$ à la base d'apprentissage : $\mathcal{L} \leftarrow \mathcal{L} \cup (\mathbf{x}^*, f(\mathbf{x}^*))$

end

3.2.3 Méthodes d'échantillonnage sélectif

On présente ici une sélection des principales méthodes d'échantillonage sélectif, la plupart tirée des ouvrages (Settles, 2010, 2012) qui présentent un état de l'art assez riche sur le domaine de l'apprentissage actif. Bien que les travaux présentés dans la suite s'adressent principalement au problème de classification, les notions qu'ils impliquent peuvent parfaitement s'étendre à la régression. Ces travaux sont divisés en trois catégories : la réduction d'espace des versions, la réduction d'incertitude et l'approche par comité.

a. Réduction d'espace des versions.

La notion d'*espace des versions* a été introduite par Tom Mitchell (Mitchell, 1982) dans le cadre de l'apprentissage de supervisée de concept ($\mathcal{Y} = \{0, 1\}$) où l'on cherche à apprendre une fonction de décision qui détermine si un exemple est membre ou non d'un concept.

Définition 3.3. Espace des versions.
L'espace des versions est le sous-ensemble $\mathcal{V}(\mathcal{S}) \subseteq \mathcal{H}$ des hypothèses cohérentes par rapport aux données d'apprentissage.

$$\mathcal{V}(\mathcal{S}) = \{h \in \mathcal{H} \ / \ h \text{ est cohérent avec } \mathcal{S}\}.$$

Typiquement en classification, en supposant qu'aucune erreur n'est possible, l'espace des versions pourra s'écrire $\mathcal{V}(\mathcal{S}) = \{h \in \mathcal{H} \ / \ \forall (x, y) \in \mathcal{S}^2, h(x_i) = y_i\}$.
L'idée de l'apprentissage actif par réduction d'espace des versions est de sélectionner les exemples permettant de réduire le plus la « taille » de l'espace des versions. Dans la cadre bayésien, en supposant l'existence d'une loi d'une loi de probabilité sur l'espace des hypothèses $P_{\mathcal{H}}$, la probabilité $P_{\mathcal{H}}(\mathcal{V}(\mathcal{S}))$ peut être employée pour représenter la taille de l'espace des versions.

b. Réduction d'incertitude.

L'*échantillonnage selon l'incertitude* étudié par Lewis and Catlett (1994) consiste à sélectionner des exemples pour lesquels le modèle a le moins confiance en ses prédictions. Cela nécessite un modèle qui, en plus de prédire, fournit une estimation du niveau de confiance de sa prédiction. Culotta and McCallum (2005) proposent d'utiliser un critère de *moindre confiance* (*least confidence*) obtenue à l'aide de l'algorithme CRF (*constrained forward-backward*).

$$x^*_{LC} = \underset{x \in \mathcal{U}}{\operatorname{argmin}}\, 1 - P(y|x; \mathcal{L}) \tag{3.3}$$

Une autre approche proposée consiste à échantillonner dans la *région d'incertitude* (Cohn et al., 1994), qui un sous-ensemble de l'espace des versions dans lequel les prédictions sont incertaines.

Définition 3.4. Région d'incertitude.
Soit $\mathcal{L} \subset \mathcal{X}$ un échantillon d'apprentissage. La région d'incertitude $\mathcal{I}_{\mathcal{L}}$ associée à l'échantillon $\mathcal{L}$ est l'ensemble des exemples d'apprentissage $x \in \mathcal{X}$ pour lequel il existe deux hypothèses contradictoires dans l'espace des versions $\mathcal{V}$.

$$\mathcal{I}_{\mathcal{L}} = \{x \in \mathcal{X} \ / \ \exists (h_1, h_2) \in \mathcal{V}(\mathcal{L})^2, h_1(x) \neq h_2(x)\}$$

Cependant, représenter et maintenir cet espace de manière fidèle s'avère le plus souvent irréaliste du point de vue calculatoire, c'est pourquoi les auteurs Seung et al. (1992), Cohn et al. (1994) et Dasgupta et al. (2007) présentent différentes approches visant à simplifier la mise à jour de cette région d'incertitude. Par exemple, Cohn et al. (1994) proposent de remplacer $\mathcal{I}_{\mathcal{L}}$ par un sous-ensemble $\mathcal{I}_{\mathcal{L}}^{-}$ ou par un sur-ensemble $\mathcal{I}_{\mathcal{L}}^{+}$ le plus fidèle possible de l'ensemble original tout en étant simple à construire.

c. Approche par comité.
L'algorithme **QBC** (**Q**uery-**b**y-**c**ommiteed) introduit par Seung et al. (1992) consiste à maintenir un comité d'hypothèses candidates entraînées à partir d'une même base d'apprentissage, pour ensuite les mettre en compétition. Les exemples candidats sont soumis au comité, chaque modèle votant en étiquetant l'exemple selon ses prédictions. L'exemple d'apprentissage le plus favorable à une requête est alors celui qui met le plus en désaccord le comité.
L'application de l'algorithme QBC nécessite de réunir deux conditions :

(i) Avant tout il faut construire un comité représentatif de l'espace des versions.

(ii) Puis établir une mesure du niveau désaccord des membres du comité.

Pour la première tâche, Seung et al. (1992) constituent un comité de deux hypothèses tirées uniformément dans l'espace des versions. Une approche à un seul modèle est employée par Lewis and Catlett (1994) pour des modèles utilisant une valeur seuil pour déterminer la classe d'un exemple. L'idée est alors de choisir les exemples proches de la frontière de décision du modèle. Néanmoins cette approche est très critiquée par Cohn et al. (1994), du fait qu'un modèle unique a peu de chance d'être représentatif de l'espace des versions. Abe and Hiroshi (1998) constituent les comités à l'aide de méthodes d'ensemble (*boosting* (Freund et al., 1997), *bagging* (Breiman, 1996)). Il n'y a pas dans la littérature de consensus sur la taille adéquate du comité car cela dépend du problème ou de la famille de modèles choisie (Settles, 2010).

Pour la seconde tâche, Dagan and Engelson (1995) proposent d'utiliser le *vote par entropie* pour mesurer le niveau de désaccord du comité. On considère un problème multi-classes. Étant donné un comité de modèles $\mathcal{C} = \{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(C)}\}$:

$$x^*_{VE} = \underset{x \in \mathcal{U}}{\operatorname{argmax}} - \sum_i \frac{\nu(x, y_i)}{|\mathcal{C}|} \cdot \log \left(\frac{\nu(x, y_i)}{|\mathcal{C}|} \right), \tag{3.4}$$

où y_i est l'étiquette de i-ème classe, et la quantité $\nu(x, y_i)$ représente le nombre total de votes du comité attribuant la classe y_i à l'exemple x.
Comme autre mesure possible du désaccord du comité, McCallum and Nigamy (1998) proposent d'utiliser la moyenne de la *divergence de Kullback-Lieber*.

$$x^*_{KL} = \underset{x \in \mathcal{U}}{\operatorname{argmax}} \frac{1}{|\mathcal{C}|} \sum_{\boldsymbol{\theta} \in \mathcal{C}} \sum_i P_{\boldsymbol{\theta}}(y_i|x) \cdot \log \left(\frac{P_{\boldsymbol{\theta}}(y_i|x)}{P_{\mathcal{C}}(y_i|x)} \right), \tag{3.5}$$

où $P_{\mathcal{C}}(y_i|x)$ est définie comme la distribution de probabilité « consensus » de l'appartenance de l'exemple x à la classe y_i.

$$P_{\mathcal{C}}(y_i|x) = \frac{1}{|\mathcal{C}|} \sum_{\boldsymbol{\theta} \in \mathcal{C}} P_{\boldsymbol{\theta}}(y_i|x) \tag{3.6}$$

3.3 Apprentissage par renforcement

Dans le cadre de l'apprentissage par renforcement, l'apprenant ne se contente plus de recevoir des données, mais il interagit avec son environnement à travers un ensemble d'actions (voir figure 3.3) pour acquérir de l'expérience. L'apprenant est représenté par un agent, possédant un état qui détermine les actions possibles, celles-ci entraînant un changement d'état et l'attribution d'une récompense (positive ou négative).

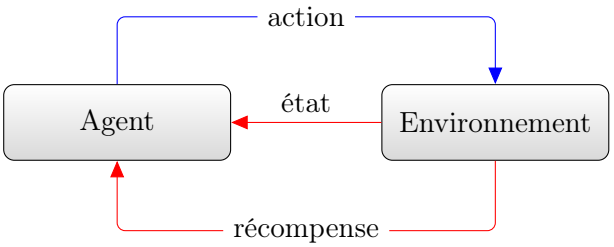

FIGURE 3.3 – Scénario général du problème d'apprentissage par renforcement (Mohri et al., 2012).

L'objectif est de construire une stratégie visant à déterminer l'action à réaliser en fonction de l'état de l'agent, dans le but de maximiser le cumul des récompenses au cours du temps.

3.3.1 Processus de décision de Markov

Les processus de décision de Markov ou MDP [3], offrent un cadre probabiliste pour formaliser les problèmes de décision séquentielle. Un MDP modélise le scénario dans lequel un agent, connaissant son état, prend des décisions dans un environnement dans lequel le résultat de ses actions est incertain, ce qui en fait un outil de référence pour traiter des problèmes d'apprentissage par renforcement (figure 3.3). Le MDP est introduit pour la première fois par Howard (1960), avec l'idée de combiner le modèle des chaînes de Markov et les outils de programmation dynamique s'appuyant sur les travaux de Bellman (1952, 1957). Comme son nom l'indique, un MDP utilise une chaîne de Markov pour représenter la succession d'états dans lequel l'agent peut se trouver. On fait donc l'hypothèse que l'état du processus à l'instant t, dépend uniquement de son état à l'instant $t-1$.

Plus formellement, un MDP sera décrit par un quadruplet $\mu = \{\mathcal{S}, \mathcal{A}, T, R\}$, à savoir :

- Un espace d'état $\mathcal{S}$.
- Un espace d'action $\mathcal{A}$.

3. en anglais : Markov Decision Process

- Une fonction de densité de probabilité de transition.

$$T : \begin{cases} \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1] \\ (s,a,s') \mapsto p(s'|s,a) \end{cases} \tag{3.7}$$

  telle que $\int_{s' \in \mathcal{S}} T(s,a,s')ds' = 1$
- Une fonction de récompense.

$$R : \begin{cases} \mathcal{S} \times \mathcal{A} \to \mathbb{R} \\ (s,a) \mapsto R(s,a) \end{cases} \tag{3.8}$$

La figure (3.4) illustre le fonctionnement d'un MDP par un exemple avec un nombre fini d'états et de transitions. L'espace d'état étant $\mathcal{S} = \{s_1, s_2, s_3\}$ et l'espace des actions étant $\mathcal{A} = \{a_1, a_2\}$, on peut résumer l'ensemble des transitions à l'aide de deux matrices de transition T_1 et T_2 associées respectivement aux actions a_1 et a_2.

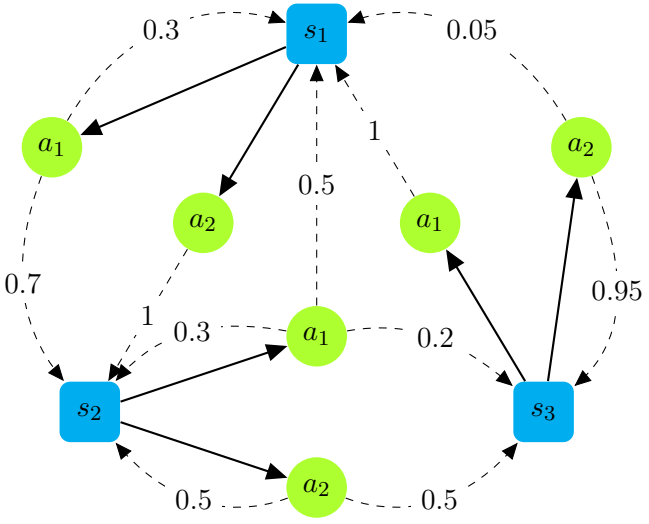

FIGURE 3.4 – Schéma d'un processus de décision Markovien. Les flèches en pointillés représentent les transitions possibles, les carrés bleus sont les états et les cercles verts des actions. Les étiquettes sur chaque flèche expriment les probabilités de transition.

$$T_1 = \begin{pmatrix} 0.3 & 0.7 & 0 \\ 0.5 & 0.3 & 0.2 \\ 1 & 0 & 0 \end{pmatrix}, \; T_2 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0.5 & 0.5 \\ 0.05 & 0 & 0.95 \end{pmatrix} \tag{3.9}$$

Cette forme de représentation n'est possible que lorsque les nombres d'états et de transitions sont finis.

Définition 3.5. Stratégie déterministe

Une stratégie déterministe est une fonction qui, à un état donné, associe une action.

$$\pi : \mathcal{S} \mapsto \mathcal{A}$$

La stratégie π détermine donc le comportement de l'agent. Lorsque celui-ci se trouve dans l'état s, il est amené à réaliser l'action $a = \pi(s)$. Une stratégie optimale vise à maximiser le cumul pondéré des récompenses aléatoires sur un horizon infini.

$$\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \tag{3.10}$$

Si la récompense est bornée, la constante $\gamma \in [0, 1[$ assure la convergence de la série en accordant un poids plus faible aux récompenses obtenues plus tard ($\gamma^t \xrightarrow[t \to +\infty]{} 0$). Pour le cas d'un horizon fini, γ vaut tout simplement 1.
La grandeur (3.10) étant aléatoire, on peut considérer son espérance pour une stratégie et un état initial donnés.

Définition 3.6. Fonction de valeur

On appelle fonction de valeur, l'espérance du cumul pondéré des récompenses obtenues pour une stratégie π et l'état initial s_0 donnés.

$$V_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \middle| s_0 = s; \pi \right] \tag{3.11}$$

Proposition 3.7. Équation de Bellman

La fonction de valeur $V_\pi(s)$ associée à une stratégie π et à l'état $s \in \mathcal{S}$ satisfait l'équation suivante :

$$\forall s \in \mathcal{S}, V_\pi(s) = R(s, \pi(s)) + \gamma \sum_{s'} P(s'|s, \pi(s)) V_\pi(s'). \tag{3.12}$$

Démonstration. La preuve consiste dans un premier temps à décomposer la somme de la fonction de valeur (a). En opérant un changement de variable sur l'indice de la somme ($\tau = t - 1$), on voit réapparaître la fonction de valeur (b). Ensuite, on utilise le théorème de l'espérance totale (c).

$$
\begin{aligned}
V_\pi(s) &= \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,\pi(s_t))\middle| s_0=s;\pi\right] \\
&\overset{(a)}{=} R(s,\pi(s)) + \mathbb{E}\left[\sum_{t=1}^{\infty}\gamma^t R(s_t,\pi(s_t))\middle| s_0=s;\pi\right] \\
&\overset{(b)}{=} R(s,\pi(s)) + \gamma\mathbb{E}\left[\sum_{\tau=0}^{\infty}\gamma^\tau R(s_\tau,\pi(s_\tau))\middle| s_0=s;\pi\right] \\
&\overset{(c)}{=} R(s,\pi(s)) + \gamma\sum_{s'}P(s'|s,\pi(s))\mathbb{E}\left[\sum_{\tau=0}^{\infty}\gamma^\tau R(s_\tau,\pi(s_\tau))\middle| s_0=s';\pi\right] \\
&= R(s,\pi(s)) + \gamma\sum_{s'}P(s'|s,\pi(s))V_\pi(s').
\end{aligned}
\tag{3.13}
$$

□

Définition 3.8. Stratégie optimale

La solution d'un MDP parmi un ensemble donné de stratégies Π est une *stratégie optimale* π^* telle que :

$$\pi^* \in \underset{\pi\in\Pi, s\in\mathcal{S}}{\operatorname{argmax}} V_\pi(s)$$

On appelle alors *valeur de fonction optimale* $V^* = V^{\pi^*}$, la valeur de fonction associée à cette politique optimale.

3.3.2 Problème bandit

a. Description du problème bandit à K bras

Un bandit multi-bras peut être vu comme un cas particulier de MDP à un seul état non affecté par les actions, avec une récompense stochastique (Feldman and Domshlak, 2014). Le problème bandit, introduit par Robbins (1985), modélise la situation d'un agent confronté à K bras (actions) possibles qu'il doit choisir d'actionner. À chaque instant t, l'agent choisit une action k ($\in [1, K]$) et reçoit alors une récompense $x_{k,t}$ issue d'une distribution ν_k de loi inconnue supposée fixée mais dépendant de l'action choisie. L'objectif est d'accumuler la plus grande somme de récompenses au bout d'une certaine durée donnée. L'agent est alors confronté au compromis *exploration-exploitation*. En effet, pour maximiser le cumul des récompenses, il a intérêt à répéter la meilleure action obtenue jusqu'alors (*exploitation*) ; cependant pour déterminer la meilleure action, il est nécessaire d'essayer d'autres actions qui peuvent s'avérer sous-optimales (*exploration*). Ces étapes d'exploration sont d'autant plus importantes que, comme la récompense attribuée après une action donnée est tirée dans une distribution de probabilité, plusieurs itérations sont nécessaires pour en évaluer l'intérêt réel en termes d'espérance de récompense. Dans les problèmes bandits,

ainsi que dans de nombreux algorithmes d'optimisation dits heuristiques (algorithmes génétiques (Golberg, 1989), optimisation par essaims de particules (Eberhart and Kennedy, 1995), méthode du recuit simulé (Kirkpatrick et al., 1983), etc.), on va généralement privilégier l'exploration en début de recherche puis, lorsque suffisamment d'information a été recueillie, progressivement donner la priorité à l'exploitation.

a..1 Le regret cumulé. Pour décider quelle action choisir à chaque instant t, l'agent emploie une stratégie π. Notons μ^* l'espérance de la récompense associée à la meilleure action, et x_t la récompense reçue à l'instant t indépendamment du bras sélectionné. Afin d'évaluer la performance d'une stratégie donnée, on introduit la notion de **regret cumulé** après n tirages, définie par la relation :

$$R_n = n \cdot \mu^* - \sum_{t=1}^{n} x_t \tag{3.14}$$

Auer et al. (2002) proposent un ensemble de stratégies dites de type UCB (Upper Confident Bounds) garantissant que $\mathbb{E}[R_n]$, le regret cumulé moyen, ait un comportement logarithmique en n. L'algorithme 3 montre la résolution d'un problème bandit à l'aide de la stratégie UCB1, la plus connue des stratégies de type UCB.

Algorithm 3: Stratégie déterministe UCB1 (Auer et al., 2002)

Input: Nombre de tirages : T ; K bras : $\{1, \ldots, K\}$

`/* Initialisation */`

foreach $k \in \{1, \ldots, K\}$ **do**

  Jouer le bras k et recevoir la récompense $x_{k,0}$;

  $\bar{\mu}_{k,0} \leftarrow x_{k,0}$;

  $n_{k,0} \leftarrow 1$;

end

for $t = 1 \ldots T$ **do**

  $I \leftarrow \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \bar{\mu}_{k,t} + \sqrt{\frac{2 \cdot \log(t)}{n_{k,t}}}$; `// Sélection du bras selon le critère UCB`

  $x_{I,t} \leftarrow$ jouer le bras I; `// Récupération de la récompense`

  $\bar{\mu}_{I,t} \leftarrow \frac{(n_{I,t-1}) \cdot \bar{\mu}_{I,t-1} + x_{I,t}}{n_{I,t-1} + 1}$; `// Mise à jour de la récompense moyenne`

  $n_{I,t} \leftarrow n_{I,t-1} + 1$; `// Mise à jour du nombre de tirage`

end

À chaque instant t, UCB1 choisit le bras I à jouer en résolvant :

$$I = \underset{k \in \{1,\dots,K\}}{\operatorname{argmax}} \bar{\mu}_{k,t} + \sqrt{\frac{2 \cdot \log(t)}{n_{k,t}}}$$

où le terme $n_{k,t}$ représente le nombre fois que le bras k a été sélectionné à l'instant t et le terme $\bar{\mu}_{k,t} = \frac{1}{n_{k,t}} \sum_{\tau=1}^{n_{k,t}} x_{k,\tau}$ est la moyenne empirique de la récompense associée à l'action k à l'instant t. Le terme $\bar{\mu}_{k,t}$ a tendance à favoriser l'*exploitation* des bras les plus prometteurs, alors que le terme $\sqrt{\frac{2 \cdot \log(t)}{n_{k,t}}}$, proportionnel à l'inverse de la racine du nombre de sélections du bras, favorise l'*exploration* des bras les moins souvent sélectionnés.

b. Une variante, le problème d'exploration pure

Bien que le problème bandit soit suffisamment général pour couvrir un large champ d'applications, il existe une autre catégorie de problèmes assez proche pour laquelle la formulation classique du problème n'est pas adaptée. Il s'agit des problèmes de type *exploration pure* (Bubeck et al., 2009, 2011). Contrairement au cadre classique des problèmes bandits où les phases d'exploration et d'exploitation s'entremêlent, elles sont bien distinctes dans un problème d'exploration pure. Un scénario typique du problème bandit classique est celui des tests cliniques. Dans ce cas, à chaque essai l'agent réalise un compromis entre deux actions : *soigner le plus grand nombre de patients (exploitation)* ou *trouver un(le) meilleur traitement (exploration)*. Considérons à présent le scénario présenté par Bubeck et al. (2009) d'une entreprise de produits cosmétiques qui réalise une phase de test sur ses produits, avant la mise sur le marché. Dans ce scénario la phase de test correspond à une phase d'exploration « pure », alors que la décision du produit final à mettre sur le marché constitue la phase d'exploitation.

b..1 Les stratégies de recommandation. Pour formuler le problème d'exploration pure, on introduit la notion de stratégie de recommandation ψ. Bubeck et al. (2009) présentent trois types de fonctions de recommandation éligibles.

- **EBA - Empirical best arm**
  Cette stratégie de recommandation vise à sélectionner le bras associé à la meilleure moyenne empirique des récompenses. Cette stratégie peut sembler judicieuse ; elle est néanmoins très risquée, le risque étant de choisir un bras sous-optimal mais ayant été trop peu visité pour s'en rendre compte. L'EBA est une méthode de recommandation particulièrement appropriée lorsqu'elle est combinée avec une stratégie de sélection uniforme.

- **MPA - Most Played Arm**
  On recommande le bras joué le plus souvent. Cette stratégie est plus prudente que la dernière. Elle est bien adaptée lorsqu'elle est combinée avec une stratégie d'exploration de type UCB qui vise à jouer plus fréquemment les bras les plus prometteurs.

- **Empirical distribution of plays**
  On choisit un bras aléatoirement selon une distribution de probabilité construite à partir des moyennes empiriques des récompenses de chaque bras.

b..2 Le simple regret. Le notion de regret cumulé n'est pas adaptée à l'évaluation des performances des stratégies (exploration et recommandation) pour les problèmes d'exploration pure, car seule la recommandation finale compte. Pour pallier à ce problème, Bubeck et al. (2009) introduisent la notion de **simple regret**. À un instant t, le simple regret est défini comme la différence entre l'espérance du meilleur bras connu μ^* et l'espérance du bras recommandé $\mu_{\psi(t)}$.

$$r_t = \mu^* - \mu_{\psi(t)} \tag{3.15}$$

Bubeck et al. (2009) établissent un lien entre les concepts de regret cumulé et celui de simple regret, qui se résume par la formule suivante :« Plus le regret cumulé est petit, plus le simple regret est grand ». En effet, un regret cumulé petit implique qu'on a consacré plus de ressources à l'exploitation, et donc qu'on a négligé l'exploration ; en conséquence, le simple regret est plus grand car on n'est pas mesure de fournir la meilleure recommandation.

3.3.3 Arbres de recherche de Monte-Carlo

Comme le mentionnent Feldman and Domshlak (2014), lorsque un MDP est donné de manière explicite, la résolution exacte par programmation dynamique est applicable ; en revanche si celui-ci est donné de manière générative par le bais de simulateurs, les arbres de recherche de Monte-Carlo (MCTS) sont plus appropriés pour la résolution du MDP. MCTS est un algorithme permettant d'approcher efficacement les solutions optimales d'un problème de décision dans un environnement incertain. MCTS a notamment acquis une grande notoriété dans la résolution du jeu de Go par ordinateur (Brügmann, 1993, Gelly

and Silver, 2007, Silver et al., 2016), en montrant des performances honorables là où les approches traditionnelles de recherche arborescente (A^* (Hart et al., 1968), *Minimax* ou son amélioration *Alpha-Beta*[4] (Knuth and Moore, 1976)) avaient échoué. Cet algorithme est particulièrement bien adapté pour les problèmes de prises de décision dont l'espace de décision peut être représenté par un arbre. En combinant un arbre de recherche avec un échantillonnage aléatoire, MCTS construit et explore au fur et à mesure un arbre de recherche sur la base des données statistiques enregistrées lors des précédentes visites. L'arbre se construit en favorisant les chemins qui semblent les plus prometteurs sans toutefois négliger l'exploration de nouveaux chemins. Sous sa forme la plus simple, MCTS comporte principalement 4 étapes (voir figure 3.5) :

1. **Sélection.** Durant la phase de sélection, l'algorithme emploie une stratégie pour sélectionner récursivement un chemin depuis la racine jusqu'à un nœud terminal (feuille de l'arbre). La stratégie se fonde sur l'information acquise lors des précédentes explorations pour décider le prochain nœud fils à visiter. Les nœuds fils n'ayant encore jamais été visités sont testés en priorité.
2. **Expansion.** Un ou plusieurs nœuds fils correspondant à des séquences d'actions jamais explorées sont ajoutés à l'arbre courant.
3. **Simulation.** Une simulation aléatoire correspondant à la séquence d'actions choisie est réalisée. Cette simulation génère une sortie.
4. **Rétro-propagation.** L'information relative à la dernière simulation est rétro-propagée à l'ensemble des nœuds appartenant au chemin parcouru.

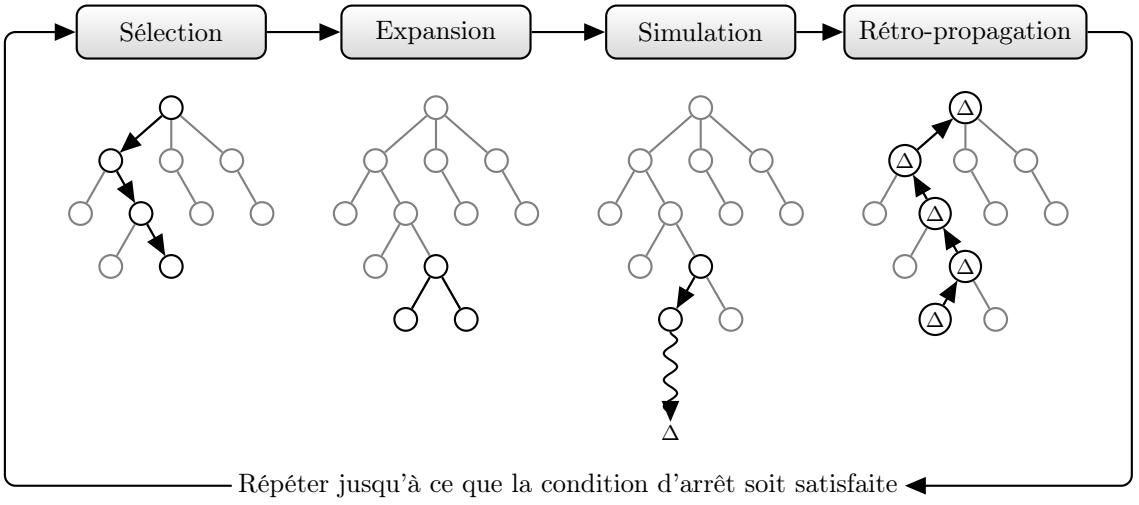

FIGURE 3.5 – Approche générale des arbres de recherche de Monte-Carlo (Browne et al., 2012)

4. en l'occurrence, l'algorithme *Alpha-Beta* a été employé sur la machine *Deep Blue* (Campbell et al., 2002) dans l'affrontement qui l'a opposé à Garry Kasparov, champion du monde de jeu d'échecs en 1997.

Les phases de *sélection* et d'*expansion* font appel à une stratégie (dite *tree policy*) ayant pour but de résoudre le compromis entre l'exploration (couvrir les régions encore inexplorées de l'arbre) et l'exploitation (choisir un nœud assurant un gain optimal, selon la connaissance actuelle). On verra dans la section (a.) l'algorithme UCT (Upper Confident bounds for Trees) (Kocsis and Szepesvári, 2006), une variante de MCTS utilisant une stratégie de sélection basée sur les algorithmes ayant été développés pour la résolution du problème bandit à plusieurs bras (voir la section 3.3.2) (Robbins, 1985).

L'algorithme est exécuté jusqu'à ce qu'une condition d'arrêt soit satisfaite. Cette condition peut être de différentes natures : limite de temps ou de mémoire, nombre maximum de parcours d'arbre etc. Dans tous les cas, l'algorithme est conçu pour retourner la meilleure décision ayant pu être testée durant l'ensemble des itérations. Si l'algorithme pouvait être exécuté pour une durée infinie, il examinerait toutes les possibilités.

Recommandation finale. Une fois l'exploration arrêtée, il faut exploiter l'information stockée dans l'arbre pour décider quelle séquence d'actions doit être réalisée. Schadd (2009) dresse une liste de 4 stratégies distinctes. On notera des similitudes entre les fonctions de recommandation **EBA** et **MPA** introduites par Bubeck et al. (2009) et les recommandations suivantes :

- **Max child :** Choisir l'action ayant le plus haut score moyen. Cette stratégie peut être risquée notamment si le nœud associé a été très peu visité, auquel cas la grande variance sur le score obtenu pour ce nœud entache d'incertitude son gain associé. On optera plutôt pour la méthode suivante, plus robuste.
- **Robust child :** Choisir l'action associée au nœud le plus visité. Cette méthode a du sens dans la mesure où la stratégie de sélection employée tend à favoriser l'exploration des chemins associés aux séquences d'actions les plus prometteuses au détriment des moins prometteuses.
- **Max-Robust child :** Choisir l'action associée au nœud ayant à la fois le plus haut score moyen et le plus grand nombre de visite. Si aucun nœud ne satisfait ces deux conditions, mieux vaut poursuivre l'exécution de l'algorithme.
- **Secure child :** Choisir l'action qui maximise une borne de confiance inférieure.

Au final, le choix d'une stratégie adaptée va dépendre des contraintes du problème, de la stratégie employée durant la phase de sélection et également du type d'information stockée durant la phase de rétro-propagation. Cette liste n'est donc pas exhaustive.

a. UCT

Introduit par Kocsis and Szepesvári (2006), l'algorithme UCT (pour Upper Confident bounds for Trees) est la combinaison de l'algorithme MCTS et de l'algorithme UCB décrit

précédemment. Aussi appelé *bandits hiérarchiques*, UCT consiste à associer à chaque nœud non-terminal un problème bandit dans lequel les K bras correspondent à K nœuds d'un arbre parmi lesquels il faut choisir pour construire récursivement un chemin. De la même manière que la solution de UCB converge vers le meilleur bras, la solution de UCT converge le meilleur chemin. L'algorithme UCB1, suggéré en tant que *Tree Policy*, introduit une légère modification pour tenir compte de la déviation de la valeur de récompense, liée au fait qu'un nœud est suivi par un sous-arbre progressivement exploré. Cette adaptation est caractérisée par l'introduction d'une constante C_p.

UCT tree policy.

- Initialisation : explorer une fois les nœuds fils.
- A l'instant t, choisir le nœud fils I_t tel que :

$$I_t = \underset{k \in \{1 \dots K\}}{\operatorname{argmax}} \bar{\mu}_{k,t} + 2C_p \sqrt{\frac{\log(t)}{n_{k,t}}} \tag{3.16}$$

La valeur de la constante C_p doit être choisie de manière à satisfaire les inégalités sur les queues (3.17) dérivées de l'inégalité de Hœffding, en faisant l'hypothèse que les valeurs de récompense obtenues sont identiquement et indépendamment distribuées.

$$\begin{gathered} P\left(\bar{\mu}_{k,t} \geq \mu_k + 2C_p \sqrt{\frac{\log(t)}{n_{k,t}}}\right) \leq t^{-4} \\ P\left(\bar{\mu}_{k,t} \leq \mu_k - 2C_p \sqrt{\frac{\log(t)}{n_{k,t}}}\right) \leq t^{-4} \\ \text{où } \mu_k = \lim_{t \to \text{inf}} \bar{\mu}_{k,t} \end{gathered} \tag{3.17}$$

Le reste de l'algorithme reste inchangé par rapport à la forme classique de MCTS.

3.3.4 BAAL : Un algorithme bandit dédié à l'apprentissage actif

Dans la thèse de Philippe Rolet (Rolet, 2011), l'auteur propose l'algorithme BAAL (*BAndit-based Active Learner*) (Rolet et al., 2009a,b), un algorithme exploitant UCT pour résoudre le problème de classification dans le cadre de l'apprentissage actif.

Dans le contexte de ces travaux en classification, l'apprenant joue contre une fonction oracle inconnue, supposée toutefois appartenir à l'espace des versions généré à partir du jeu initial de données étiquetées. À chaque tour de jeu, certaines instances sont sélectionnées et l'oracle en fournit une étiquette. À la fin du jeu, T exemples ont ainsi été étiquetés et forment l'ensemble d'apprentissage sur la base duquel la récompense est définie selon l'erreur de généralisation associée à l'hypothèse ainsi apprise. En amont de ce « vrai »

jeu, il est possible d'entraîner le joueur, et ainsi de mettre en place une bonne stratégie avant la confrontation réelle, en imitant le jeu présenté ci-dessus grâce à un oracle virtuel ou *subrogé* que l'on tire uniformément dans l'espace des versions. La récompense est alors calculée de la même façon, selon l'erreur de généralisation évaluée par rapport à l'oracle subrogé. On peut montrer qu'en répétant ces tours d'entraînement de nombreuses fois, la récompense empirique moyenne converge vers l'espérance de cette variable.

Chapitre 4

Planification d'expériences pour l'estimation paramétrique de modèles

Sommaire

« C'est la théorie qui décide de ce que nous pouvons observer. »

Albert Einstein

Nous nous intéressons ici à la planification d'expériences dans le cadre d'un problème d'estimation paramétrique de modèle. La définition d'une expérience est à comprendre au sens large puisqu'elle peut inclure le mode d'observation, les variables à observer, la résolution temporelle, ainsi que les conditions de l'expérimentation et le type de perturbations à appliquer au système étudié.

4.1 Planification d'expériences

4.1.1 Plan d'expériences classique et orienté-modèle

Dans le domaine de la planification d'expériences, on peut distinguer deux types d'approches.

1. L'approche *Black-Box Design of Experiment.*
   L'approche classique, introduite par Ronald Fisher en 1935, est qualifiée de **Black-box Design of Experiment** par Franceschini and Macchietto (2008). Cette approche s'appuie sur des *modèles comportementaux* (Walter and Pronzato, 1997) qui lient les *facteurs* (variables de contrôle d'un système) et les réponses (sorties du système) au cours d'une d'expérience. Dans cette approche, un plan d'expériences consiste à décider de la combinaison de facteurs qui engendrent la réponse la plus informative sur les caractéristiques du système. Par contraste avec les modèles dits mécanistes ou phénoménologiques, un modèle comportemental a pour principal objectif de produire un comportement équivalent à celui du système réel sans pour autant chercher à reproduire toutes les chaînes de causalité des différents processus qui le composent. En pratique, ces modèles sont choisis de manière arbitraire parmi une famille de modèles, généralement linéaires ou quadratiques. Cette approche est surtout adaptée aux problèmes statiques où les facteurs et les réponses ne varient pas au cours du temps. Elle fait appel aux plans factoriels (plans factoriels complets, fractionnaires, simplexes, ou encore les plans de Plackett et Bruman) décrits par Tinsson (2010). Comme ce type d'approche est peu adapté aux systèmes dynamiques, nous nous intéresserons ici au second type d'approches que représentent les plans d'expériences *orientés-modèle.*
2. L'approche *Model-based Design of Experiment.*
   La planification d'expérience orientée modèle (**MBDOE** ou **Model-based Design of Experiment**) (Galvanin et al., 2007, Franceschini and Macchietto, 2008, Galvanin et al., 2010) s'appuie sur des *modèles phénoménologiques* (Walter and Pronzato, 1997) visant à identifier les mécanismes sous-jacents au fonctionnement d'un système. Ces modèles découlent de la connaissance des principes fondamentaux de la

science en jeu (physique, chimie ou biologie, par exemple) et exploitent l'ensemble des connaissances, plus ou moins complètes, de la structure du système. Un avantage majeur de ce type d'approche est d'offrir un cadre approprié à la planification d'expériences pour les systèmes dynamiques (Shirt et al., 1994).

4.1.2 Approche générale pour la sélection de modèle et l'estimation paramétrique

Franceschini and Macchietto (2008) présentent une approche générale pour la planification d'expériences appliquée à l'estimation de paramètres (voir la figure 4.1). Celle-ci se décompose en trois étapes.

1. **Analyse d'identifiabilité et de sensibilité**.
   Cette première étape ne nécessite pas de traiter des données expérimentales mais consiste en une analyse mathématique des modèles candidats proposés. Durant cette phase, on commence par déterminer si les modèles sont structurellement identifiables (voir section 2.3). Ensuite on réalise une analyse de sensibilité. L'analyse de sensibilité peut permettre, entre autres, d'identifier les expériences pertinentes à envisager pour la prochaine étape. On se référera à la thèse de Dobre (2010) qui traite de manière détaillée de ces notions d'identifiabilité et de sensibilité locale ou globale.
2. **Discrimination de modèle**.
   Une fois les analyses de sensibilité terminées, on obtient un certain nombre de modèles identifiables et éligibles parmi lesquels il va falloir sélectionner le modèle le plus adéquat (ou plusieurs). L'étape suivante consiste alors à choisir les expériences les plus discriminantes possibles. On entame alors un cycle dans lequel on planifie des expériences, que l'on réalise ensuite, puis on estime les paramètres des modèles à partir des nouvelles données, en éliminant les modèles non conformes au fur et à mesure jusqu'à obtenir, idéalement, un seul modèle final.
3. **Précision des paramètres**.
   La dernière étape consiste à estimer le plus précisément possible les paramètres du modèle sélectionné au cours des étapes précédentes. On peut envisager deux types d'approches : l'approche séquentielle et l'approche parallèle. L'approche séquentielle consiste à identifier une expérience puis à la réaliser et à ré-itérer ce processus en se servant de l'information ainsi acquise. Cette approche est classiquement privilégiée lorsque les expériences ont un coût élevé, qui implique que leur nombre sera limité et leur choix crucial, ou bien lorsque l'équipement ne permet pas de réaliser plusieurs expériences simultanément. Il faut également qu'elles soient de suffisamment courte durée pour qu'il soit possible de les exécuter une à une dans un temps total

raisonnable. Dans l'approche parallèle, plusieurs expériences sont réalisées simultanément : l'utilisateur doit donc définir ces expériences en se servant seulement des données accumulées précédemment, si elles existent. Les données récoltées, dans ce cas, ne sont donc pas exploitées immédiatement mais seulement lorsque toute la série d'expériences simultanées est terminée, comme décrit dans la figure 4.1. On devine que l'approche séquentielle sera plus sûre car dans ce cas les décisions sont prises à l'aide de l'information la plus récente. Néanmoins, l'approche parallèle peut présenter des avantages pratiques notamment si elle est liée à des contraintes techniques. On peut également imaginer une approche hybride qui alterne les deux types d'approches.

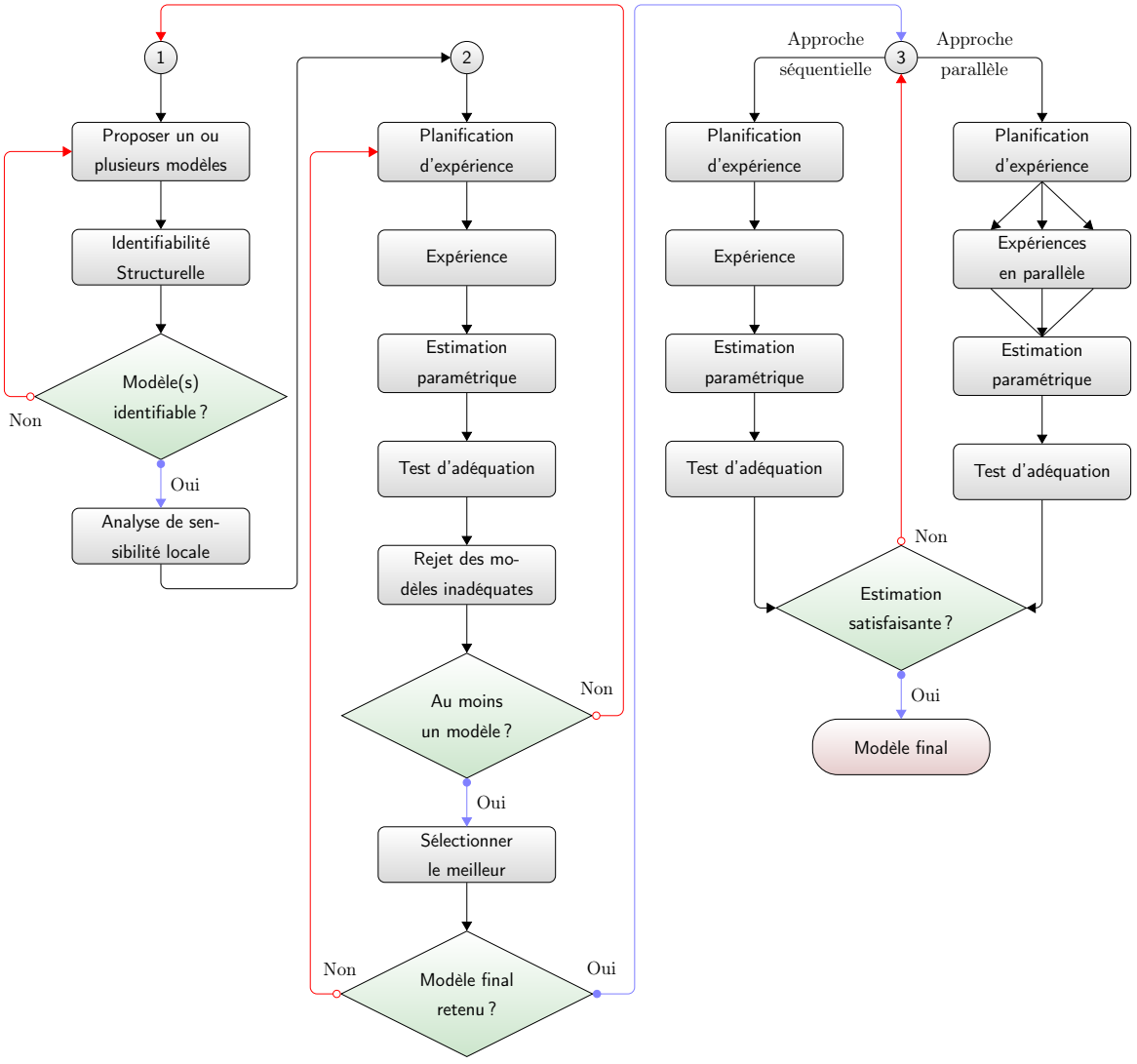

FIGURE 4.1 – Approche générale pour la planification d'expériences orientée modèle (Franceschini and Macchietto, 2008)

4.2 Plan d'expériences optimal

Cette section s'appuie en grande partie sur l'ouvrage de Atkinson et al. (2007). Le cadre des plans d'expériences « optimum » a été introduit en tout premier lieu dans Kiefer (1959). Alors qu'à l'époque ces travaux n'avaient pas rencontré pas un franc succès [1], il aura fallu attendre des travaux ultérieures (Atkinson, 1996) pour mettre en exergue l'apport de la théorie de Jack Kiefer pour la conception de plan d'expériences et la vérification de leur optimalité.

On considère le modèle paramétrique (4.1) décrivant un phénomène observé $\mathbf{y} = (y_1, y_2, \cdots, y_n)$ dépendant des *facteurs* $\mathbf{u} = (u_1, u_2, \cdots, u_m)$ de paramètres inconnus $\boldsymbol{\theta} \in \mathbb{R}^d$. On supposera une erreur d'observation additive, homoscédastique et sans biais $\boldsymbol{\epsilon} = (\epsilon_1, \epsilon_2, \cdots, \epsilon_n)$ dont chaque composante est indépendante des autres. Dans ce contexte, définir un plan d'expériences revient à sélectionner et à modifier la valeur des facteurs appropriés. La figure 4.2 montre le schéma du modèle d'expérience considéré.

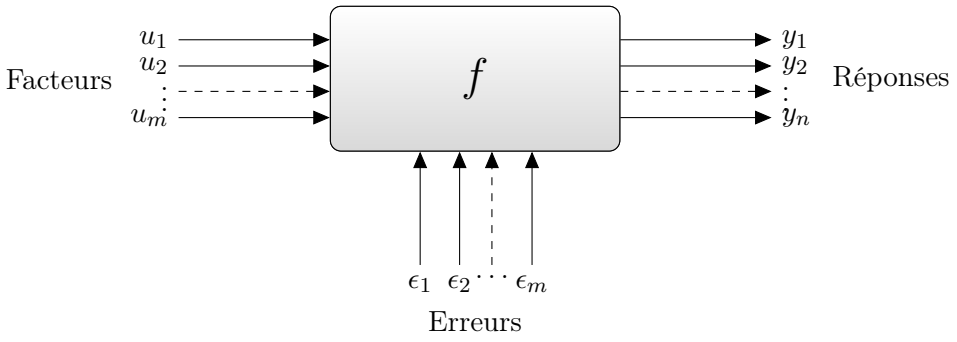

FIGURE 4.2 – Schéma du modèle d'expérience. L'expérimentateur agit sur les différents facteurs $(u_i)_{i=1\ldots m}$ et observe la réponse du système $\mathbf{y}$ avec une erreur $\boldsymbol{\epsilon}_i$

$$\begin{aligned} \mathbf{y}_i &= f(\mathbf{u}, \boldsymbol{\theta}) + \epsilon_i \\ \text{où } \forall i,\ \epsilon_i &\sim \mathcal{N}(0, \sigma^2) \\ \text{et } \forall i \neq j,\ &\operatorname{cov}(\epsilon_i, \epsilon_j) = 0. \end{aligned} \tag{4.1}$$

Dans ce cas l'estimation du vecteur de paramètre $\hat{\boldsymbol{\theta}}$ selon le critère des moindres carrés est donnée par le problème d'optimisation suivant :

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathbb{R}^d}{\operatorname{argmin}} \left(\mathbf{y} - f(\mathbf{u}, \boldsymbol{\theta})\right)^{\mathsf{T}} \left(\mathbf{y} - f(\mathbf{u}, \boldsymbol{\theta})\right) \tag{4.2}$$

1. Kiefer, présenta ces travaux réalisés durant son année sabbatique à la Royal Society. Il s'ensuivit une discussion qui s'avéra plutôt négative, à la surprise de Kiefer. A la fin il fut remercié « sarcastiquement », (Atkinson, 1996, Lehmann, 2007)

Si f est linéaire, il existe une formulation explicite de l'estimé $\hat{\boldsymbol{\theta}}$. En revanche, si f est ne serait-ce que partiellement non-linéaire par rapport au paramètre $\boldsymbol{\theta}$, son estimé $\hat{\boldsymbol{\theta}}$ ne peut être trouvé que de manière itérative.

4.2.1 Cas linéaire

On va s'intéresser en premier lieu au cas d'un modèle linéaire, avant d'étendre au cas non-linéaire. Dans ce cas le modèle s'exprime sous la forme $f(\boldsymbol{\theta}) = \mathbf{F}\boldsymbol{\theta}$, où $\mathbf{F}$ est une matrice de taille $(n \times d)$. Dans ce cas le problème d'optimisation devient le problème de régression linéaire suivant :

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathbb{R}^d}{\operatorname{argmin}} (\mathbf{y} - \mathbf{F}\boldsymbol{\theta})^\mathsf{T} (\mathbf{y} - \mathbf{F}\boldsymbol{\theta}) \tag{4.3}$$

En supposant que la matrice $\mathbf{F}^\mathsf{T}\mathbf{F}$ soit inversible on en déduit que le problème (4.3) admet une unique solution (4.4) obtenue par annulation du gradient de la fonction objectif.

$$\hat{\boldsymbol{\theta}} = (\mathbf{F}^\mathsf{T}\mathbf{F})^{-1}\mathbf{F}^\mathsf{T}\mathbf{y} \tag{4.4}$$

D'après l'égalité (4.4), l'estimé $\hat{\boldsymbol{\theta}}$ s'exprime comme une combinaison linéaire d'observations issues d'une loi normale : on peut donc en déduire que $\hat{\boldsymbol{\theta}}$ est normalement distribué. Sa matrice de covariance s'écrit :

$$var[\hat{\boldsymbol{\theta}}] = \sigma^2 (\mathbf{F}^\mathsf{T}\mathbf{F})^{-1} \tag{4.5}$$

Cette expression fait intervenir la *matrice d'information de Fisher* (MIF), qui s'exprime dans le cas présent comme $\boldsymbol{\mathcal{I}} = \mathbf{F}^\mathsf{T}\mathbf{F}$. Cette quantité est déterminante dans le choix d'expériences à réaliser pour obtenir les données : en effet, plus la MIF est « grande » (en un sens qui sera défini dans le paragraphe ci-après), plus les termes de la matrice de covariance de $\hat{\boldsymbol{\theta}}$ sont faibles et l'expérience associée peut être considérée comme informative.

4.2.2 Critère d'optimalité

Les critères d'optimalité présentés ci-dessous reposent sur l'interprétation, à l'aide d'une valeur scalaire, de la minimisation de l'inverse de la MIF $\mathbf{M}$ (ou de manière équivalente la minimisation de la matrice de covariance $\boldsymbol{\Sigma}$) relative au vecteur de paramètres estimé $\hat{\boldsymbol{\theta}}$. On ne présentera ici que les critères les plus couramment utilisés, à savoir les critères :A, B et E-optimalité. On notera pour une expérience e, $\boldsymbol{\mathcal{I}}(e)$ sa MIF, et e^* l'expérience optimale. $\boldsymbol{\mathcal{I}}(e)$ est toujours supposée inversible.

Définition 4.1. A-Optimalité

Un plan d'expériences est dit A-optimal (A vient du terme anglais *Average*) s'il minimise la moyenne des variances des estimateurs, ou de manière équivalente, s'il minimise la trace de l'inverse de la MIF.

$$e^* = \underset{e \in \mathcal{E}}{\operatorname{argmin}} \left(\operatorname{Tr}(\boldsymbol{\mathcal{I}}(e)^{-1}) \right)$$

Définition 4.2. D-Optimalité

Un plan d'expériences est dit D-optimal (D pour *Determinant*) s'il minimise le déterminant de l'inverse la MIF.

$$e^* = \underset{e \in \mathcal{E}}{\operatorname{argmin}} \left(\det(\boldsymbol{\mathcal{I}}(e)^{-1}) \right)$$

Définition 4.3. E-Optimalité

Un plan d'expériences est dit E-optimal (E pour *Eigenvalue*) s'il minimise la plus grande valeur propre de l'inverse la MIF.

$$e^* = \underset{e \in \mathcal{E}}{\operatorname{argmin}} \left(\max(\{\lambda \in \Lambda(\boldsymbol{\mathcal{I}}(e)^{-1})\}) \right)$$

La figure 4.3 illustre ces différentes notions d'optimalité en référence aux régions de confiance pour un vecteur de paramètres dans $\mathbb{R}^2$:

- Le critère d'E-optimalité, visant à réduire la plus grande valeur propre de $\boldsymbol{\mathcal{I}}^{-1}$, tend à réduire la taille de la zone de confiance des paramètres dans la direction du vecteur propre associé à la plus grande valeur propre.
- Comme la trace d'une matrice est aussi égale à la somme de ses valeurs propres et que les axes de l'ellipsoïde délimitant la zone de confiance sont dirigés selon les vecteurs propres associés, l'A-optimalité tend à réduire la moyenne des longueurs des axes de l'ellipsoïde.
- Enfin le déterminant étant égal au produit des valeurs propres, la D-optimalité permet de réduire le volume (la surface en 2D) de la région de confiance.

Pour ces propriétés le critère D-optimal est souvent préféré aux deux autres (Walter and Pronzato, 1997).

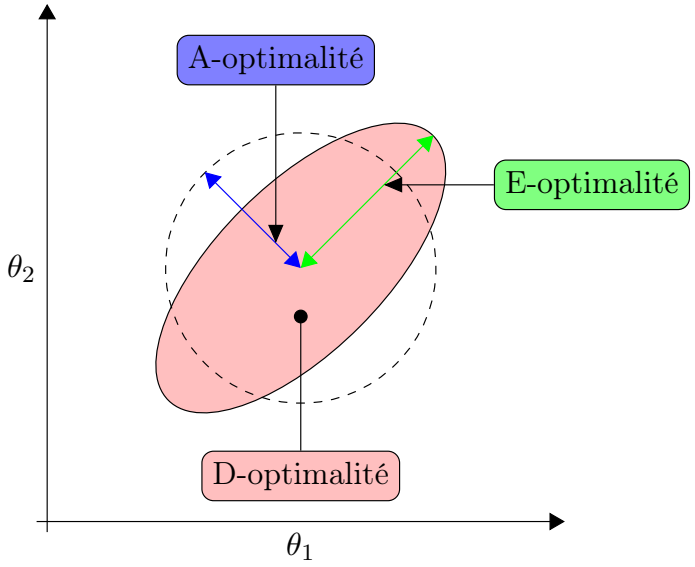

FIGURE 4.3 – Interprétation géométrique en dimension $d = 2$ des critères d'optimalité A, D et E. La région rose est l'ellipsoïde de confiance à $\alpha\%$ (généralement 95%) des paramètres. Le diamètre du cercle en pointillé vaut exactement la moyenne du grand axe et du petit axe de l'ellipse (Franceschini and Macchietto, 2008, Mazur, 2012)

Plus tard, Kiefer (1975) introduit la notion d'optimalité universelle, dont voici une définition :

Définition 4.4. Optimalité Universelle.
Un plan d'expériences e^* est dit universellement optimal si e^* minimise $\phi(\boldsymbol{\mathcal{I}}(e))$ où ϕ est une fonction scalaire ayant les propriétés suivantes :

1. ϕ est convexe.
2. $\phi(\alpha\boldsymbol{\mathcal{I}}(e))$ est non-croissante en fonction de $\alpha > 0$.
3. ϕ est invariant par permutation des lignes et des colonnes de $\boldsymbol{\mathcal{I}}(e)$.

Les optimalités A, D et E sont des cas particuliers d'optimalité universelle (Tinsson, 2010).

4.2.3 Cas non-linéaire

Dans le cas non-linéaire, l'estimation $\hat{\boldsymbol{\theta}}$ n'est en général pas possible sous une forme une analytique et sa valeur doit être approchée à l'aide d'algorithmes itératifs. Les critères d'optimalité décrits précédemment peuvent être adaptés en recourant à l'approximation de Taylor, comme le suggèrent Box and Wilson (1951).

Formule de Taylor (pour une fonction multi-variée).
Soit $f : \mathbb{R}^n \mapsto \mathbb{R}$ une fonction k fois différentiable au point $\mathbf{a} \in \mathbb{R}^n$. Alors il existe

$h_\alpha : \mathbb{R}^n \mapsto \mathbb{R}$ telle que :

$$f(x) = \sum_{|\alpha| \leq k} \frac{D^\alpha f(\mathbf{a})}{\alpha!}(\mathbf{x}-\mathbf{a})^\alpha + \sum_{|\alpha|=k} h_\alpha(\mathbf{x})(\mathbf{x}-\mathbf{a})^\alpha$$
$$\lim_{\mathbf{x}\to\mathbf{a}} h_\alpha(\mathbf{x}) = 0 \tag{4.6}$$
$$\text{où l'on a noté : } D^\alpha f = \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1} \dots \partial x_n^{\alpha_n}}$$

Pour illustrer la méthode de linéarisation, considérons le cas simple d'un modèle à réponse univariée, comme cela est fait par Mazur (2012). Dans ce cas on peut écrire :

$$E(\mathbf{y}) = \begin{pmatrix} f(u_1, \boldsymbol{\theta}) \\ \vdots \\ f(u_m, \boldsymbol{\theta}) \end{pmatrix} \tag{4.7}$$

En appliquant une approximation à l'ordre 1 à f en $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$ on obtient :

$$E(\mathbf{y}) \approx \begin{pmatrix} f(u_1, \hat{\boldsymbol{\theta}}) + \sum_{k=1}^{d} \frac{\partial f(u_1, \boldsymbol{\theta})}{\partial \theta_k}\Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} \cdot (\theta_i - \hat{\theta}_i)) \\ \vdots \\ f(u_n, \hat{\boldsymbol{\theta}}) + \sum_{k=1}^{d} \frac{\partial f(u_m, \boldsymbol{\theta})}{\partial \theta_k}\Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} \cdot (\theta_i - \hat{\theta}_i)) \end{pmatrix} \tag{4.8}$$

On en déduit l'équation linéaire suivante :

$$E(\mathbf{y}) - \begin{pmatrix} f(u_1, \boldsymbol{\theta}) \\ \vdots \\ f(u_m, \boldsymbol{\theta}) \end{pmatrix} = \mathbf{F}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) \tag{4.9}$$

où $\mathbf{F}$ est la matrice de sensibilité de la réponse f par rapport aux paramètres.

$$\mathbf{F} = \begin{pmatrix} \frac{\partial f(u_1, \boldsymbol{\theta})}{\partial \theta_1} & \cdots & \frac{\partial f(u_1, \boldsymbol{\theta})}{\partial \theta_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial f(u_m, \boldsymbol{\theta})}{\partial \theta_1} & \cdots & \frac{\partial f(u_m, \boldsymbol{\theta})}{\partial \theta_d} \end{pmatrix}\Bigg|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} \tag{4.10}$$

Les formules décrites précédemment sont alors appliquées à la matrice $\mathbf{F}$.

On pourra trouver d'autres exemples de critères d'optimalité dans la revue de Kreutz and Timmer (2009) qui inclut également des critères pour le problème de la planification d'expériences pour la discrimination entre différentes structures de modèles (supposée connue dans notre problème).

4.3 Plan d'expériences bayésien

Cette section s'inspire des travaux de thèse de Johanna Mazur (Mazur, 2012) consacrés aux plans d'expériences bayésiens pour l'inférence de réseaux de régulation génique. Pour une étude plus approfondie sur le sujet, notamment dans d'autres cadres d'application, le lecteur pourra se référer à Chaloner and Verdinelli (1995). Aussi appelée BED (pour Bayesian Experimental Design), cette théorie, introduite pour la première fois dans Lindley (1972), s'appuie sur des éléments de l'approche bayésienne et de la théorie de la décision.

Dans un plan d'expériences bayésien, on se place dans le cadre de l'estimation bayésienne (introduit en annexe 9) du paramètre $\boldsymbol{\theta} \in \Theta$ à estimer.

Un plan d'expériences bayésien est caractérisé par les éléments suivants :

1. Une distribution sur les paramètres caractérisée par une densité $p(\theta)$.
2. Une distribution sur la réponse observée $\mathbf{y}$ du système lors d'une expérience $e \in \mathcal{E}$, caractérisée par la densité conditionnelle (appelée *vraisemblance*), $p(\mathbf{y} \mid \boldsymbol{\theta})$.
3. Une fonction d'utilité $u(e, \boldsymbol{\theta}, \mathbf{y})$ décrivant l'utilité de réaliser l'expérience e et ainsi de collecter la donnée $\mathbf{y}$ quand le modèle prend en paramètres la valeur $\boldsymbol{\theta}$.

On définit alors l'utilité espérée (4.11) d'une expérience comme l'espérance sur tous les paramétrages possibles et toutes les données pouvant être issues de cette expérience, de l'utilité pondérée par la vraisemblance.

Définition 4.5. Utilité espérée.

$$U(e) = \int_{\mathbf{y} \in \mathcal{Y}} \int_{\boldsymbol{\theta} \in \Theta} u(e, \boldsymbol{\theta}, \mathbf{y}) p(\mathbf{y}|e) d\boldsymbol{\theta} d\mathbf{y} \tag{4.11}$$

Un plan d'expériences bayésien optimal consiste à choisir l'expérience qui maximise l'utilité espérée (4.12).

Définition 4.6. Maximum d'utilité espérée. (DeGroot, 1970)

$$e^* = \underset{e \in \mathcal{E}}{\operatorname{argmax}} \int_{\mathbf{y} \in \mathcal{Y}} \int_{\boldsymbol{\theta} \in \Theta} u(e, \boldsymbol{\theta}, \mathbf{y}) p(\mathbf{y}|e) d\boldsymbol{\theta} d\mathbf{y} \tag{4.12}$$

Il reste alors à construire une fonction d'utilité qui soit adaptée aux objectifs attendus des expérimentations. En effet « un plan d'expériences optimal pour l'estimation paramétrique ne l'est pas nécessairement pour la prédiction » (Chaloner and Verdinelli, 1995).

4.3.1 Fonctions d'utilités

Même en s'intéressant uniquement à la tâche d'estimation paramétrique, il est possible de construire différentes fonctions d'utilité plus ou moins appropriées.

Information de Shannon. Lindley (1956) propose d'employer l'*information de Shannon espérée*[2] comme fonction d'utilité adéquate :

$$U(e) = \int_{\mathbf{y}\in\mathcal{Y}} \int_{\boldsymbol{\theta}\in\Theta} \log\left(\frac{p(\boldsymbol{\theta}|\mathbf{y},e)}{p(\theta)}\right) p(\boldsymbol{\theta}|\mathbf{y},e)p(\mathbf{y}|e)d\boldsymbol{\theta}d\mathbf{y} \tag{4.13}$$

Opposée de la variance a posteriori espérée. Lorsque l'objectif attendu est une estimation ponctuelle des paramètres, l'utilité suivante est appropriée :

$$U(e) = -\int_{\mathbf{y}\in\mathcal{Y}} (\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})^{\mathsf{T}} A(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}) p(\boldsymbol{\theta}|\mathbf{y},e)p(\mathbf{y}|e)d\boldsymbol{\theta}d\mathbf{y} \tag{4.14}$$

où A est une matrice symétrique définie positive.

4.4 Plan d'expériences séquentiel

La planification séquentielle d'expériences consiste en une alternance entre les phases d'estimation et de planification d'expériences (voir figure 4.4) alors qu'en planification non-séquentielle d'expériences, un seul plan d'expériences est proposé puis réalisé. Cette approche est née des travaux sur l'analyse séquentielle de Wald (1945) dans le cadre des tests d'hypothèses statistiques, qui ont ensuite été généralisés à la sélection optimale d'expériences pour l'estimation par Chernoff (1959, 1972, 1981). L'idée générale est fondée sur le fait que les premières expériences peuvent éventuellement donner assez d'informations permettant ainsi d'éviter de réaliser inutilement d'autres expériences coûteuses. On cherche donc à identifier une sous-séquence d'expériences qui soit la plus efficace possible.

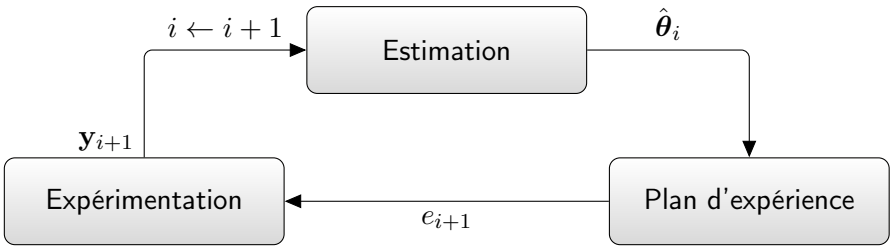

FIGURE 4.4 – Planification séquentielle d'expériences.

2. ou de manière équivalente *Divergence de Kullback-Leiber espérée*

À chaque itération, de nouvelles données $\mathbf{y}_i$ doivent être prises en compte lors de l'estimation courante $\hat{\boldsymbol{\theta}}_i$. La manière la plus naïve de procéder serait de moyenner les estimations obtenues à partir de chaque expérience individuelle. En notant $\hat{\boldsymbol{\theta}}(i)$ l'estimé à partir i-ème observation $\mathbf{y}_i$ et $\hat{\boldsymbol{\theta}}_k$ l'estimation à l'étape k à partir de l'ensemble des expériences réalisées jusqu'à l'étape k on obtiendrait :

$$\hat{\boldsymbol{\theta}}^k = \frac{1}{k}\sum_{i=1}^{k}\hat{\boldsymbol{\theta}}(i) \tag{4.15}$$

Cependant, comme l'indiquent Walter and Pronzato (1997), il est fortement déconseillé de procéder ainsi, car chaque estimation $\hat{\boldsymbol{\theta}}(i)$ est biaisée (Box, 1971), ce qui n'offre aucune garantie de convergence de $\hat{\boldsymbol{\theta}}_k$ vers le vrai paramètre lorsque le nombre de données d'expériences k tend vers l'infini. Il est donc préférable, bien que plus coûteux en terme de calcul, d'estimer $\hat{\boldsymbol{\theta}}_k$ à partir de l'ensemble des données $\mathbf{Y}_k = \cup_{i=1}^{k}\{\mathbf{y}_i\}$ et ce, à chaque itération.

4.4.1 Plan d'expériences séquentiel bayésien

Les plans d'expériences bayésiens se prêtent particulièrement bien à une approche séquentielle. Dans une approche bayésienne séquentielle, à chaque itération la probabilité *a posteriori* sur les paramètres obtenue à partir des précédente expériences sert de densité *a priori* pour les futures estimations, dans l'esprit de cette maxime :

Today's posterior is tomorrow's prior.

Plusieurs stratégies ont été développées dans la littérature. Berry (1985) développe une approche de type « bandit » (voir section 3.3.2). Des approches hybrides (mêlant des plans d'expériences séquentielle et parallèle) permettant de définir de façon séquentielle des groupes d'expériences ont aussi été envisagées par Zacks (1977) et Ridout (1995).

4.4.2 Plan d'expériences par apprentissage actif

Les disciplines de l'apprentissage actif et de la planification d'expériences ont été développées séparément par les communautés du domaine de l'apprentissage automatique et des statistiques. Il n'en demeure pas moins que ces deux disciplines partagent la même finalité, qui est de maximiser l'utilité des données ou des expériences. Il n'est donc pas étonnant que plusieurs travaux mêlent ces deux disciplines (Mary, 2005, Singh et al., 2005, Ho et al., 2011, Pauwels, 2013, Pauwels et al., 2014). On peut s'intéresser plus particulièrement aux travaux de Pauwels (2013), Pauwels et al. (2014) qui portent sur une problématique semblable à la nôtre, celle de la planification séquentielle d'expériences pour l'estimation

de paramètres d'équations différentielles ordinaires. En se plaçant dans le cadre d'une approche séquentielle bayésienne et en partant d'un critère d'utilité qui s'appuie sur la notion de risque empirique (que nous évoquions dans la section 3.1), les auteurs proposent un algorithme sélectionnant la prochaine expérience de manière à maximiser la réduction du risque espéré divisée par le coût de cette même expérience.

Algorithm 4: L'algorithme *NextExperiment* (Pauwels et al., 2014)

Input:

- Distribution courante : $\pi_k(\boldsymbol{\theta})$;
- Ensemble fini d'expériences : $\mathcal{E}$;
- Taille d'échantillon du jeu de paramètres : N;
- Taille d'échantillon du jeu d'observations : M;

Output: e_{k+1}

`/* Échantillonage des hypothèses a priori */`

$\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N \sim \pi_k(\boldsymbol{\theta})$

foreach $e \in \mathcal{E}$ **do**
  for $i = 1$ *to* N **do**
    `/* Echantillonnage des observables */`
    $\mathbf{Y}_1^i, \ldots, \mathbf{Y}_M^i \sim P(Y|\boldsymbol{\theta}_i)$;
    foreach $\mathbf{Y}_1^i, \ldots, \mathbf{Y}_M^i$ **do**
      `/* Calcul du risque empirique */`
      $R^N(e; \pi_k) = \frac{1}{N^2} \sum_{p,q=1}^{N} \ell(\boldsymbol{\theta}_p, \boldsymbol{\theta}_q) \cdot \omega_{pq}^M(e)$;
    end
  end
end

return $e_{k+1} = \underset{e \in \mathcal{E}}{\operatorname{argmax}} \dfrac{R(\pi_k) - R^N(e; \pi_k)}{b(e)}$

Le risque empirique est approché à l'aide des poids $\omega_{ij}^M(e)$ traduisant la similarité des comportements simulé à partir des hypothèses candidates $\boldsymbol{\theta}_i$ et $\boldsymbol{\theta}_j$.

$$\omega_{ij}^M(e) = \frac{1}{M} \sum_{m=1}^{M} \frac{P(\mathbf{Y}_m^i|\boldsymbol{\theta}_j; e)}{\sum_{n=1}^{N} P(\mathbf{Y}_n^i|\boldsymbol{\theta}_j; e)} \tag{4.16}$$

Le point fort de cette méthode est qu'elle prend en compte le coût des expériences et propose une fonction d'utilité pertinente. En revanche le budget, lui, n'est pris en considération, puisque l'approche est trop gloutonne pour permettre d'envisager l'anticipation

des futures expériences réalisable avec ce budget : en effet l'algorithme 4 requiert le parcours exhaustif d'un nombre arbitraire d'échantillons de paramètres et de réalisations d'observations pour chaque expérience. Dans ces conditions il n'est pas réaliste d'explorer des combinaisons d'expériences même pour des problèmes de taille modeste. Dans la suite de ce manuscrit nous comparerons cette approche à la nôtre sur l'un des problème de la compétition DREAM (voir la section 7.3.4).

Après ce nécessaire chapitre de fondamentaux et d'état de l'art, les prochains chapitres détaillent notre approche (Llamosi et al., 2014, Mezine et al., 2015).

Deuxième partie

Conduite d'expérience par apprentissage actif pour l'estimation de paramètres

Chapitre 5

Formulation séquentielle du problème de planification d'expériences pour l'estimation de systèmes dynamiques

Sommaire

« Une réponse approximative à une bonne question a bien plus de valeur qu'une réponse exacte à une mauvaise question. »

John Tukey

5.1 Motivation

Nous nous intéressons à la planification séquentielle d'expériences dans le but de résoudre le problème de l'identification de réseaux biologiques à partir d'observations sous la forme de séries temporelles. Pour mettre en lumière les caractéristiques du problème, nous introduisons l'exemple d'un réseau de régulation génique et de son identification. Nous suivons les règles des compétitions DREAM 6 et 7 pour la description de ce problème (voir détails en section 7.3). La structure du réseau est supposée connue et donnée sous la forme d'un graphe orienté représentant l'ensemble des gènes et de leurs interactions (activation ou inhibition). La figure 5.1 en fournit un exemple simple. Dans ce système, le gène 1 correspond à un nœud orphelin du graphe : il est dit autonome car il ne subit l'influence d'aucun autre gène. Il active l'expression des gènes 2 et 3. Le gène 2 s'auto-active et inhibe l'expression du gène 3.

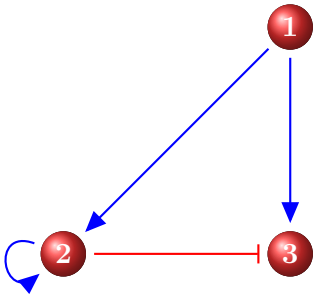

FIGURE 5.1 – Schéma d'un réseau régulation génique. Les flèches bleues représentent les relations d'activation ; les traits rouges les relations d'inhibition (Llamosi et al., 2014)

L'évolution du système est décrite par un système d'équations différentielles ordinaires, faisant intervenir les niveaux d'expression des gènes, représentés par les variables r_i et les niveaux de concentration de protéines représentés par les variables p_i. En appliquant le formalisme de Hill (voir section B.2) et à l'aide du graphe de régulation de la figure 5.1, on déduit le modèle suivant pour les produits des 3 gènes impliqués dans le système :

$$\begin{cases} \dot{r_1} &= \gamma_1 - k_{r_1} \cdot r_1 \\ \dot{r_2} &= \gamma_2 \cdot \left[\frac{{p_2}^{h22}}{{K_{22}}^{h22} + {p_2}^{h22}} \right] \cdot \left[\frac{{p_1}^{h12}}{{K_{12}}^{h12} + {p_1}^{h12}} \right] - k_{r_2} \cdot r_2 \\ \dot{r_3} &= \gamma_3 \cdot \left[\frac{{p_1}^{h13}}{{K_{13}}^{h13} + {p_1}^{h13}} \right] \cdot \left[\frac{{K_{13}}^{h13}}{{K_{13}}^{h13} + {p_1}^{h13}} \right] - k_{r_3} \cdot r_3 \\ \dot{p_1} &= \rho_1 \cdot r_1 - k_{p_1} \cdot p_1 \\ \dot{p_2} &= \rho_2 \cdot r_2 - k_{p_2} \cdot p_2 \\ \dot{p_3} &= \rho_3 \cdot r_3 - k_{p_3} \cdot p_3 \end{cases} \tag{5.1}$$

Soit sous la forme générale :

$$\begin{cases} \dot{\mathbf{x}}(t) &= f(\mathbf{x}, \boldsymbol{\theta}, t) \\ \mathbf{x}(t_0) &= \mathbf{x}_0 \end{cases} \tag{5.2}$$

Où $\mathbf{x} = [r_1, \ldots, r_d, p_1, \ldots, p_d]^\mathsf{T}$ est le vecteur d'état du système de taille $2d$, d représentant le nombre de gènes, et $\boldsymbol{\theta}^\mathsf{T} = [\boldsymbol{\theta}_{i=1\ldots p}]$ est le vecteur de paramètres à estimer. L'état initial $\mathbf{x}_0 = \mathbf{x}(t = t_0)$ est supposé connu. Comme dans les travaux de Quach et al. (2007) présentés dans le chapitre 2, il est préférable de considérer le modèle continu sous sa forme intégrale, de manière à obtenir les états à chaque instant d'observation t_k, avec k variant de 0 à K :

$$\mathbf{x}(t_k) = \mathbf{x}(t_{k-1}) + \int_{t_{k-1}}^{t_k} f(\mathbf{x}, \boldsymbol{\theta}, \tau) d\tau \tag{5.3}$$

L'état du système est partiellement observé à ces $K+1$ instants $t_0, \ldots t_K$: typiquement, sauf indication contraire, on ne mesure que les niveaux d'expression des gènes. On nomme $\mathbf{y}$ le vecteur d'observation qui est défini selon le modèle d'observation suivant :

$$\mathbf{y}(t_k) = h(\mathbf{x}, t_k) + \boldsymbol{\epsilon}_k, \, \forall k = 0 \ldots K \tag{5.4}$$

Où $h : \mathbb{R}^{2d} \times \mathbb{R} \mapsto \mathbb{R}^m$ est la fonction d'observation (ici, une projection) et $\boldsymbol{\epsilon}_k$ est une réalisation d'un bruit gaussien centré en zéro et de variance connue $\mathcal{N}(\mathbf{0}_m, (\mathbf{I}_m \cdot \boldsymbol{\sigma}_k)^2)$. On notera $\boldsymbol{\sigma}_{0:K} \in \mathbb{R}^m$, l'ensemble des $K+1$ vecteurs contenant les écarts-types du bruit selon chaque direction et à chaque instant t_k. Enfin, on notera $\boldsymbol{\Sigma}_k^{\mathcal{Y}} = (\mathbf{I}_m \cdot \boldsymbol{\sigma}_k)^2$ la matrice de covariance du bruit de mesure à l'instant t_k

Comme nous l'avons vu au chapitre 2, estimer avec une précision satisfaisante les paramètres de ce modèle ainsi que ses états cachés est un problème difficile, car se posent des problèmes de non-identifiabilité. Pour résoudre un tel problème, on peut acquérir des observations supplémentaires qui permettent de lever les incertitudes liées à cette non-identifiabilité. Perturber le système biologique, par exemple en inactivant un gène, permet d'obtenir des données très informatives qui nous renseignent sur le comportement du réseau en l'absence de l'activité potentiellement régulatrice du gène ciblé. Une telle perturbation est courante en génétique et est appelée *knock-out*. Il est également possible d'observer le comportement du réseau sous l'influence de bien d'autres types de perturbations (voir sections C.1 et C.2). Choisir les perturbations menant aux meilleures estimations des paramètres d'un modèle de réseau de régulation génique est la motivation première de cette thèse et des compétitions internationales DREAM 6 et 7.

Pour tirer parti des données observées sous l'influence des perturbations choisies, il nous faut modifier le modèle 5.2 de manière à ce qu'il puisse rendre compte de ces perturbations. Le nouveau modèle nous permettra non seulement de simuler le comportement du système

perturbé mais aussi d'exploiter plusieurs jeux d'observations, chacun correspondant à une condition basale ou à une perturbation. De manière classique, c'est en introduisant une variable de contrôle, $\mathbf{u}(t)$, que nous modélisons une perturbation. Nous en donnons ici un exemple pour une perturbation de type *knock-out* qui, appliquée au gène cible i, consiste à annuler l'expression de celui-ci et la production de la protéine correspondante. On définit la variable de contrôle $\mathbf{u}^{ko}(t)$ à l'instant $t \geq 0$ associée aux perturbations *knock-out*. Pour cibler i, on imposera : $\mathbf{u}^{ko}(t) = \mathbf{u}^{ko,i}(t)$, avec pour tout $t \geq 0$: $u_i^{ko,i}(t) = 0$ et $u_j^{ko,i}(t) = 1, \forall j \neq i$. Nous disposons ainsi de d variables de contrôle dédiées à chacun des *knock-out* possibles, chacune étant associée à un gène cible. Lorsqu'aucun gène n'est ciblé par le knock-out, on pose : : $\forall t \geq 0, \mathbf{u}^{ko}(t) = 0 \in \mathbb{R}^d$. En reprenant le système d'équations différentielles 5.1, on obtient :

$$\begin{cases} \dot{r_1} &= u_1^{ko}.\gamma_1 - k_{r_1}.r_1 \\ \dot{p_1} &= u_1^{ko}.\rho_1.r_1 - k_{p_1}.p_1 \\ \dot{r_2} &= u_2^{ko}.\gamma_2.\left[\frac{{p_2}^{h22}}{{K_{22}}^{h22}+{p_2}^{h22}}\right].\left[\frac{{p_1}^{h12}}{{K_{12}}^{h12}+{p_1}^{h12}}\right] - k_{r_2}.r_2 \\ \dot{p_2} &= u_2^{ko}.\rho_2.r_2 - k_{p_2}.p_2 \\ \dot{r_3} &= u_3^{ko}.\gamma_3.\left[\frac{{p_1}^{h13}}{{K_{13}}^{h13}+{p_1}^{h13}}\right].\left[\frac{{K_{13}}^{h13}}{{K_{13}}^{h13}+{p_1}^{h13}}\right] - k_{r_3}.r_3 \\ \dot{p_3} &= u_3^{ko}.\rho_3.r_3 - k_{p_3}.p_3 \end{cases} \tag{5.5}$$

Soit de manière plus générale :

$$\begin{cases} \dot{\mathbf{x}}(t) &= f(\mathbf{x}(t), \mathbf{u}^{ko}(t), \boldsymbol{\theta}, t) \\ \mathbf{x}(t_0) &= \mathbf{x}_0 \end{cases} \tag{5.6}$$

D'autres types de perturbations et d'expériences sont considérés au chapitre 7. En particulier, on peut évoquer la possibilité de choisir le modèle d'observation en indiquant quel type de variable mesurer et avec quel pas d'échantillonnage. Néanmoins pour simplifier la présentation du problème de *design expérimental* et sans perte de généralité, nous considérons simplement dans la suite que nous disposons d'un ensemble de taille n_{pert} de perturbations que nous savons modéliser par une variable de contrôle $\mathbf{u}$ pouvant prendre n_{pert} valeurs $\mathbf{u}^i, i = 1, \ldots n_{pert}$.

5.2 Formalisation du problème de planification séquentielle d'expériences

5.2.1 Ensemble d'expériences optimal par rapport à un budget donné

À partir de cette section, nous considérons plus généralement un système dynamique (5.7) dont le comportement peut être entièrement caractérisé par la connaissance du modèle f, la variable de contrôle $\mathbf{u}$ et ses n_{pert} valeurs possibles, le vecteur de paramètres $\boldsymbol{\theta} \in \mathbb{R}^d$ et l'état initial $\mathbf{x}_0$, à estimer.

$$\mathbf{x}(t) = \mathbf{x}(t_0) + \int_{t_0}^{t} f(\mathbf{x}(\tau), \mathbf{u}(\tau), \boldsymbol{\theta}, \tau) d\tau \tag{5.7}$$

Soient e, une expérience de perturbation définie par une variable de contrôle $\mathbf{u}^e$, $T_K^e = (t_i^e)_{i=0\dots K^e}$, une suite croissante d'instants de mesure, h, un modèle d'observation caractérisé par la fonction et $\boldsymbol{\epsilon}_k$, une variable aléatoire représentant le bruit. La série temporelle multivariée $[\mathbf{y}^e[0], \dots, \mathbf{y}^e[K^e]]$ résulte de l'observation bruitée du système sous l'influence de la perturbation e.

$$\begin{cases} \mathbf{x}^e(t_k) = \mathbf{x}^e(t_0) + \displaystyle\int_{t_0}^{t_k} f(\mathbf{x}^e(\tau), \mathbf{u}^e(\tau), \boldsymbol{\theta}, \tau) d\tau \\ \mathbf{y}^e[t_k] = h(\mathbf{x}^e(t_k)) + \boldsymbol{\epsilon}_k^e, \, \forall k \in T_K^e \end{cases} \tag{5.8}$$

avec $Y_e = Y(e)$, le processus aléatoire résultant de l'observation du système au cours de l'expérience e et $\mathbf{Y}_e$, une réalisation de ce même processus. Soit $\mathcal{E}$ un ensemble de n expériences représentées par leurs indices. On se donne un estimateur de $\boldsymbol{\theta}$ décrit par la fonction $\mathcal{T} : \mathbb{R}^{n(K+1)} \to \mathbb{R}^d$ dépendant des processus aléatoires $(Y_e)_{e=1\dots n}$. On notera τ l'estimation (ou l'estimé).

$$\hat{\boldsymbol{\theta}} = \mathcal{T}(Y_1, \dots, Y_n) \tag{5.9}$$

Le problème de conduite d'expériences sous contrainte de budget pour l'estimation de $\boldsymbol{\theta}$ peut être décrit de la manière suivante :

Étant donnés :

- Un ensemble $\mathcal{E}$ d'expériences réalisables.
- Une fonction de coût $b : \mathcal{E} \to \mathbb{N}^*$ associant un coût à chaque expérience.
- Un budget $B \in \mathbb{N}$ dédié à la réalisation de ces expériences.
- Une fonction d'*utilité* u qui mesure l'intérêt de réaliser une combinaison $\mathbf{c} \in \mathcal{P}(\mathcal{E})$ de $n_{\mathbf{c}}$ expériences $\mathbf{c} = \cup_{i=1}^{n_{\mathbf{c}}} \{e_i\}$ pour l'estimation de $\boldsymbol{\theta}$. Cette fonction peut, ou pas, dépendre de $\boldsymbol{\theta}$ lui-même (c'est par exemple le cas si l'utilité dépend du biais de l'estimateur) : $u(\mathcal{T}(\mathbf{Y}_{(\mathbf{c})}), \boldsymbol{\theta})$

- Une fonction d'*utilité générale* U qui est l'espérance de u selon la loi conditionnelle de $\mathbf{Y_c}$ sachant $\boldsymbol{\theta}$.

$$U(\mathbf{c}, \boldsymbol{\theta}) = \mathbb{E}[u(\mathcal{T}(\mathbf{Y_c}), \boldsymbol{\theta})|\boldsymbol{\theta}], \tag{5.10}$$

  où $\mathbf{Y_{(c)}} = (\mathbf{Y}_i)_{i=1,\dots,n_\mathbf{c}}$ est l'ensemble des processus aléatoires résultant de l'observation du système dynamique de paramètre $\boldsymbol{\theta}$ pour les $n_\mathbf{c}$ expériences de la combinaison $\mathbf{c}$.

Si nous connaissions $\boldsymbol{\theta}$, nous chercherions à construire un ensemble d'expériences $\mathbf{c}^*$ respectant le budget B et qui maximiserait cette utilité générale :

$$\begin{aligned} \mathbf{c}^* &= \underset{\mathbf{c}\in\mathcal{P}(\mathcal{E})}{\operatorname{argmax}}\, U(\mathbf{c}, \boldsymbol{\theta}) \\ \text{s.c. } & b(\mathbf{c}^*) \leq B \end{aligned} \tag{5.11}$$

La valeur de $\boldsymbol{\theta}$ étant bien sûr inconnue, on adopte une vision bayésienne : $\boldsymbol{\theta}$ est un vecteur aléatoire et on suppose connue sa loi de probabilité. On cherche alors à trouver le sous-ensemble $\mathbf{c}^*$ des expériences qui maximise l'espérance de l'utilité (ou *utilité espérée*), calculée par rapport à la loi jointe de $\boldsymbol{\theta}$ et des processus Y_i. L'expression devient alors :

$$\mathbf{c}^* = \underset{\mathbf{c}\in\mathcal{P}_B(\mathcal{E})}{\operatorname{argmax}}\, \mathbb{E}_{\boldsymbol{\theta}}[U(\mathbf{c}, \boldsymbol{\theta})] \tag{5.12}$$

Dans (5.12) on a introduit la notation $\mathcal{P}_B(\mathcal{E}) = \{\mathbf{c} \in \mathcal{P}(\mathcal{E}) \ / \ b(\mathbf{c}) \leq B\}$ qui représente les combinaisons d'expériences dont le coût ne dépasse pas le budget B. De ce fait, il n'est plus utile de faire apparaître explicitement la contrainte du problème d'optimisation.

5.2.2 Formulation du problème séquentiel

Les expériences de perturbation étant coûteuses, il est pertinent de s'intéresser au problème de planification séquentielle d'expériences, visant à autoriser la réalisation et l'exploitation d'une expérience dès qu'elle est recommandée. Ceci permet en particulier de mieux choisir les expériences suivantes. C'est le cadre qui a été proposé par le consortium DREAM et que nous nous proposons de résoudre. En partant d'une séquence initialement vide $\mathbf{s}_0 = \{\}$ et de l'ensemble des expériences $\mathcal{E}_0 = \mathcal{E}$, on construit, à chaque *tour*, les séquences suivantes $\mathbf{s}_n$ de manière itérative. Soit $B_n = B - b(\mathbf{s}_{n-1})$ le budget disponible à l'étape n, et $\mathcal{E}_n = \mathcal{E}_{n-1}/\{e_{n-1}\}$ l'ensemble d'expériences mis à jour en éliminant e_{n-1} la dernière expérience réalisée[1]. On peut alors ré-écrire l'équation (5.12) en faisant apparaître, dans les arguments de la fonction d'utilité, les $n-1$ premières variables correspondant aux données déjà obtenues aux tours précédents : l'espérance est donc calculée sur les variables aléatoires restantes, allant de n à $n + n_\mathbf{c}$. La meilleure combinaison

d'expériences en moyenne $\mathbf{c}_n^*$ s'écrit ainsi comme la solution du problème d'optimisation suivant :

$$\mathbf{c}_n^* = \underset{\mathbf{c}\in\mathcal{P}_{B_n}(\mathcal{E}_n)}{\operatorname{argmax}} \mathbb{E}_{\boldsymbol{\theta}|\mathbf{Y}_{1:n-1}} \mathbb{E}_{Y_{n:n+n_\mathbf{c}}} \left[u(\mathcal{T}(\mathbf{Y}_{1:n-1}, Y_{n:n+n_\mathbf{c}}), \boldsymbol{\theta})\right] \tag{5.13}$$

Dans cette expression, la deuxième espérance est calculée selon la loi des processus aléatoires résultant de l'observation du système dynamique de paramètre $\boldsymbol{\theta}$ donné tandis que la première est calculée selon la loi de la variable aléatoire $\boldsymbol{\theta}$ conditionnellement aux données déjà acquises $\mathbf{Y}_{1:n-1}$.

Cependant, même si l'on souhaite en effet anticiper sur les expériences suivantes qui pourront être faites, notre objectif est ici de ne recommander qu'une seule expérience à réaliser lors de ce tour, et non pas une combinaison d'expériences, afin de profiter des nouvelles données pour les recommandations aux tours suivants. La formulation complète du problème séquentiel que l'on va chercher à résoudre est donc finalement la suivante : étant données les observations réalisées $\mathbf{Y}_{1:n-1}$ à l'aide d'expériences précédemment choisies :

$$e_n^* = \underset{e\in\mathcal{E}_n}{\operatorname{argmax}} \sum_{\{\mathbf{c}\in\mathcal{P}_{B_n}(\mathcal{E}_n) \ / \ e\in\mathbf{c}\}} \frac{\mathbb{E}_{\boldsymbol{\theta}|\mathbf{Y}_{1:n-1}} \mathbb{E}_{Y_{n:n+n_\mathbf{c}}} \left[u(\mathcal{T}(\mathbf{Y}_{1:n-1}, Y_{n:n+n_\mathbf{c}}), \boldsymbol{\theta})\right]}{|\{\mathbf{c} \in \mathcal{P}_{B_n}(\mathcal{E}_n) \ / \ e \in \mathbf{c}\}|} \tag{5.14}$$

Ce problème d'optimisation (5.14) consiste à déterminer au sein de $\mathcal{E}$ l'expérience e_n^* qui, combinée avec d'autres expériences, permet de maximiser l'utilité espérée. Il s'agit donc de trouver l'expérience maximisant la moyenne empirique de cette utilité espérée sur l'ensemble des combinaisons qui la contiennent et qui respectent le budget B_n. Il est nécessaire de normaliser par le cardinal de cet ensemble $\{\mathbf{c} \in \mathcal{P}_{B_n}(\mathcal{E}_n) \ / \ e \in \mathbf{c}\}$ car ce nombre peut différer selon l'expérience e considérée : par exemple pour une expérience ayant un coût élevé, les combinaisons d'expériences la contenant tout en respectant le budget seront en moyenne moins longues, et par conséquent moins nombreuses.

Une fois cette expérience obtenue, il n'y a plus qu'à l'ajouter à la séquence courante.

$$\mathbf{s}_n = \mathbf{s}_{n-1} \cup e_n^* \tag{5.15}$$

1. sans perte de généralité, on fait le choix ici de présenter une formulation du problème dans le cas où les expériences choisies ne sont pas répétées. En pratique on pourra choisir de répliquer une expérience déjà réalisée ($\mathcal{E}_n = \mathcal{E}_{n-1}$) sans que cela n'affecte notre formulation. On peut également envisager de renouveler complètement le jeu d'expériences proposé en fonction des résultats des précédentes expériences.

5.3 Résolution approchée du problème de planification d'expériences

5.3.1 Approximation de l'utilité espérée

Étant donné le problème (5.14) introduit précédemment, nous notons ici $U(e, B_n, \mathcal{E}_n)$ l'utilité espérée de l'expérience e étant donnés le budget B_n et l'ensemble d'expériences $\mathcal{E}_n$.

$$U(e, B_n, \mathcal{E}_n) = \sum_{\{\mathbf{c} \in \mathcal{P}_{B_n}(\mathcal{E}_n) \ / \ e \in c\}} \frac{\mathbb{E}_{\boldsymbol{\theta}|\mathbf{Y}_{1:n-1}} \mathbb{E}_{Y_{n:n+n_\mathbf{c}}} \left[u(\mathcal{T}(\mathbf{Y}_{1:n-1}, Y_{n:n+n_\mathbf{c}}, \boldsymbol{\theta})) \right]}{|\{\mathbf{c} \in \mathcal{P}_{B_n}(\mathcal{E}_n) \ / \ e \in \mathbf{c}\}|} \tag{5.16}$$

Nous avons introduit la notion d'espace des versions dans la section 3.2.3. On le note ici $\mathcal{V}(\mathbf{Y}_{1:n-1})$ et il s'agit de l'ensemble des hypothèses cohérentes avec les données disponibles. Nous reviendrons sur cette notion de cohérence dans la section 6.3.1. Une première approximation dans le calcul de l'utilité consiste à exprimer le terme d'espérance en parcourant les paramètres non pas sur l'ensemble des hypothèses tout entier mais sur l'espace des versions. L'argument avancé est qu'il vaut mieux négliger l'influence des hypothèses incohérentes dans le calcul du maximum d'utilité espérée. On approche donc l'espérance sur $\boldsymbol{\theta}$ à l'aide d'un sous-ensemble représentatif de cet espace des versions que l'on note $\mathcal{V}(\mathbf{Y}_{1:n-1})$.

Concernant le calcul de l'espérance sur les processus Y, on note $j(\mathbf{c})$ le nombre de réalisations du bruit d'observation pour chaque combinaison $\mathbf{c}$. On obtient alors l'approximation suivante :

$$U(e, B_n, \mathcal{E}_n) \approx \sum_{\substack{\mathbf{c} \in \mathcal{P}_{B_n}(\mathcal{E}_n) \\ e \in \mathbf{c}}} \sum_{\boldsymbol{\theta} \in \mathcal{V}(\mathbf{Y}_{1:n-1})} \sum_{j=1}^{j(\mathbf{c})} \frac{u(\tau(\mathbf{Y}_{1:n-1}, \mathbf{Y}^j_{n:n+n_\mathbf{c}}), \boldsymbol{\theta})}{|\{\mathbf{c} \in \mathcal{P}_{B_n}(\mathcal{E}_n) \ / \ e \in \mathbf{c}\}| \cdot |\mathcal{V}(\mathbf{Y}_{1:n-1})| \cdot j(\mathbf{c})} \tag{5.17}$$

5.3.2 Formalisation dans le cadre des MDP et jeu à un joueur

Nous pouvons représenter notre problème sous la forme d'un processus de décision de markovien (MDP, voir section 3.3.1) à horizon fini. Cette formalisation nous permettra de proposer une méthode d'approximation adaptée. On représente donc notre problème sous la forme d'un MDP $\mu = \{\mathcal{S}, \mathcal{A}, T, R\}$, où :

- L'ensemble d'états $\mathcal{S} = \mathcal{P}_B(\mathcal{E})$ contient l'ensemble des combinaisons d' expériences réalisables à partir de l'ensemble d'expériences $\mathcal{E}$ étant donné le budget B.
- L'ensemble d'actions $\mathcal{A} = \mathcal{E}$ est représenté par l'ensemble d'expériences. Plus précisément, une action a_e implique la réalisation de l'expérience e pour laquelle on observe la donnée $\mathbf{Y}_e$.
- La fonction de transition T n'est pas explicitée. On fait donc appel à un simulateur.
- La récompense R est minimale (vaut 0) pour l'ensemble des états non terminaux ; sinon la récompense correspond à l'utilité associée à une réalisation des données de la séquence d'expériences considérée.

$$R(\mathbf{s}, e) = \begin{cases} u(\tau(\mathbf{Y}_\mathbf{s}, \boldsymbol{\theta})) \text{ si } \mathcal{P}_{B-b(\mathbf{s})-b(e)}(\mathcal{E}) = \{\} \\ \inf\limits_{\mathbf{s}\in\mathcal{S}} R(\mathbf{s}, e) \text{ sinon.} \end{cases} \tag{5.18}$$

Cette façon de définir la récompense en affectant la récompense minimale aux états non terminaux, c'est-à-dire les états où le budget n'est pas encore épuisé, est assez classique en théorie des jeux (Browne et al., 2012) et s'inspire plus précisément des travaux de Rolet (2011) en apprentissage actif pour la classification avec une utilité correspondant à l'erreur de généralisation (voir la section 3.3.4). Les auteurs montrent que dans ce cas la stratégie optimale au sens de Bellman du MDP fournit une stratégie optimale pour le problème d'apprentissage actif. Par ailleurs, comme ils le font également, on peut montrer que notre problème peut être vu comme un **jeu à un joueur** dans lequel, à chaque tour, un oracle fournit une information (un label dans le cas du problème de classification) qui permet de juger de la qualité de l'expérience sélectionnée durant la phase d'apprentissage.

Dans notre cas, le jeu à un joueur se formule ainsi :

- à chaque tour de jeu, étant donnés le budget courant et l'ensemble des expériences possibles ainsi que les données expérimentales déjà acquises, le joueur doit choisir une expérience à réaliser
- en retour, le laboratoire réalise cette expérience, fournit les données expérimentales correspondantes, décroît le budget et met à jour l'ensemble des expériences.

Résoudre ce MDP par programmation dynamique n'est pas envisageable en pratique en raison de son aspect combinatoire. En effet, la programmation dynamique exige d'énumérer l'ensemble des transitions possibles, ce qui n'est pas réaliste dans le cas où l'espace des états $\mathcal{S}$ est de taille exponentielle par rapport au nombre d'expériences. En revanche la recherche d'une solution **approchée** est réalisable à l'aide des arbres de recherche de Monte-Carlo (MCTS). La stratégie adoptée pour résoudre ce MDP est donc de combiner l'exploration efficace de l'ensemble des séquences d'expériences à l'aide d'algorithme de la famille MCTS, avec une exploration uniforme des espaces de paramètres candidats et de l'espace de sortie des expériences.

5.3 Synthèse

Dans ce chapitre, nous avons décrit la forme générale du problème cible de cette thèse et nous l'avons illustré par l'exemple concret d'un réseau de régulation génique dont la dynamique est régie par un système d'équations différentielles, pour lequel nous avons décrit un ensemble d'expériences (perturbations et mesures) applicable sur ce système. Nous avons proposé une formulation séquentielle du problème de planification d'expériences pour l'estimation de systèmes dynamiques biologiques. En s'appuyant sur la théorie des plans d'expériences bayésiens, nous avons proposé une extension de l'expression du critère d'utilité espérée qui prend en considération le budget, le coût des différentes expériences et l'utilité des combinaisons d'expériences. À l'aide du formalisme des modèles de décision Markoviens, nous avons replacé le problème dans le cadre de l'apprentissage par renforcement. Cette démarche nous permet d'avoir accès aux méthodes de résolution approchées propres à ce type de modèle.

Chapitre 6

EDEN, méta-algorithme d'apprentissage actif pour la planification séquentielle d'expériences

Sommaire

« Vie ou néant ! choisir. Ah ! quelle discipline ! - Que n'est-il un Eden entre ces deux usines ? »

Jules Laforgue

Dans ce chapitre, nous proposons de résoudre le problème de planification séquentielle d'expériences sous contrainte de budget pour l'estimation d'un système dynamique par apprentissage actif. Une vue globale de l'algorithme est d'abord présentée avant de décrire chaque composante de la solution.

6.1 Description générale de l'approche adoptée dans EDEN

6.1.1 Principe général de l'approche

Pour résoudre notre problème de planification d'expériences, nous proposons une approche incrémentale baptisée EDEN (pour « **E**xperimental **DE**sign for **N**etwork inference », illustrée par la figure 6.1). La figure reprend l'approche générale de planification séquentielle décrite dans 4.1, à l'étape 3 où le modèle n'est plus remis en question et où on cherche exclusivement à estimer précisément les valeurs des paramètres. À la différence de l'approche générale présentée dans la figure 4.1, notre approche prend en compte un budget et le coût de chaque expérience dans la planification des expériences et dans le critère d'arrêt.

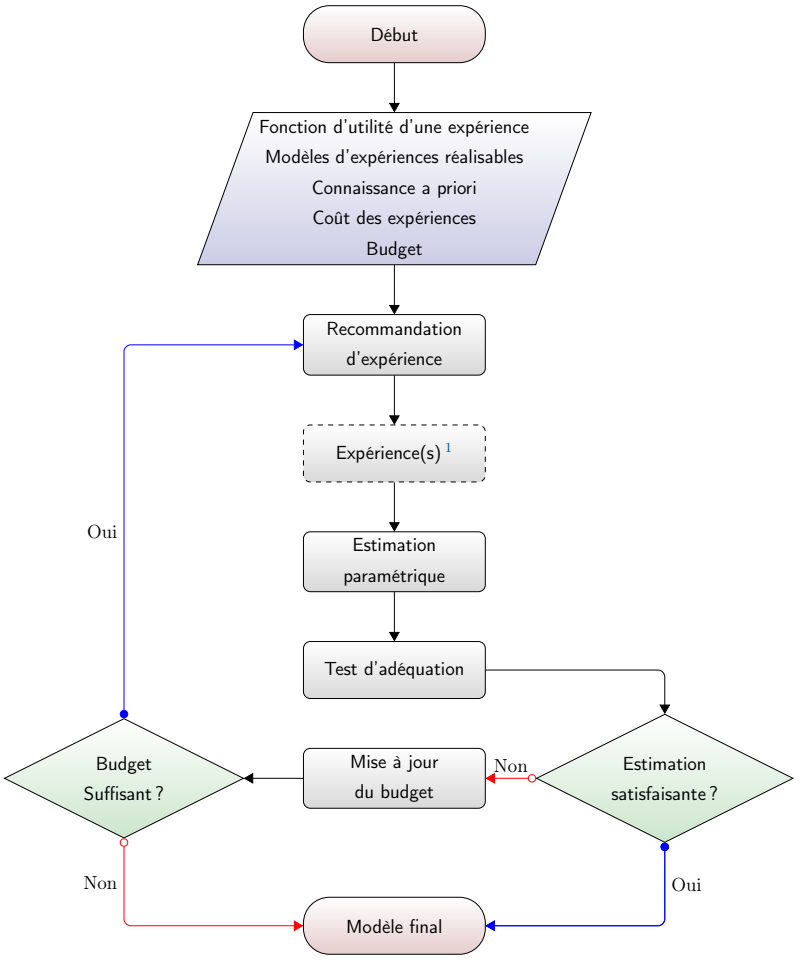

FIGURE 6.1 – Approche générale adoptée par le méta-algorithme EDEN (**E**xperimental **DE**sign for **N**etwork inference)

EDEN alterne les quatre étapes suivantes :

1. recommander une/des expérience(s) en s'appuyant sur les résultats d'une exploration efficace de l'espace des combinaisons d'expériences réalisables avec le budget disponible (voir section 6.3),
2. réaliser les expériences par l'intermédiaire d'un expert humain ou d'un robot sur la base des recommandations fournies à l'étape précédente,
3. estimer des paramètres et les états cachés du modèle à partir des données courantes (voir section 6.2),
4. répéter les étapes 1 à 3 jusqu'à ce qu'une condition d'arrêt soit satisfaite.

Les conditions d'arrêt correspondent aux deux *scenarii* suivants :

- la qualité du modèle est satisfaisante sans que le budget soit totalement épuisé et dans ce cas l'algorithme fournit un modèle satisfaisant à un coût réduit. Ce cas

1. dans un problème réel, cette étape d'expérimentation est réalisée par un opérateur humain dans un laboratoire.

peut être identifié par le fait que l'acquisition de nouvelles données ne fait plus varier significativement les valeurs des paramètres estimés et que les prédictions du modèle sont en bon accord avec toutes les données accumulées, avec une tolérance à définir (liée aux différentes incertitudes sur le modèle ou sur le bruit des données, en particulier).

- le budget est épuisé, auquel cas l'algorithme fournit une des meilleures estimations possibles des paramètres du modèle. C'est le cas le plus courant dans les contextes qui nous concernent dans cette thèse.

Alors que dans le premier scénario, l'objectif visé est de dépenser le minimum de ressources, dans le second scénario, il s'agit de fournir l'estimation la plus précise possible. L'incertitude initiale sur les paramètres rend ces objectifs difficiles à atteindre avec un seul plan d'expériences. L'approche séquentielle vise à réduire progressivement l'incertitude sur les paramètres en fournissant des recommandations de plus en plus pertinentes de manière à approcher au mieux l'un de ces deux objectifs.

Afin d'offrir un cadre plus général, EDEN est formulé sous la forme d'un *méta-algorithme* décrit par l'algorithme 5. Le terme « méta-algorithme » exprime l'idée qu'EDEN fait appel à différents types d'algorithmes afin de s'adapter au problème spécifique à traiter, la structure globale étant conservée. On peut voir, par exemple, que l'étape qui consiste en l'estimation du modèle à partir des données courantes, comme on le verra par la suite, différentes méthodes d'inférence sont éligibles pour réaliser cette tâche. Les types d'expériences possibles peuvent également être très variables suivant les contextes : il peut s'agir, comme dans le problème qui nous occupe, d'altérer un réseau de régulation mais il pourrait également s'agir de manière plus générale de faire varier les variables de contrôle d'un système dynamique.

En proposant un tel méta-algorithme, c'est-à-dire un algorithme générique, on facilite ainsi l'automatisation et la mise en oeuvre du programme, tout en laissant à l'opérateur humain une marge de manoeuvre pour adapter EDEN au type de système considéré.

Algorithm 5: EDEN : Experimental DEsign for Network inference

Data: Budget total : B ; Ensemble d'expériences discret et fini : $\mathcal{E}$;
Modèles de chaque expérience : $(\mathcal{M}_e)_{e\in\mathcal{E}}$; Expériences déjà réalisées : $\mathbf{s}_0$;
Données initiales : $\mathbf{Y}_0 = \{\mathbf{y}^e_{0:K^e}\}_{e\in s_0}$; Connaissance *a priori* : $\Theta_0 \leftarrow \Theta|_{\mathbf{Y}_0}$;
Fonction de coût d'une expérience : $b()$;

Algo : Estimation de paramètres : $\mathcal{L}earner$; Calcul d'utilité : $\mathcal{R}eward$.

Input: Modèle initial : $\mathcal{M}_{\mathbf{s}_0} = \cup_{e\in\mathbf{s}_0}(\mathcal{M}_e)$; Fonction de perte : $\ell()$;
Budget intermédiaire : $\beta \leq B$; Nombre de parcours d'arbre : N ;
Horizon : L ; Taille de l'espace des versions : $N_{\mathcal{V}}$;
Stratégie de sélection : ϕ ; Stratégie de recommandation : ψ ;

Output: Loi *a posteriori* : $\hat{\Theta}|_{\mathbf{Y}_{\mathbf{s}_n}}$.

`/* Initialisation */`
$n \leftarrow 0$; $\mathcal{E}_n \leftarrow \mathcal{E}$; $B_n \leftarrow B$; $\mathbf{Y}_{\mathbf{s}_n} \leftarrow \mathbf{Y}_0$; $\mathcal{M}_{\mathbf{s}_n} \leftarrow \mathcal{M}_0$;

while $\inf_{e\in\mathcal{E}_n} b(e) \leq B \vee \neg$(*Modèle suffisamment précis*) **do**

`/* Représentation discrète de l'espace des versions */`
$\mathcal{V}(\mathbf{Y}_{\mathbf{s}_n}) \leftarrow \left\{(\boldsymbol{\theta}_i)_{i=1:N_{\mathcal{V}}} \overset{iid}{\sim} \Theta_0 \ / \ \forall i = 1\dots N_{\mathcal{V}}, \boldsymbol{\theta}_i \text{ est cohérent avec } \mathbf{Y}_{\mathbf{s}_n}\right\}$

`/* Exploration par recherche arborescente de Monte-Carlo */`
$tree \leftarrow \text{MCTS}(\phi, N, \mathcal{E}, \mathcal{V}(\mathbf{Y}_{\mathbf{s}_n}), min(\beta, B), L, \mathcal{L}earner, \mathcal{R}eward)$;
$e_n \leftarrow \psi(tree)$; `// Recommandation d'expérience(s)`

$\mathbf{Y}_n \leftarrow Experimentation(e_n)$; `// Expérience dans le monde réel`

`/* Mise à jour des données du problèmes */`
$\mathcal{E}_n \leftarrow \mathcal{E}_n \backslash \{e_n\}$; `// Mise à jour de l'ensemble d'expériences`
$B_n \leftarrow B_n - b(e_n)$; `// Mise à jour du budget`
$\mathbf{Y}_{\mathbf{s}_n} \leftarrow \mathbf{Y}_{\mathbf{s}_n} \cup \{\mathbf{Y}_{e_n}\}$; `// Concaténation des données`
$\mathcal{M}_{\mathbf{s}_n} \leftarrow fusion(\mathcal{M}_{\mathbf{s}_n}, \mathcal{M}_{e_n})$; `// Fusion de modèles`

`/* Estimation a posteriori */`
$\hat{\Theta}|_{\mathbf{Y}_{\mathbf{s}_n}} \leftarrow \mathcal{L}earner(\Theta_0, \mathcal{M}_n, \mathbf{Y}_{\mathbf{s}_n}, \ell())$
$\Theta_0 \leftarrow \hat{\Theta}|_{\mathbf{Y}_{\mathbf{s}_n}}$; `// Mise à jour de l'a priori`
$n \leftarrow n + 1$;

end

return $\hat{\Theta}|_{\mathbf{Y}_{\mathbf{s}_n}}$

6.1.2 Méta-algorithme EDEN : l'enchaînement des différents modules qui le constituent

L'algorithme 5 décrit l'application du méta-algorithme EDEN à l'estimation des paramètres d'un modèle dynamique. Nous allons ici en décrire l'enchaînement puis nous détaillerons dans les parties suivantes de ce chapitre chacun des modules qui le constituent.

On se donne tout d'abord un jeu de données initial $\mathbf{Y}_0$ qui est généralement acquis sur le système non perturbé (dit *wildtype* dans le cadre de notre application sur les réseaux de régulation). On dispose également d'un budget total B et d'une liste d'expériences définissant l'espace de design $\mathcal{E}$, qui correspondent dans notre cadre d'application à des perturbations à appliquer au réseau *wildtype*. La connaissance *a priori* Θ_0 contient l'information disponible sur les paramètres à estimer issue de précédentes estimations ou de connaissance experte. On peut l'exprimer sous différentes formes (population d'hypothèses candidates, domaine de variation, loi de probabilité). On se donne également une fonction permettant d'attribuer à chaque expérience son coût $b()$.

Tant que le budget le permet ou que les estimations obtenues ne sont pas encore satisfaisantes, on va réitérer un ensemble d'étapes se caractérisant par une expérience à réaliser. Dans la réalité, chaque expérience est effectuée en laboratoire et correspond à l'une des perturbations incluses dans l'espace de design $\mathcal{E}$. Comme nous n'avons pas accès ici à de l'expérimentation réelle, ces expériences sont simulées *in silico*.

Cet ensemble d'étapes débute tout d'abord par la constitution d'une représentation discrète de l'espace des versions. Cet espace est constitué de vecteurs de paramètres candidats, aussi appelés *hypothèses* ou *oracles*, qui génèrent des trajectoires simulées suffisamment proches des données, en un sens à définir (section 6.3.1). Il s'agit ensuite de déterminer quelle expérience recommander, en tenant compte du fait qu'elle sera suivie par d'autres expériences, pour autant que le budget restant le permette. On utilise pour cela une représentation des différentes séquences d'expériences possibles sous forme d'arbre que l'on explore via un algorithme MCTS (section 6.3), afin d'estimer de manière efficace l'espérance des fonctions d'utilité associées à ces différentes séquences d'expériences pour en identifier la plus intéressante (section 6.3.3). On réalise l'expérience e_n ainsi déterminée ce qui produit un nouveau jeu de données $\mathbf{Y}_{e_n}$ venant s'ajouter aux données $\mathbf{Y}_{s_n}$. On soustrait au budget le coût de l'expérience e_n et, si l'on ne souhaite pas répliquer les expériences (ce qui est souvent le cas sauf si le bruit est très important et nécessite des répétitions), on la retire du jeu d'expériences possibles. On réalise également la fusion du modèle $\mathcal{M}_{e_n}$ avec le modèle courant $\mathcal{M}_{\mathbf{s}_n}$ (section a.). On peut alors réaliser l'étape d'estimation qui fournit les distributions *a posteriori* utilisées comme nouveaux *a priori* pour l'étape suivante (voir section 6.2).

On va à présent décrire plus en détail chacun des modules qui constituent EDEN.

6.2 Description du module d'estimation

Plusieurs méthodes peuvent être envisagées pour le module d'estimation et dans tous les cas, il est nécessaire de pouvoir prendre en compte un ensemble de données qui augmente à chaque nouvelle phase d'acquisition.

6.2.1 Estimation de modèles fusionnés

a. Fusion de modèles

Réaliser une estimation à partir de plusieurs jeux de données correspondant à des expériences différentes pose le problème de tenir compte de chaque modèle avec des valeurs de variables de contrôle différentes. Hecker et al. (2009) proposent de résoudre le problème du filtrage de Kalman à partir de plusieurs sources par une stratégie heuristique basée sur des scores de corrélation afin de déterminer un ordre optimal dans lequel alimenter le filtre de Kalman.

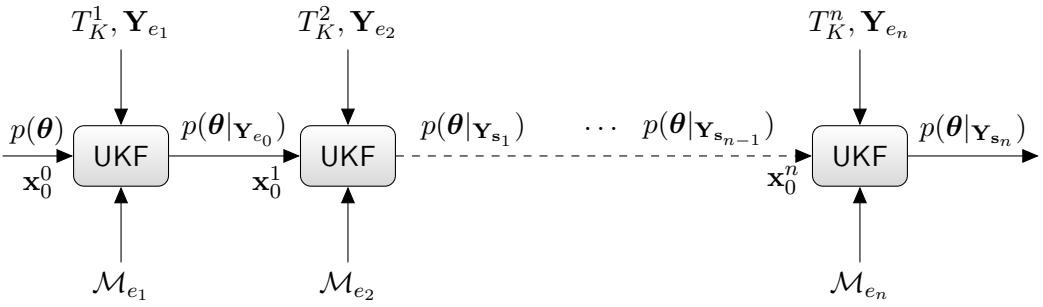

FIGURE 6.2 – Méthode standard d'estimation en ligne avec UKF à partir de plusieurs d'expériences

À l'origine le filtre de Kalman est conçu pour résoudre des problèmes en temps réel, dans lesquels les données sont traitées à la volée dans l'ordre dans lequel elles sont générées. Or, dans la configuration dans laquelle on se place, similaire à celle des travaux de Quach et al. (2007), Hecker et al. (2009), Baker et al. (2011), les données sont traitées en aval : on peut donc tout à fait adapter l'ordre dans lequel les observations sont données au filtre. Cependant, comme le soulignent les auteurs, le choix de l'ordre est un problème $NP-$difficile puisque l'espace de recherche est l'ensemble des permutations simples (de taille $n!$, où n représente le nombre total d'expériences). Les plans d'expériences que nous présenterons dans la partie 7 comportent jusqu'à 13 expériences, soit plus de 6 milliards

de permutations possibles. En admettant l'existence d'un ordre optimal de traitement des données, et d'un algorithme capable d'approcher cet ordre, il reste que l'estimation est toujours réalisée en ligne. Or une méthode hors-ligne, qui prend en compte toutes les données d'un seul coup, donne en général de bien meilleurs résultats qu'une méthode en ligne. Satz (2001) réalise une analyse comparative entre le filtre Kalman et la méthode BLS (Batch Least Square), mettant en évidence le fait qu'une méthode hors-ligne converge plus rapidement et plus précisément qu'une méthode en ligne.
Dans cette thèse nous proposons aussi une solution permettant d'appliquer le filtre de Kalman de façon hors ligne vis à vis des expériences, mais cette solution est de différente nature. Il s'agit de construire artificiellement un super-modèle fusionné combinant les états et les variables de contrôle de chaque expérience, mais partageant le même vecteur de paramètres. La figure 6.3 montre la façon d'estimer les paramètres à partir du modèle fusionné.

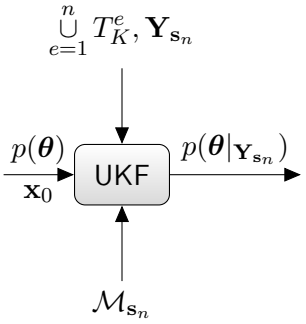

FIGURE 6.3 – Estimation hors ligne avec UKF à l'aide de modèle fusionné

Du point de vue du filtre de Kalman, il s'agit d'un filtrage classique ; cependant à chaque étape du filtrage, c'est bien l'ensemble des données expérimentales qui sont prises en compte simultanément.

Pour construire ce modèle fusionné, on considère comme nouveau vecteur d'état la concaténation des vecteurs d'état associés à chacune des expériences réalisées jusqu'à l'étape courante : $\mathbf{x} \leftarrow (\mathbf{x}, \mathbf{x}^i) \in \mathbb{R}^{d \times i}$ où les vecteurs d'état des différents modèles sont supposés de taille identique d ($d/2$ étant ainsi le nombre de gènes du réseau dans l'exemple présenté en section 5.1). Le modèle fusionné $\mathcal{M}$ s'écrit, à partir des notations de l'équation 5.8 :

$$\begin{cases} \mathbf{x}(t_0) = \mathbf{x}_0 \\ \mathbf{x}(t_k) = \mathbf{x}(t_0) + \int_{t_0}^{t_k} F(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau, \boldsymbol{\theta}) d\tau \\ \mathbf{y}(t_k) = h(\mathbf{x}(t_k), t_k) + \boldsymbol{\epsilon}_k, \forall k = 0 \dots K^e \end{cases} \tag{6.1}$$

où $\mathbf{u}(t) = \left[\mathbf{u}^1(t), \mathbf{u}^2(t), \ldots, \mathbf{u}^k(t)\right]^\mathsf{T}$ est le vecteur formé de la concaténation des vecteurs de variables de contrôle des différents modèles $\mathcal{M}_e,\ e = 1, .., n$ correspondant aux n expériences réalisées. La fonction d'évolution F donne l'évolution du vecteur d'état fusionné. Le vecteur des erreurs d'observations s'écrit enfin : $\boldsymbol{\epsilon}_k = \left[\boldsymbol{\epsilon}_k^1, \boldsymbol{\epsilon}_k^2, ..., \boldsymbol{\epsilon}_k^n\right]^\mathsf{T}$ où l'indice k désigne le temps d'acquisition de la donnée observée et l'indice en exposant de 1 à n désigne le numéro de la perturbation associée. Le vecteur de paramètres $\boldsymbol{\theta}$ est quant à lui supposé commun à tous les ensembles d'équations qui composent le modèle fusionné. La figure 6.4 schématise le modèle d'espace d'état associé à la fusion de modèle.

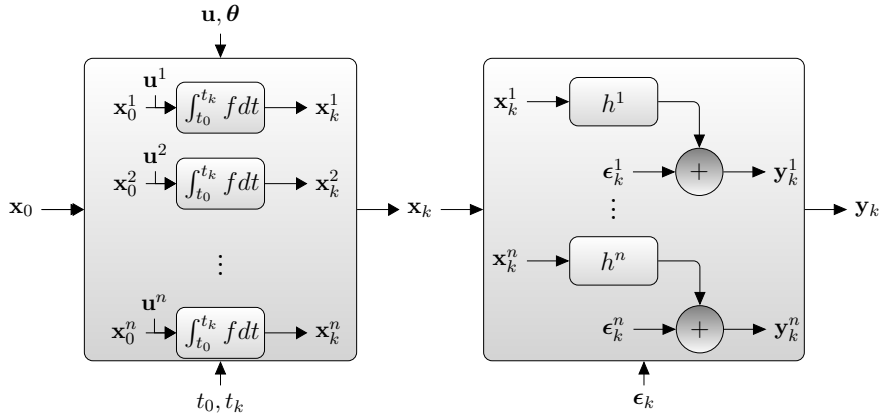

FIGURE 6.4 – Schéma du modèle à espace d'état fusionné. Le bloc de gauche représente le modèle d'évolution du modèle fusionné qui à partir des vecteurs d'état $\mathbf{x}_0$ de contrôle $\mathbf{u}$ et de paramètre partagés $\boldsymbol{\theta}$, propage les états projetés $\mathbf{x}_0^1, \ldots \mathbf{x}_0^n$ de l'instant t_0 à t_k pour ensuite les concaténer en $\mathbf{x}_k$. À droite, le modèle d'observation du modèle fusionné calcule à partir $\mathbf{x}_k$ et du bruit additif $\boldsymbol{\epsilon}_k$ les états observés de chaque expérience indépendante pour ensuite les concaténer en $\mathbf{y}_k$

Pour donner un aperçu de la forme que peut prendre un modèle fusionné, reprenons le système que nous évoquions précédemment (5.1). On se place dans un scénario où on dispose des observations issues de deux expériences : la première expérience comprend des mesures réalisées sur le système *wildtype* ($\forall t \geq 0, \forall i, u_i^{ko}(t) = 1$), la seconde expérience observe le comportement système ayant subi une perturbation de type *knock-out* sur gène 1 en affectant la valeur 0 à la variable de contrôle adéquate ($\forall t \geq 0, u_1^{ko}(t) = 0$). On introduit la variable d'état du modèle fusionné $\mathbf{x} = \left[r_1^1, p_1^1, \ldots, r_3^1, p_3^1, r_1^2, p_1^2, \ldots, r_3^2, p_3^2\right]$ dont la dimension est le double de la dimension initiale. Le modèle obtenu contient donc deux fois plus de variables scalaires d'état, mais possède le même nombre de paramètres inconnus qu'avant fusion puisqu'il représente toujours le même système observé dans deux conditions de perturbations différentes.

$$\begin{cases} \dot{r}_1^1 &= \gamma_1 - k_{r_1}.r_1^1 \\ \dot{p}_1^1 &= \rho_1.r1 - k_{p_1}.p_1^1 \\ \dot{r}_2^1 &= \gamma_2.g_{2,2}^+(p_2^1).g_{2,1}^+(p_2^1) - k_{r_2}.r_2^1 \\ \dot{p}_2^1 &= \rho_2.r_2^1 - k_{p_2}.p_2^1 \\ \dot{r}_3^1 &= \gamma_3.g_{3,1}^+(p_3^1).g_{3,1}^-(p_3^1) - k_{r_3}.r_3^1 \\ \dot{p}_3^1 &= \rho_3.r_3^1 - k_{p_3}.p_3^1 \\ \dot{r}_1^2 &= 0 - k_{r_1}.r_1^2 \\ \dot{p}_1^2 &= 0 - k_{p_1}.p_1^2 \\ \dot{r}_2^2 &= \gamma_2.g_{2,2}^+(p_2^2).g_{2,1}^+(p_2^2) - k_{r_2}.r_2^2 \\ \dot{p}_2^2 &= \rho_2.r_2^2 - k_{p_2}.p_2^2 \\ \dot{r}_3^2 &= \gamma_3.g_{3,1}^+(p_3^2).g_{3,1}^-(p_3^2) - k_{r_3}.r_3^2 \\ \dot{p}_3^2 &= \rho_3.r_3^2 - k_{p_3}.p_3^2 \end{cases} \tag{6.2}$$

b. Algorithme d'estimation basé sur l'algorithme UKF avec plusieurs initialisations

L'algorithme UKF étant sensible aux conditions initiales, la présence de minima locaux peut entraîner une convergence vers différentes solutions pour différentes conditions initiales données. Pour résoudre ce problème, nous appliquons une procédure consistant à relancer plusieurs estimations à partir d'initialisations différentes (voir la figure 6.5). À partir de l'espace des versions associé aux données acquises lors des précédentes expériences $\mathcal{V}(\mathbf{Y}_{0:n-1})$, nous définissons des bornes inférieures et supérieures sur les valeurs des paramètres. Ces bornes permettent de réduire l'espace de recherche des paramètres à l'hypercube contenant l'espace des versions et, également, à initialiser l'espérance associée à la loi *a priori* sur les paramètres. La matrice de covariance est, quant à elle, toujours initialisée à la matrice identité.

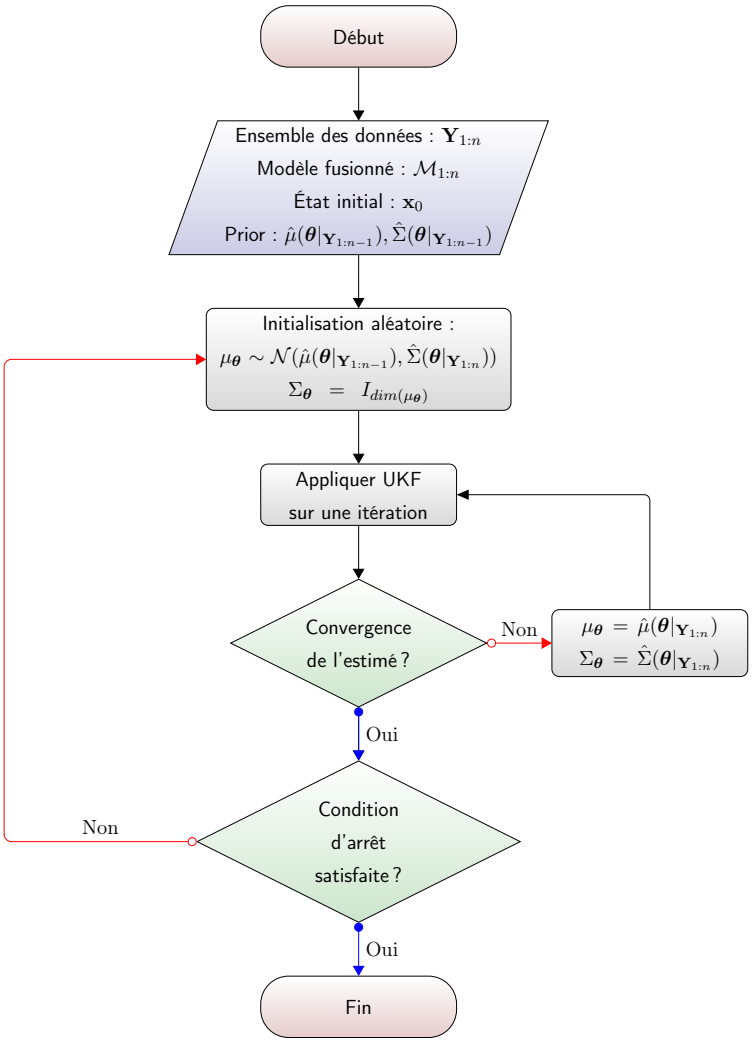

FIGURE 6.5 – Procédure d'estimation par filtrage de Kalman avec plusieurs ré-initialisations

c. Algorithme d'estimation basé sur l'algorithme ESS

L'algorithme ESS classique (b.) étant avant tout un algorithme d'optimisation globale, il retourne uniquement la solution optimale parmi les points explorés, ce qui correspond donc à l'un des minima locaux trouvés. Cependant dans notre cas, le problème n'est pas vraiment d'exhiber une unique solution mais plutôt d'identifier l'ensemble des points qui génèrent des trajectoires cohérentes avec les données disponibles. Du fait des problèmes de non-identifiabilité, cet ensemble peut potentiellement contenir un grand nombre de points. Comme on explore l'espace de recherche de manière discrète, il s'agit donc de faire en sorte que l'algorithme renvoie une population de points respectant le critère de cohérence défini en section 6.3.1. Il faut également veiller à garder une diversité suffisamment importante pour ne pas que cette population soit toute entière concentrée sur une seule zone de cet espace, ce qui se produit lorsqu'on laisse fonctionner un algorithme heuristique contenant

un mécanisme d'exploitation des zones contenant les optima locaux trouvés. Pour résoudre ce problème, nous appliquons la version classique de l'algorithme ESS et nous construisons en parallèle l'échantillon de l'espace des versions. Pour ce faire, à chaque évaluation de la fonction objectif par l'algorithme ESS nous évaluons la cohérence du vecteur de paramètres associé afin de déterminer s'il fait partie de l'espace des versions. Si tel est le cas il est ajouté à l'échantillon. On stoppe l'algorithme dès que l'on a atteint une taille d'échantillon suffisant.

d. Fonctions objectifs

Test du $\mathcal{X}^2$. Nous avons testé deux types de fonctions objectifs différentes. L'une s'appuie sur la statistique du χ^2 et l'autre sur la statistique de Kolmogorov-Smirnov. On présente ici une formulation dans le cadre de l'estimation à partir d'expériences multiples. Le test du χ^2 défini en Equation (6.3) est le plus couramment utilisé. L'algorithme UKF tend à minimiser ce critère.

$$\ell_{\chi^2}(\mathcal{M}, \mathbf{Y}, \boldsymbol{\theta}) = \sum_{\mathbf{Y}_e \in \mathbf{Y}} \sum_{\mathbf{y}_k^e \in \mathbf{Y}_e} \left(\mathbf{y}_k^e - h^e(\mathbf{x}_k^e(\boldsymbol{\theta}), t_k)\right)^{\mathsf{T}} \left(\boldsymbol{\Sigma}_k^e\right)^{-1} \left(\mathbf{y}_k^e - h^e(\mathbf{x}^e(\boldsymbol{\theta}), t_k)\right) \tag{6.3}$$

$$\text{où } \boldsymbol{\Sigma}_k^e = \text{diag}(\boldsymbol{\sigma}_k^e)^2$$

où l'on a repris les notations de la section 5.2 : $\mathbf{Y}_e$ parcourt l'ensemble des vecteurs de données disponible, noté $\mathbf{Y}$, qui rassemble, pour chaque expérience réalisée sous les variables de contrôle $\mathbf{u}^e$, l'ensemble des m_e variables, observées aux différentes dates associées $k = 0, \ldots, K^e$. Le terme $\mathbf{y}_k^e$ représente donc le vecteur des mesures réalisées lors de l'expérience e au temps t_k. Le terme $\boldsymbol{\Sigma}_k^e$ est la matrice de covariance du bruit de mesures de l'expérience e à l'instant de mesure t_k. On donnera, dans la partie résultat, un exemple concret de la forme que peut prendre ce terme $\boldsymbol{\sigma}_k^e$ dans le cas d'une combinaison de bruits multiplicatif et additif dans le cadre du challenge DREAM. Cependant, contrairement au critère quadratique moyen minimisé par le filtre de Kalman, ce critère ne prend pas en compte le caractère gaussien du modèle de bruit qui est l'une de nos hypothèses.

Test de Kolmogorov-Smirnov. La statistique de Kolmogorov-Smirnov est un choix approprié pour construire une fonction objectif qui prenne en compte le caractère gaussien du bruit d'observation : ce ne sera bien sûr la plus intéressante que si cette information est disponible, comme c'est le cas dans un de nos cas d'application. Le test de Kolmogorov-Smirnov permet de déterminer avec un certain niveau de confiance si un échantillon est issu d'une loi normale. Le test se base sur une comparaison entre la fonction de répartition théorique d'une loi normale et la fonction de répartition empirique de l'échantillon testé.

On note $S_{\mathcal{M},\mathbf{Y},\boldsymbol{\theta}}$ l'échantillon normalisé des résidus entre l'ensemble des données $\mathbf{Y}$ et la réponse du modèle fusionné $\mathcal{M}$ pour l'hypothèse $\boldsymbol{\theta}$:

$$S_{\mathcal{M},\mathbf{Y},\boldsymbol{\theta}} = \left\{ \frac{\mathbf{y}_k^e[j] - h_j^e(\mathbf{x}_k^e(\boldsymbol{\theta}), t_k))}{\boldsymbol{\sigma}_k^e[j]} \right\}_{j=\{1,\dots,m_e\}, \mathbf{y}_k^e \in \mathbf{Y}_e, \mathbf{Y}_e \in \mathbf{Y}}$$

Où $\mathbf{y}_k^e[j]$ et $h_j^e(\mathbf{x}_k^e(\boldsymbol{\theta}))$ sont respectivement la valeur observée et réponse du modèle au temps t_k de l'expérience e selon la j-ème coordonnée, et $\boldsymbol{\sigma}_k^e[j]$ est l'écart type du bruit de mesure qui leur est associé.

En notant F la fonction de répartition de loi gaussienne et $F_{S_{\mathcal{M},\mathbf{Y},\boldsymbol{\theta}}}$ la fonction de répartition empirique associée à l'échantillon $S_{\mathcal{M},\mathbf{Y},\boldsymbol{\theta}}$, la fonction objectif du problème d'optimisation basé sur la statistique de Kolmogorov-Smirnov s'écrit :

$$\ell_{ks}(\mathcal{M}, \mathbf{Y}, \boldsymbol{\theta}) = \sup_x (F(x) - F_{S_{\mathcal{M},\mathbf{Y},\boldsymbol{\theta}}}(x)) \tag{6.4}$$

6.3 Description du module de recommandation d'expériences

6.3.1 Construction de l'espace discrétisé des versions

À l'issue du module d'estimation, nous obtenons un ensemble de solutions potentielles pouvant être représentées par une distribution *a posteriori* (avec UKF) ou sous la forme d'une population de référence (avec ESS). Afin d'évaluer l'utilité des futures expériences, le module de recommandation se fonde sur cet ensemble de solutions pour simuler uniquement les trajectoires les plus plausibles. Pour y parvenir, le module de recommandation commence par constituer un ensemble discret et fini de solutions représentatives de l'espace des versions.

Nous avons introduit la notion d'espace des versions dans la section relative à l'apprentissage actif (3.2). Pour rappel, l'espace des versions $\mathcal{V}(\mathbf{Y})$ associé à un jeu de données $\mathbf{Y}$ est par définition l'ensemble des valeurs de paramètres cohérentes avec l'ensemble des observations. La stratégie que nous avons décrite dans la section précédente 5.2 requiert d'échantillonner uniformément cet ensemble. Dans cette section, nous précisons la notion de cohérence à l'aide de critères quantitatifs, puis nous proposons une méthode d'échantillonnage adaptée à notre problème.

a. Critères de cohérence

Pour un problème classique de classification supervisée sans bruit, il est aisé de définir une notion de cohérence. En effet, on pourra dire dans ce cas qu'une hypothèse est cohérente

avec l'ensemble des observations si le modèle paramétrique associé à cette hypothèse attribue aux exemples d'apprentissage les mêmes classes que l'oracle. En revanche, pour notre problème d'estimation à partir de séries temporelles bruitées, la cohérence d'une hypothèse avec les observations est plus délicate à définir. Une manière d'aborder ce problème serait de s'appuyer sur la notion de région de confiance comme nous l'évoquions dans la section 4.2.2. Ces régions de confiance qui sont représentées par des ellipsoïdes, déduites de la matrice de covariance ou de la matrice d'information de Fischer, peuvent être utilisées pour délimiter l'espace des versions. Il s'agirait de recouvrir l'espace des versions avec des ellipsoïdes comme cela est fait par Zamora-Sillero et al. (2011). Néanmoins, étant donné la forme complexe (potentiellement non-convexe) de l'espace des versions, ce type de méthode n'est pas envisageable en grande dimension. De plus, la plupart des stratégies que nous avons décrites (voir la section 4.2) pour construire ces ellipsoïdes s'appuient sur un critère mono-objectif (basé sur la vraisemblance) ; or un critère multi-objectif prenant en considération l'ensemble des jeux de données serait plus judicieux. Dans cette thèse, l'approche que nous proposons consiste à construire une population de solutions candidates satisfaisant un certain nombre de contraintes de cohérence avec les séries temporelles observées. Ces contraintes sont formulées sous la forme de tests statistiques appliqués aux vecteurs de résidus des séries temporelles de chaque observation.

À l'intérieur du module d'estimation, nous nous servons d'une fonction d'erreur pour construire une fonction de vraisemblance sur l'ensemble des données. Dans le module de recommandation nous employons cette même fonction d'erreur sur chacune des séries temporelles observées indépendamment. En nous appuyant sur ces fonctions d'erreur nous proposons la définition suivante de la notion la cohérence :

Définition 6.1. Critère de cohérence avec les observations
Soit un jeu de données $\mathbf{Y_c} = (\mathbf{Y}_e)_{e\in\mathbf{c}}$ correspondant à des mesures réalisées pour une combinaison d'expériences $\mathbf{c} \in \mathcal{P}(\mathcal{E})$ auquel est associé un ensemble de modèles $\mathcal{M}_\mathbf{c} = (\mathcal{M}_e)_{e\in\mathbf{c}}$. On dira qu'une hypothèse $\boldsymbol{\theta}$ est cohérente par rapport à cet ensemble de données, si pour chaque série temporelle relative à chaque expérience, l'écart entre l'erreur de la trajectoire hypothèse et l'erreur de la trajectoire idéale ne dépasse pas un certain seuil de tolérance.

$$\forall(\mathcal{M}_e, \mathbf{Y}_e) \in \mathcal{M}_\mathbf{c} \times \mathbf{Y_c}, \forall j \in \{1, \ldots, m_e\}, \ell^j(\mathcal{M}_e, \mathbf{Y}_e, \boldsymbol{\theta}) \leq \lambda.\ell^j_{\text{idéale}}(\mathcal{M}_e, \mathbf{Y}_e) \qquad (6.5)$$

où ℓ^i est une fonction de perte qui ne prend en compte que l'erreur selon la j-ème coordonnée des observables. $\ell^j_{\text{idéale}}$ est la valeur selon la j-ème coordonnée de la solution *idéale* du problème multi-objectifs :

$$\ell^j_{\text{idéale}}(\mathcal{M}_e, \mathbf{Y}_e) = \min_{\boldsymbol{\theta}\in\Theta} \ell^j(\mathcal{M}_e, \mathbf{Y}_e, \boldsymbol{\theta})$$

et $\lambda > 1$ est un hyperparamètre permettant d'ajuster le seuil de tolérance. Il est important de respecter l'inégalité stricte puisque qu'il n'y a aucune garantit qu'il existe un paramètre satisfaisant la trajectoire idéale.

En détaillant les fonctions de pertes selon chacune des coordonnées de manière indépendante, cela nous garantie qu'il n'y ait pas d'incohérence selon l'une de trajectoire particulière.

b. Construction de l'espace discrétisé des versions

L'échantillonnage uniforme de l'espace des versions est une condition nécessaire pour approcher correctement l'espérance d'utilité moyenne maximale donnée par l'équation (5.17). En pratique, on utilise une version discrétisée de l'espace des versions qu'on échantillonne uniformément. Cette version discrétisée de l'espace continu des versions doit refléter le mieux possible l'ensemble (infini) des paramètres cohérents avec les données jusqu'ici acquises. Nous pourrions choisir directement les optima locaux obtenus par les algorithmes d'estimation. Cependant, les optima locaux obtenus à l'étape précédente sont trop restrictifs et peuvent ne pas inclure les vraies solutions. Ainsi, pour mieux couvrir l'espace des versions (non discrétisé), nous proposons dans un premier temps, une méthode en deux étapes. Nous appliquons un algorithme de classification non supervisée sur l'ensemble des optima locaux issus de l'estimation. Après cette étape préliminaire, on obtient une partition en un nombre arbitraire de classes (ou *clusters*), regroupant les solutions similaires entre elles. Dans un second temps, on commence d'abord par échantillonner uniformément l'ensemble des partitions. Puis pour chaque partition sélectionnée on tire uniformément une solution appartenant à celle-ci. Cette dernière procédure est décrite par l'algorithme 6 étant donné un nombre N_p de partitions construit au préalable.

Algorithm 6: Méthode d'échantillonnage uniforme sur l'espace des versions

Data:

- Espace de version partitionné : $\cup_{j=1:N_p} \{(\mathcal{P}_j, \cup_{\boldsymbol{\theta}\in\mathcal{V}}\{\boldsymbol{\theta} \in \mathcal{P}_j\})\}$;

Output: Échantillon tiré uniformément : $\boldsymbol{\theta}^*$;

`/* Tirage uniforme d'une partition */`

$\mathcal{P}^* \sim \{\mathcal{P}_1, \ldots, \mathcal{P}_{N_p}\}$;

`/* Tirage uniforme d'un oracle subrogé dans la partition sélectionnée */`

$\boldsymbol{\theta}^* \sim \{\boldsymbol{\theta} \in \mathcal{P}^*\}$;

return $\boldsymbol{\theta}^*$

En ce qui concerne la méthode de classification non supervisée, nous avons opté pour la méthode des K-moyennes (Steinhaus, 1956, MacQueen et al., 1967) car elle est d'une part très simple à mettre en œuvre et elle cherche une partition sous la forme de cellules

de Voronoï bien adaptée à notre problème, contrairement à d'autres approches comme la classification spectrale (Von Luxburg, 2007). Le nombre de classes, noté N_p, est fixé comme une fraction de la taille de l'ensemble représentant l'espace des versions.

6.3.2 Recherche heuristique d'un plan d'expériences

a. Recherche arborescente de Monte-Carlo

L'algorithme UCT décrit en section 3.3.3.a. est adapté ici pour prendre en compte la nature de notre problème, en introduisant les notions de coût et de budget ainsi que les propriétés spécifiques de notre problème. On se réfère à la section 3.3.3 qui décrit les algorithmes de la famille MCTS de manière générale. Pour notre problème, la construction et le parcours d'arbres se définissent comme suit.
On se place tout d'abord sur le nœud racine de l'arbre, dont les nœuds-fils représentent les différents choix possibles pour la première expérience. On va parcourir l'arbre en le construisant simultanément. Un parcours d'arbre depuis la racine jusqu'à un nœud-feuille définit une séquence d'expériences. Au cours d'un tel parcours, à chaque nœud atteint, on choisit un nœud-fils selon la stratégie ϕ, comme celle décrite en section a. qui se fonde sur le critère UCB. Le choix de ce nœud-fils constitue une *action*. Si ce nœud n'existe pas encore dans l'arbre (i.e. s'il n'a encore jamais été exploré), on le crée au passage. Lorsqu'on arrive en fin de séquence, c'est-à-dire lorsque le coût de la séquence excède le

budget disponible ou lorsque l'horizon est atteint, on évalue l'utilité associée.

Algorithm 7: MCTS : Procédure de recherche arborescente

Input: Nombre de parcours d'arbre : N ; Ensemble d'expériences : $\mathcal{E}_n$; Espace des versions : $\mathcal{V}_n$; Budget intermédiaire : β ; Horizon : L ; Stratégie de sélection : ϕ ; $\mathcal{L}earner$; $\mathcal{R}eward$

Output: Arbre exploré : $tree$

$tree = \{root\}$;
for $t \in 0 \dots N$ **do**
  $\mathcal{A} \leftarrow \mathcal{E}_n$;
  $path = \{root\}$;
  while $b(path) < \beta \wedge |path| < L$ **do**
    $action \leftarrow \phi(tree, path, \mathcal{A})$;
    $node \leftarrow getNode(tree, path, action)$;
    if $node = \emptyset$ **then**
      /* Ajout d'un nœud fils associé à l'action sélectionnée */
      $node \leftarrow tree.addNode(path, action)$;
    end
    $\mathcal{A} \leftarrow \mathcal{A} \backslash \{action\}$; // Séquence d'expériences sans réplicat
    $path \leftarrow path \cup node$;
  end
  $\mathbf{s} \leftarrow getAction(tree, path, \mathcal{E}_t)$;
  $x_{\mathbf{s},t} \leftarrow \mathcal{R}eward(\mathbf{s}, \mathcal{V}_n, \mathcal{L}earner)$;
  /* Rétro-propagation de la récompense à travers le chemin parcouru et ses permutations */
  foreach $p \in permutation(path)$ **do**
    if $p \notin tree$ **then**
      $tree.addPath(p)$
    end
    /* Mise à jour des statistiques des nœuds visités */
    $nodes \leftarrow tree.getNode(p)$;
    foreach $node \in nodes$ **do**
      /* Cumul des récompenses */
      $tree.reward\{node\} \leftarrow tree.reward\{node\} + x_{\mathbf{s},t}$;
      /* Incrément du nombre de visites */
      $tree.visit\{node\} \leftarrow tree.visit\{node\} + 1$;
    end
  end
end
return $tree$

On effectue ensuite une phase de **rétro-propagation** en attribuant à chaque nœud de la séquence la valeur de cette utilité. Cette valeur pourra ainsi servir dans la fonction de stratégie ϕ pour les parcours futurs dans lesquels ces nœuds seraient potentiellement impliqués. Une caractéristique importante de notre algorithme est que l'on exploite l'invariance de l'utilité par permutation de l'ordre des expériences au sein d'une séquence donnée. On va donc propager l'utilité trouvée à tous les nœuds qui appartiennent à un chemin s'obtenant par permutation des expériences du chemin courant. Cela engendre un gain important en temps de convergence.

6.3.3 Algorithme de calcul d'utilité

Lors de la phase de simulation de l'algorithme de MCTS, l'algorithme calcule une utilité relative à la combinaison d'expériences associées au chemin parcouru. L'algorithme 8 calcule cette utilité.

Algorithm 8: $\mathcal{R}eward$: Algorithme générale de calcul d'utilité

Data: Données courantes disponibles : $\mathbf{Y}_{\mathbf{s}_n}$; Espace des versions : $\mathcal{V}(\mathbf{Y}_{\mathbf{s}_n})$; Ensemble d'expériences : $\mathcal{E}$; Ensemble des modèles : $(\mathcal{M}_e)_{e\in\mathcal{E}}$

Input: Critère d'optimalité : *criterion*, Combinaison d'expériences : $\mathbf{c}$

Output: Utilité de $\mathbf{c}$ selon le critère donné sachant $\mathbf{Y}_{\mathbf{s}_n}$: $x_{\mathbf{c}}$

`/* Initialisation */`

$\mathbf{Y}_{\mathbf{c}} \leftarrow \{\}$;

$\mathcal{M}_{\mathbf{c}} \leftarrow \{\}$;

`/* Tirage uniforme sur l'espace des versions (voir l'algorithme 6) */`

$\boldsymbol{\theta}^* \overset{unif}{\sim} \mathcal{V}(\mathbf{Y}_{\mathbf{s}_n})$;

`/* Simulation des données de chaque expérience dans c */`

foreach $e \in \mathbf{c}$ **do**

  $(f, h^e, \mathbf{u}^e, t_{0:K^e}, \mathbf{x}_0^e, \boldsymbol{\sigma}_{0:K^e}^e) \leftarrow \mathcal{M}_e.getInfo()$;

  $\mathbf{Y}_e = \{\}$;

  foreach $k \in 0 \dots K^e$ **do**

    `/* Simulation d'une expérience */`

    $\mathbf{x}_k^e \leftarrow f(\mathbf{x}_0^e, \mathbf{u}^e, t_k, \boldsymbol{\theta}^*)$;

    `/* Réalisation d'un bruit gaussien */`

    $\boldsymbol{\epsilon}_k^e \overset{unif}{\sim} \mathcal{N}(\mathbf{0}_{m_e}, \mathrm{diag}(\boldsymbol{\sigma}_k^e)^2)$;

    `/* Simulation des observations */`

    $\mathbf{y}_k^e \leftarrow h^e(\mathbf{x}_k^e, t_k) + \boldsymbol{\epsilon}_k^e$;

    $\mathbf{Y}_e \leftarrow \mathbf{Y}_e \cup \{\mathbf{y}_k^e\}$;

  end

  `/* Fusion des données simulées */`

  $\mathbf{Y}_{\mathbf{c}} \leftarrow \mathbf{Y}_{\mathbf{c}} \cup \mathbf{Y}_e$;

  `/* Fusion de modèle */`

  $\mathcal{M}_{\mathbf{c}} \leftarrow fusion(\mathcal{M}_{\mathbf{c}}, \mathcal{M}_e)$;

end

`/* Fusion de l'ensemble des modèles */`

$\mathcal{M}_{\mathbf{s}_n \cup \mathbf{c}} \leftarrow fusion(\mathcal{M}_{\mathbf{s}_n}, \mathcal{M}_{\mathbf{c}})$;

`/* Fusion des données réelles et des données simulées */`

$\mathbf{Y}_{\mathbf{s}_n \cup \mathbf{c}} \leftarrow \mathbf{Y}_{\mathbf{s}_n} \cup \mathbf{Y}_{\mathbf{c}}$;

$x_{\mathbf{c}} \leftarrow utility(\mathcal{M}_{\mathbf{s}_n \cup \mathbf{c}}, \mathbf{Y}_{\mathbf{s}_n \cup \mathbf{c}}, criterion)$;

return $x_{\mathbf{c}}$

L'algorithme $\mathcal{R}eward$ commence par choisir un vecteur "oracle" de paramètres, $\boldsymbol{\theta}^*$, tiré uniformément sur l'espace des versions courant $\mathcal{V}(\mathbf{Y}_{\mathbf{s}_n})$. À l'aide de ces valeurs, on génère

une réalisation de chaque expérience par simulation du modèle et génération d'une réalisation du bruit de mesure. Ensuite, l'algorithme combine les données simulées et les données réelles disponible à l'étape n de EDEN, et crée le modèle fusionné de la même manière que dans la phase d'estimation par fusion de modèles 6.2.a.. Enfin, la procédure *utility*() qui sera décrite dans la section suivante calcule l'utilité associée à l'ensemble des données en fonction du type de critère d'optimalité choisi (défini à l'aide de l'hyperparamètre *criterion*).

a. Utilité pour la minimisation d'incertitude

Nous avons présenté dans la section 4.2.2 les critères d'optimalité issus de la littérature sur les plans d'expériences optimaux. Ces critères permettent de mesurer l'utilité d'une expérience sur la base de l'inverse de la matrice d'information de Fischer ou de la matrice de covariance. Ces critères peuvent être appliqués à notre problème pour calculer l'utilité d'une réalisation d'un jeu de données au cours du calcul d'utilité. Nous proposons donc trois critères d'utilités mesurant l'incertitude associée à l'estimation à partir de la matrice de covariance.

Étant donné une combinaison d'expérience $\mathbf{c}$, notons $\hat{\boldsymbol{\theta}}|\mathbf{Y}_{\mathbf{s}_n \cup \mathbf{c}} = \tau(\mathbf{Y}_{\mathbf{s}_n \cup \mathbf{c}})$ l'estimé du paramètre $\boldsymbol{\theta}$ à partir de la fusion des données disponibles à la n-ième itération de EDEN et des données simulées issues de la combinaison d'expériences $\mathbf{c}$. Les trois critères d'utilité s'écrivent alors :

$$u_A(\mathbf{c}, \mathbf{Y}_{\mathbf{s}_n \cup \mathbf{c}}) = -\text{trace}\left(\text{Var}\left[\hat{\boldsymbol{\theta}}|\mathbf{Y}_{\mathbf{s}_n \cup \mathbf{c}}\right]\right) \tag{6.6}$$

$$u_D(\mathbf{c}, \mathbf{Y}_{\mathbf{s}_n \cup \mathbf{c}}) = -\det\left(\text{Var}\left[\hat{\boldsymbol{\theta}}|\mathbf{Y}_{\mathbf{s}_n \cup \mathbf{c}}\right]\right) \tag{6.7}$$

$$u_E(\mathbf{c}, \mathbf{Y}_{\mathbf{s}_n \cup \mathbf{c}}) = -\max\left(\Lambda(\text{Var}\left[\hat{\boldsymbol{\theta}}|\mathbf{Y}_{\mathbf{s}_n \cup \mathbf{c}}\right])\right) \tag{6.8}$$

b. Utilité pour la réduction du biais

Alors que les précédentes définitions d'utilité s'appuyaient essentiellement sur les propriétés de l'estimé, tenant compte des caractéristiques de la matrice de covariance, les fonctions d'utilité pour la réduction du biais font intervenir l'oracle $\boldsymbol{\theta}^*$. Ceci n'est bien sûr possible que dans le jeu simulé où l'on sera en mesure de calculer ces utilités dans différents *scenarii* : dans le monde réel, le vrai paramètre étant inconnu, on ne peut pas

vérifier la valeur d'utilité réellement obtenue.

$$u_{bias}(\mathbf{c}, \mathbf{Y}_{\mathbf{s}_n \cup \mathbf{c}}, \boldsymbol{\theta}^*) = 1 - \frac{\left\| \boldsymbol{\theta}^* - \mathbb{E}_{\mathbf{Y}_{\mathbf{s}_n \cup \mathbf{c}}}[\boldsymbol{\theta}] \right\|_2}{\left\| \boldsymbol{\theta}^* - \mathbb{E}_{\mathbf{Y}_{\mathbf{s}_n}}[\boldsymbol{\theta}] \right\|_2} \tag{6.9}$$

c. Utilité pour la réduction de l'erreur d'optimisation

L'estimation de paramètres à partir des données cachées et bruitées peut être vu comme un problème inverse mal posé. En effet, si on cherche à optimiser le modèle par rapport aux observations, il arrive que la solution dite optimale soit très éloignée de la réalité. Bien qu'il existe des techniques dites de régularisation qui jouent sur la complexité du modèle pour atténuer ce phénomène, on peut jouer sur le choix des données en employant une fonction d'utilité qui prenne cet effet considération. On propose donc une fonction d'utilité faisant intervenir la solution optimale. Le but étant de privilégier les expériences réduisant l'écart entre les vrais paramètres et les paramètres optimaux.

$$u_{opt}(\mathbf{c}, \mathbf{Y}_{\mathbf{s}_n \cup \mathbf{c}}, \boldsymbol{\theta}^*) = 1 - \frac{\left\| \boldsymbol{\theta}^* - \operatorname{argmin}_{\boldsymbol{\theta} \in \mathcal{V}} \ell(\mathcal{M}, \mathbf{Y}_{\mathbf{s}_n \cup \mathbf{c}}) \right\|_2}{\left\| \boldsymbol{\theta}^* - \operatorname{argmin}_{\boldsymbol{\theta} \in \mathcal{V}} \ell(\mathcal{M}, \mathbf{Y}_{\mathbf{s}_n}) \right\|_2} \tag{6.10}$$

Remarque : Dans l'équation (6.10) on suppose que la fonction de perte n'admet qu'un seul optimum global. En toute rigueur, rien ne garantit cette unicité, par conséquent pour la suite, on considéra toujours l'optimum global le plus éloigné du vrai paramètre pour avoir une estimation de l'utilité dans le pire des cas.

d. Utilité calculée par estimation de UKF

Le filtre de Kalman peut estimer à la fois la loi *a posteriori* des états cachés et les paramètres d'un modèle à espace d'états. Dans le cas d'un modèle non linéaire, on peut utiliser l'estimation par UKF pour calculer l'utilité via un des critères d'optimalité basés soit sur la réduction d'incertitude, soit sur la réduction du biais. L'algorithme 9 implémente le calcul d'utilité à l'aide d'UKF pour mesurer un critère d'optimalité relatif à l'ensemble des paramètres. On peut étendre cet algorithme pour évaluer un critère relatif aux états cachés, ou à une combinaison d'un sous-ensemble de paramètres et d'états cachés.

De même que dans la phase d'estimation à partir des données réelles (décrite dans la section a.), on applique l'algorithme d'estimation au modèle fusionné à partir des expériences déjà faites et des expériences issues de la phase de simulation de MCTS. Pour réduire le temps de calcul, on applique une seule itération, nous donnant ainsi pour chaque simulation, l'effet immédiat des données simulées sur la covariance et l'espérance. À la

fin de l'estimation par UKF, on calcule le critère d'optimalité donné par l'utilisateur à partir des espérances et des matrices de covariance calculées par le filtre de Kalman.

Algorithm 9: *utility* : Calcul de l'utilité à partir des estimations que fournit UKF

Data: L'estimation courante de $\boldsymbol{\theta}$: $\left(\hat{\boldsymbol{\mu}}_{\Theta_0}, \hat{\boldsymbol{\Sigma}}_{\Theta_0}\right)$; Critère d'optimalité : *criterion*
Algo : Estimation de paramètres : $\mathcal{UKF}$
Input: Oracle : $\boldsymbol{\theta}^*$; Modèle fusionné : $\mathcal{M}$; Données fusionnées : $\mathbf{Y}$
Output: Utilité : u
$(F, H, \mathbf{x}_0, \mathbf{u}, t_{0:K}, \boldsymbol{\Sigma}^{\mathcal{Y}}_{0:K}) \leftarrow getInfo(\mathcal{M})$;

`/* Mise à jour de l'a priori via une itération de UKF */`
$\left[\hat{\boldsymbol{\mu}}_{\Theta|\mathbf{Y}}, \hat{\boldsymbol{\Sigma}}_{\Theta|\mathbf{Y}}\right] \leftarrow \mathcal{UKF}(\mathbf{Y}, \hat{\boldsymbol{\mu}}_{\Theta_0}, \hat{\boldsymbol{\Sigma}}_{\Theta_0}, F, H, \mathbf{x}_0, \mathbf{u}, t_{0:K}, \boldsymbol{\Sigma}^{\mathcal{Y}}_{0:K})$;

`/* Calcul du critère d'optimalité */`
switch *criterion* **do**
  case *'A-optimality'*
    return $u_A = -\operatorname{Tr}(\hat{\boldsymbol{\Sigma}}_{\Theta|\mathbf{Y}})$
  end
  case *'D-optimality'*
    return $u_D = -\det(\hat{\boldsymbol{\Sigma}}_{\Theta|\mathbf{Y}})$
  end
  case *'E-optimality'*
    return $u_E = -\max(\{\lambda \in \Lambda(\hat{\boldsymbol{\Sigma}}_{\Theta|\mathbf{Y}})\})$
  end
  case *'Bias-reduction'*
    `/* Calcul du biais du posterior */`
    return $u_{bias} = 1 - \dfrac{\|\boldsymbol{\theta}^* - \hat{\boldsymbol{\mu}}_{\Theta_0}\|_2}{\|\boldsymbol{\theta}^* - \hat{\boldsymbol{\mu}}_{\Theta|\mathbf{Y}}\|_2}$
  end
endsw

e. Utilité calculée à l'aide de l'algorithme Nelder-Mead

En toute rigueur, l'utilité u_{opt} décrite précédemment nécessiterait l'emploi d'un algorithme d'optimisation globale, afin de déterminer la solution $\operatorname{argmin}_{\boldsymbol{\theta}\in\mathcal{V}} \ell(\mathcal{M}, \mathbf{Y}_{1:n+n_c})$. Cependant, en pratique, cette approche est bien trop coûteuse pour le calcul d'utilité. Afin de réduire les temps de calcul, la solution que nous proposons consiste à d'abord évaluer la fonction objectif sur le nombre fini de candidats de l'espace des versions, puis d'appliquer un algorithme d'optimisation globale à partir des meilleurs candidats. L'algorithme

Nelder-Mead peut être initialisé avec un simplexe : on initialise donc l'algorithme à l'aide des $d+1$ meilleurs candidats, où d est la dimension du vecteur de paramètres.

Algorithm 10: *utility* : Calcul de l'utilité à l'aide de la méthode de Nelder-Mead

Data: Espace des versions : $\mathcal{V}(\mathbf{Y}_{\mathbf{s}_n})$; Fonction de perte : ℓ ;

Input: Oracle subrogé : $\boldsymbol{\theta}^*$; Modèle fusionné : $\mathcal{M}_{\mathbf{s}_n}$; Données fusionnées : $\mathbf{Y}_{\mathbf{s}_n}$

Output: Utilité : u_{opt}

`/* Récupération de la meilleure solution courante */`

$\boldsymbol{\theta}^{opt}_{old} \leftarrow \text{argmin}_{\boldsymbol{\theta} \in \mathcal{V}(\mathbf{Y}_{\mathbf{s}_n})} \ell(\mathcal{M}_{\mathbf{s}_n}, \mathbf{Y}_{\mathbf{s}_n}, \boldsymbol{\theta})$;

`/* Création d'un simplexe à partir d'un tirage aléatoire sur l'espace des versions */`

$\boldsymbol{\Theta}_{\mathbf{init}} \leftarrow \{\boldsymbol{\Theta} \in \mathcal{V}(\mathbf{Y}_{\mathbf{s}_n})^{d+1} \ / \ \forall \boldsymbol{\theta} \in \boldsymbol{\Theta}, \boldsymbol{\theta} \sim \mathbf{Y}_{\mathbf{s}_n}\}$

`/* Appliquer l'algorithme Nelder-Mead */`

$\boldsymbol{\theta}^{opt}_{new} \leftarrow nelderMead(\ell, \mathcal{M}, \mathbf{Y}, \boldsymbol{\Theta}_{\mathbf{init}})$;

`/* Calcul de récompense */`

return $u_{opt} = 1 - \frac{\|\boldsymbol{\theta}^* - \boldsymbol{\theta}^{opt}_{new}\|_2}{\|\boldsymbol{\theta}^* - \boldsymbol{\theta}^{opt}_{old}\|_2}$

6.4 Discussion

6.4.1 Discrimination de modèles.

Dans certains travaux comme ceux de Kreutz and Timmer (2009), différents modèles candidats sont comparés à l'aide de critères classiques de type AIC (Aikaike Information Criterion). Les auteurs insistent sur le fait qu'un plan d'expériences optimal pour la discrimination de modèle ne le sera pas forcément pour l'estimation de paramètre et posent le problème de la définition d'un critère combiné intégrant ces deux objectifs. Ce type de problème pourrait tout à fait trouver sa place dans notre approche également, même si nous nous plaçons ici dans un cadre dans lequel la structure du modèle est donnée.

6.4.2 Prise en compte d'un budget limité.

Cette contrainte est rarement incluse dans les travaux ayant des objectifs similaires au nôtre, comme ceux de Galvanin et al. (2007), Asprey and Macchietto (2000), Kreutz and Timmer (2009) et Franceschini and Macchietto (2008). Dans ce cas, il s'agit simplement de trouver quelles expériences réaliser, en nombre fixé d'avance. Ceci correspond dans

notre approche à imposer un budget unitaire identique pour chaque expérience et à fixer l'horizon de recherche lors du parcours de l'arbre. Cette simplification est précieuse pour tester le fonctionnement des algorithmes mais l'introduction d'un coût différentiel par type d'expérience parait importante vis à vis du réalisme de l'approche et de son applicabilité pour venir en aide aux biologistes.

6.4.3 Anticipation des expériences à venir.

A notre connaissance, il n'existe pas dans la littérature d'approche de planification d'expériences permettant d'envisager une anticipation sur de futures expériences lors d'une recommandation. Du fait de la complexité des mesures utilités, les contraintes de temps imposent d'adopter une stratégie gloutonne, consistant à choisir une expérience parmi d'autres sans tester leur combinaison. Bien que l'anticipation des expériences à court ou moyen terme puisse être bénéfique à la planification d'expériences, ce type de stratégie ajoute une complexité trop importante à moins de réduire considérablement le nombre d'expériences et risquer d'écarter une combinaison d'expériences potentiellement utile sur le long terme.
Notre approche permet d'envisager une telle stratégie en fournissant une solution approchée au problème. L'idée est qu'au lieu de calculer précisément l'utilité de l'ensemble des séquences d'expériences possibles, on peut de manière itérative approcher ces utilités en privilégiant les scénarii les plus prometteurs. Ce type d'approche est permis grâce aux algorithmes de recherche arborescente (MCTS) et aux algorithmes de type bandit qui permettent une allocation efficace des ressources de calcul.

6.4 Synthèse

Dans ce chapitre, nous avons présenté EDEN, un méta-algorithme qui résout automatiquement le problème de planification séquentielle d'expériences pour l'estimation de systèmes dynamiques. A notre connaissance, il s'agit de la première application d'une approche d'apprentissage actif capable d'anticipation pour l'estimation de paramètres d'un système dynamique. EDEN repose essentiellement sur la construction, l'exploration et l'exploitation d'un arbre de recherche de Monte-Carlo à chaque étape de la recommandation d'expériences.

L'évaluation de l'utilité d'une expérience est réalisée en faisant appel d'une part, à des simulations du système dynamique et d'autre part, à un algorithme d'estimation des paramètres capable de traiter des données hétérogènes issues de perturbations de différents type. Nous avons proposé à titre d'exemple deux algorithmes, l'un fondé sur UKF et l'autre sur ESS. Dans le cas de l'algorithme UKF, nous proposons une méthodologie originale pour fusionner au mieux les données consistant à construire artificiellement un modèle fusionné. Pour chacun de ces deux algorithmes, nous avons défini la manière dont est construit un échantillon de l'espace des versions servant à la phase suivante de recommandation.

Dans le module de recommandation d'expériences, nous avons adapté l'algorithme MCTS pour prendre en compte le budget et le coût des différentes expériences. Le calcul de l'utilité d'un ensemble d'expériences associé à une feuille de l'arbre étant invariant par permutation, nous avons réduit la complexité du calcul de l'utilité en mettant en œuvre un mécanisme de transposition qui permet de propager une utilité à un nœud non visité.

Troisième partie

Résultats numériques

Chapitre 7

Illustration du fonctionnement de l'algorithme EDEN

Sommaire

« Que la stratégie soit belle est un fait, mais n'oubliez pas de regarder le résultat. »

Winston Churchill

7.1 Implémentation au sein des librairies ODESSA et EDEN

L'ensemble des résultats présentés dans ce chapitre ont été obtenus à l'aide de deux librairies MATLAB® conçues par nos soins dans un style orienté objet, ODESSA [1] et EDEN. Le logiciel ODESSA intègre un ensemble d'outils de modélisation, de simulation, d'estimation de paramètres et d'inférence de réseau développés à partir des travaux de cette thèse et également d'autres travaux voisins sur l'inférence de réseaux Fouchet et al. (2013), Lim (2015) réalisés au sein de l'équipe AROBAS du laboratoire IBISC de l'Université d'Evry. ODESSA permet de créer et de manipuler avec aisance les modèles d'équations différentielles ordinaires et les modèles à espace d'états. La simulation des EDO est accélérée par l'aide du solveur fortran SUNDIALS [2] (Hindmarsh et al., 2005) grâce à l'interface CVODE [3]/MEX [4] développée par Vanlier et al. (2012). Une interface objet a également été conçue pour prendre en charge la libraire EKF/UKF (Hartikainen et al., 2011) afin de simplifier la mise en œuvre de notre approche d'estimation de paramètres et d'états cachés par fusion de modèle (voir la section 6.2.a.). Les estimations par eSS sont réalisées par la librairie MEIGO [5] (Egea et al., 2014). Le logiciel EDEN repose sur ODESSA et intègre l'implémentation de la version actuelle du méta-algorithme EDEN. Les outils de recherche arborescente ont été entièrement développés pour ce problème ; quant aux stratégies de sélection, elles s'appuient en grande partie sur la librairie *pymabandit* (Garivier and Cappé, 2011, Kaufmann et al., 2012).

7.2 Contrôle optimal d'un réseau de signalisation

Avant d'aborder le problème du challenge DREAM qui a motivé nos travaux, nous nous intéressons au fonctionnement de l'algorithme EDEN sur un modèle dans lequel les expériences correspondent au choix d'une variable de contrôle en entrée, qui peut ainsi être visualisée.

7.2.1 Description du réseau JAK/STAT

JAK/STAT est un réseau de signalisation qui permet le transfert d'une information générée par des signaux chimiques à l'extérieur de la cellule vers l'intérieur pour déclencher sa réponse.

1. Ordinary Differential Equation and State Space model Analysis
2. SUite of Nonlinear and DIfferential/ALgebraic equation Solvers
3. C-language Variable-coefficients ODE solver
4. Matlab Extension File
5. Metaheuristics for systems biology and bIoinformatics Global Optimization

La présence de l'hormone Erythropoïétin (Epo) au niveau des tissus membranaires déclenche le signal du récepteur *EpoR* qui amorce alors la phosphorylation des protéines STAT-5 inactives (x_1) par l'intermédiaire des molécules tyrosine-kinases JAK. Les protéines STAT-5 phosphorylisées (x_2) peuvent alors se combiner entre elles pour former des dimères (x_3). Les dimères pénètrent alors dans le noyau où ils jouent le rôle de facteur de transcription.

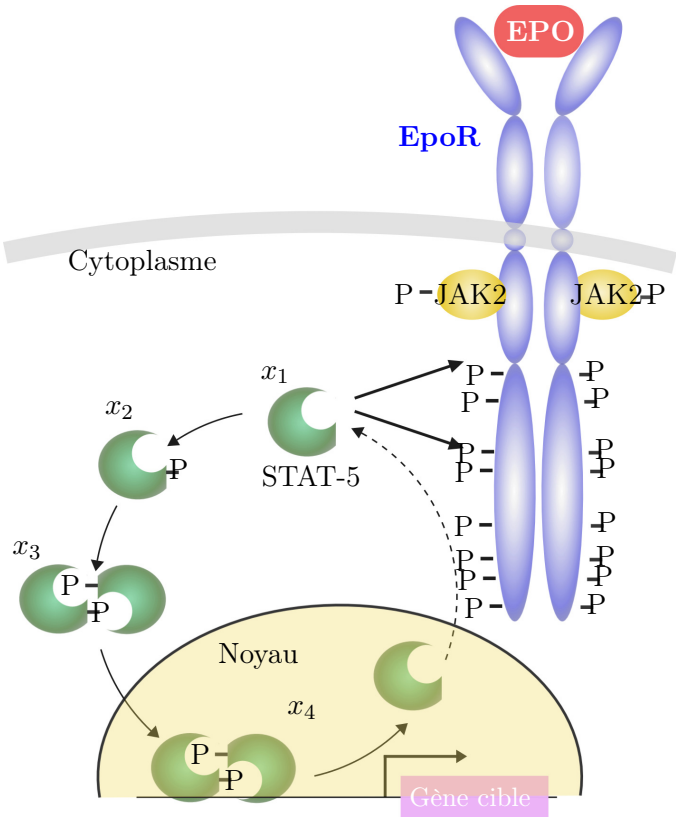

FIGURE 7.1 – Voie de transmission du signal JAK/STAT.
Source : Swameye et al. (2003).

7.2.2 Modélisation de JAK-JAK/STAT

Modèle d'évolution. Un modèle communément accepté pour représenter le système JAK/STAT est celui proposé par Swameye et al. (2003) que nous reprenons ici en (7.1). Ce modèle met en jeu les quantités x_1, x_2, x_3, et x_4 décrites précédemment et introduit un retard τ qui correspond au temps nécessaire au transport du dimère à l'intérieur du noyau. Il s'écrit donc sous la forme du système d'équations différentielles non-linéaires

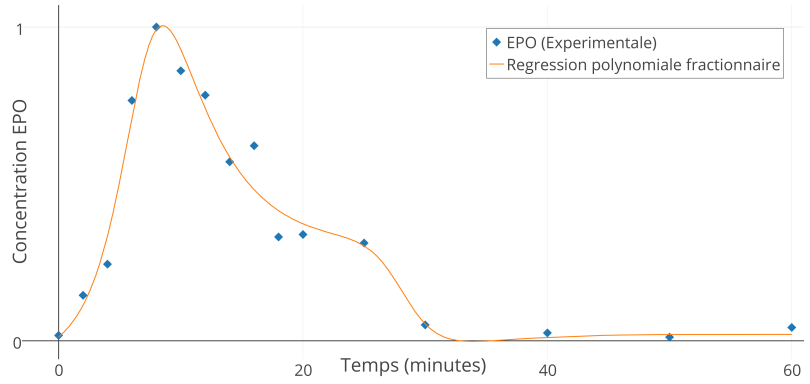

FIGURE 7.2 – Évolution de l'entrée *EpoR*

avec retard suivant :

$$\begin{aligned}
\dot{x_1}(t) &= -k_1.x_1(t).EpoR(t) + 2.k_4.x_3(t-\tau).\mathbb{1}_{\{t>\tau\}}(t) \\
\dot{x_2}(t) &= -k_2.x_2^2(t) + k_1.x_1(t).EpoR(t) \\
\dot{x_3}(t) &= -k_3.x_3(t) + \frac{1}{2}.k_2.x_2^2(t) \\
\dot{x_4}(t) &= -k_4.x_3(t-\tau).\mathbb{1}_{\{t>\tau\}} + k_3.x_3(t)
\end{aligned} \tag{7.1}$$

où k_1, k_2, k_3, k_4 sont les paramètres du modèle. La variable *EpoR* correspond à une donnée d'entrée dont l'évolution est contrôlée par la concentration de l'hormone Erythropoietin : la figure 7.2 montre l'évolution de la phosphorylation de l'EpoR. Les données expérimentales correspondent au jeu de données « DATA1_hall_inp » de Swameye et al. (2003). Comme les simulations nécessitent de pouvoir calculer l'entrée de manière continue, nous avons choisi de la représenter sous forme d'une fraction polynomiale (7.2) dont les variations sont régies par un vecteur de 8 paramètres **u** :

$$EpoR(t) = \frac{u_1.t^3 + u_2.t^2 + u_3.t + u_4}{t^4 + u_5.t^3 + u_6.t^2 + u_7.t + u_8} \tag{7.2}$$

La courbe de la figure 7.2 représente le résultat de la régression polynomiale fractionnaire sur ces données expérimentales, effectuée avec la méthode 'trust-region' de la toolbox 'curve-fitting' de Matlab avec une normalisation de la variable t, centrée autour de 18.8 et réduite avec un facteur de 18.

Modèle d'observation. Il n'est pas possible d'observer directement les quantités de toutes les espèces chimiques x_i impliquées, mais uniquement la quantité de molécules STAT-5 présentes dans le cytoplasme sous forme phosphorilisée (y_1) ou non phosphorilisée (y_2).

$$\begin{aligned}
y_1(t) &= x_2(t) + 2.x_3(t) \\
y_2(t) &= x_1(t) + x_2(t) + 2.x_3(t)
\end{aligned} \tag{7.3}$$

Planification d'expériences. L'objectif de notre plan d'expériences est de choisir la séquence d'actions à réaliser sur le récepteur Epo permettant d'obtenir une réponse du système adéquate pour l'estimation précise des inconnues du modèle, c'est-à-dire les paramètres du modèle d'évolution k_1, k_2, k_3, k_4 et l'état initial $x_1(t = 0)$ (les autres variables valant 0 à l'initialisation, cf tableau 7.2). On agira pour cela sur les coefficients du vecteur $\mathbf{u}$ de (7.2) en définissant pour chacun d'eux un domaine de variation donné dans le tableau 7.1. Ce domaine est défini par défaut à $[0\,;\,10]$ pour tous les coefficients mais cet intervalle est tronqué ou élargi de sorte à ce que l'espace de variation ainsi défini contienne les valeurs estimées et ne contienne, autant que possible, que des fonctions bien définies (en évitant en particulier que certains zéros de la fraction ne soient inclus dans l'intervalle de temps considéré).

Entrée	Valeur	Domaine de variation
u_1	$+0.050$	$[+0.05, 10.00]$
u_2	-0.040	$[-0.04, 10.00]$
u_3	-0.080	$[-0.08, 0.055]$
u_4	$+0.060$	$[+0.06, 10.00]$
u_5	-0.071	$[-0.34, 10.00]$
u_6	-0.465	$[-0.70, 10.00]$
u_7	-0.031	$[-0.03, 0.039]$
u_8	$+0.139$	$[+0.08, 10.00]$

TABLE 7.1 – Tableau des variables de contrôle du modèle JAK/STAT estimées à partir des données expérimentales figure 7.2 et domaines de variation associés choisis pour générer les 100 courbes correspondant aux 100 expériences possibles considérées pour le plan d'expériences.

Condition initial	Valeur	Domaine de recherche
x_1	5 nmol	$[0, 10]$
x_2	0 nmol	$\{0\}$
x_3	0 nmol	$\{0\}$
x_4	0 nmol	$\{0\}$

TABLE 7.2 – Tableau des conditions initiales du modèle JAK/STAT

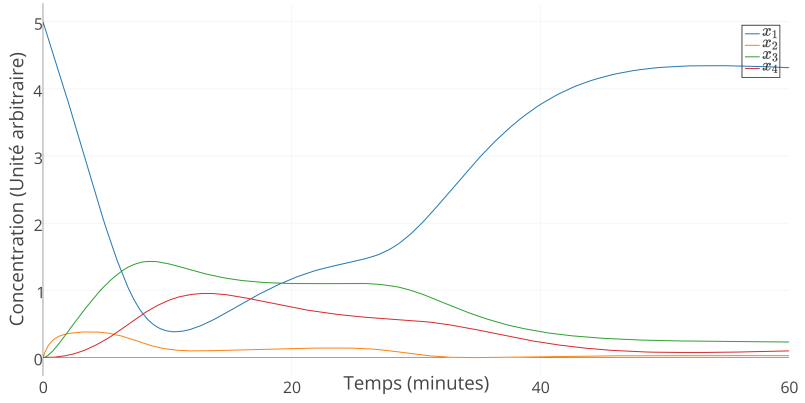

FIGURE 7.3 – Simulation des états cachés du système JAK/STAT à partir des vrais paramètres choisis pour le modèle.

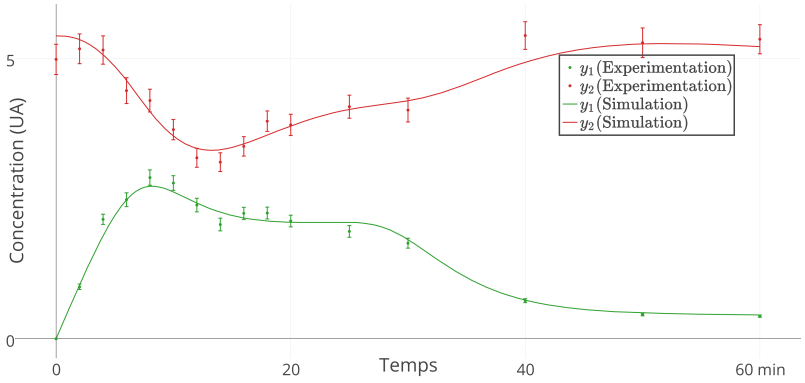

FIGURE 7.4 – Observations du système JAK/STAT. Données simulées et données bruitées (représentant des données expérimentales simulées avec un bruit multiplicatif gaussien)

Paramètre	Valeur	Domaine de recherche
k_1	0.5100 s^{-1}	$]0, 10]$
k_2	1.2800 s^{-1}	$]0, 10]$
k_3	0.1025 s^{-1}	$]0, 10]$
k_4	0.0900 s^{-1}	$]0, 10]$
τ	6 min	$\{6\}$

TABLE 7.3 – Tableau des paramètres du modèle JAK/STAT

On va tout d'abord générer un jeu de données expérimentales par simulation : il nous servira pour tester notre méthode. Les figures 7.3 et 7.4 montrent une simulation des comportements des états respectivement cachés et observables du système obtenu à partir des valeurs contenues dans les tableaux 7.1, 7.2 et 7.3. La figure 7.4 contient les données bruitées, représentant des données expérimentales simulées avec un bruit multiplicatif gaussien, qui sont utilisées pour l'estimation des quatre paramètres et de l'état initial de x_1.

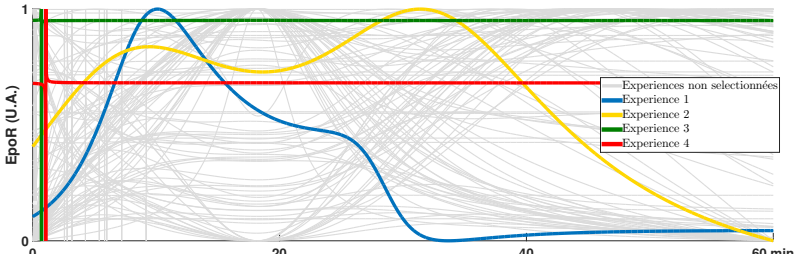

FIGURE 7.5 – Simulation des 100 fonctions d'entrées correspondant aux différentes expériences réalisables. L'expérience 1 est imposée dès le départ. Les expériences 2, 3 et 4 ont été choisies par l'algorithme EDEN. Les trajectoires grisées correspondent aux autres expériences, qui n'ont pas été retenues.

7.2.3 Configuration de l'algorithme EDEN

En utilisant le modèle paramétrique de la fonction d'entrée (équation 7.2) et en tirant les valeurs des 8 coefficients aléatoirement selon une loi uniforme dans leurs domaines de variation du tableau 7.1, nous avons généré 100 trajectoires. Elles représentent les différentes expériences qu'il sera possible de réaliser et parmi lesquelles on cherche à identifier celles qui permettront une estimation aussi précise que possible des paramètres du modèle.

La première expérience est imposée : c'est celle correspondant à la fonction d'entrée expérimentale (i.e. celle correspondant aux valeurs des coefficients estimées, tableau 7.1). Nous avons ensuite appliqué l'algorithme EDEN pour décider des expériences à réaliser pour réduire l'incertitude sur la condition initiale $x_1(t = 0)$ ainsi que les paramètres k_1, k_2, k_3 et k_4. Le tableau 7.4 reporte les données de configuration de l'algorithme EDEN. On se donne un horizon maximal de 3 expériences à choisir ainsi successivement. On s'autorise à répliquer les expériences : on obtient donc un nombre maximal de nœuds dans l'arbre des expériences de $100^3 = 10^6$ lors de la détermination de la première expérience. Une fois celle-ci fixée, il n'y a plus à explorer que des séquences de 2 puis 1 expérience(s), ce qui diminue à chaque fois d'un facteur 100 le nombre de nœuds associés.

La figure 7.5 montre l'allure des fonctions d'entrées $EpoR(t)$ possibles (en grisé) et celles des 3 fonctions sélectionnées par l'algorithme EDEN (la première étant imposée). Ces trajectoires sont générées via l'équation (7.2). On note que les fonctions correspondant aux expériences 3 et 4 ont des variations très abruptes (mais sont continues).

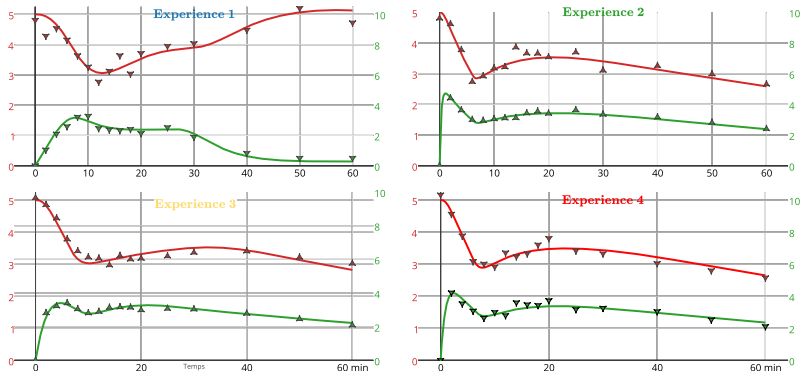

FIGURE 7.6 – Adéquation du modèle estimé aux données observées de chaque expérience. Les petites rectangles représentent les données d'observations relatives à chacune des expériences réalisées. Les trajectoires optimales estimée à partir des données disponibles aux différents stades du plan d'expériences séquentiel. Les courbes vertes (resp. rouges) simulent le comportement de la quantité y_1 (resp. y_2)

Hyperparamètre		Itération 1	Itération 2	Itération 3
Nombre total d'expériences :	$\|\mathcal{E}\|$	100	100	100
Taille de l'espace des version :	N_{VS}	1000	1000	1000
Seuil de rejet des solutions :	λ	1.1	1.1	1.1
Budget/profondeur d'arbre :	$B = T$	3	2	1
Nombre de nœuds maximum :	$\|tree\|$	10^6	10^4	100
Nombre de parcours d'arbre :	N	10^6	10^5	$5{\cdot}10^4$
Stratégie de sélection :	ϕ	$UCB1(C_p = \frac{1}{\sqrt{2}})$		
Fonction de recommandation :	ψ	Robust Child [6]		
Fonction de récompense :	R	u_A		

TABLE 7.4 – Configuration de l'algorithme EDEN pour le problème JAK/STAT. Le nombre d'expériences ne change pas car on autorise la réplication d'expériences.

7.2.4 Estimation des paramètres du modèle de JAK/STAT

La figure 7.6 montre les résultats d'estimation pour chaque jeu de données acquis. Les trajectoires sont optimisées à l'aide du test du Chi2 sur l'ensemble des données disponibles à chaque étape.

Les valeurs des paramètres utilisés pour simuler les trajectoires de la figure 7.6 sont reportées dans le tableau 7.5, ainsi que la distance quadratique moyenne entre les valeurs estimées et les 'vraies' valeurs, les cibles. Il faut toutefois noter que l'algorithme d'optimisation utilisé (eSS) n'offre aucune garantie quant à l'obtention de l'optimum global.

6. on rappelle que la stratégie *Robust Child* consiste à recommander la séquence d'expériences associés au chemin le plus visité de l'arbre (voir section 3.3.3)

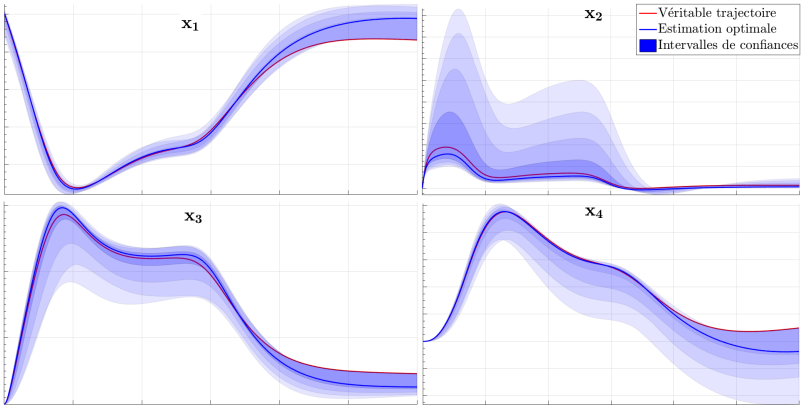

FIGURE 7.7 – Estimation des états cachées du système JAK/STAT. Les régions en bleu représentent les intervalles d'incertitude obtenus à chaque ajout d'expériences en prenant le minimum et le maximum des trajectoires générées avec toutes les solution contenues dans l'espace des versions.

	$x_1(0)$	k_1	k_2	k_3	k_4	rmse
Expérience 1	4.075	0.532	0.742	0.120	0.108	0.069
Expériences 1 à 2	4.073	0.467	0.714	0.122	0.109	0.080
Expériences 1 à 3	4.815	0.457	0.944	0.103	0.091	0.030
Expériences 1 à 4	5.011	0.465	1.430	0.100	0.088	0.005
Cible	5.000	0.510	1.280	0.103	0.090	

TABLE 7.5 – Liste des meilleures solutions obtenues à partir de chaque jeu de données. La dernière colonne affiche l'erreur quadratique moyenne entre les paramètres estimés et les paramètres cibles (dernière ligne du tableau).

Pour démontrer les performances du plan expérience séquentiel pour l'amélioration de la qualité d'estimation des états, nous avons tracé sur la figure 7.7 l'évolution des intervalles d'« incertitude » au fur et à mesure que des expériences sont ajoutées. Ces intervalles sont obtenus en simulant toutes les solutions contenues dans l'échantillon de l'espace des versions et en prenant le maximum et le minimum des trajectoires obtenues. On voit que l'ajout d'expériences, en apportant plus d'information, augmente la précision avec laquelle les états cachés peuvent être estimés. La courbe en bleu foncé ('estimation optimale') correspond à celle simulée avec les paramètres estimés une fois toutes les données acquises. Elle diffère de la véritable trajectoire, en rouge, en particulier du fait du bruit imposé aux données.

Une autre façon de visualiser l'amélioration de la précision sur l'estimation apportée par les expériences successives est représentée en figure 7.8. Cette figure représente les projections des hypothèses constituant l'espace des versions sur toutes les combinaisons possibles de 2 dimensions de cet espace (qui est à 5 dimensions : 4 paramètres et un état initial). Les différentes couleurs représentent les différentes étapes de l'algorithme, chaque étape

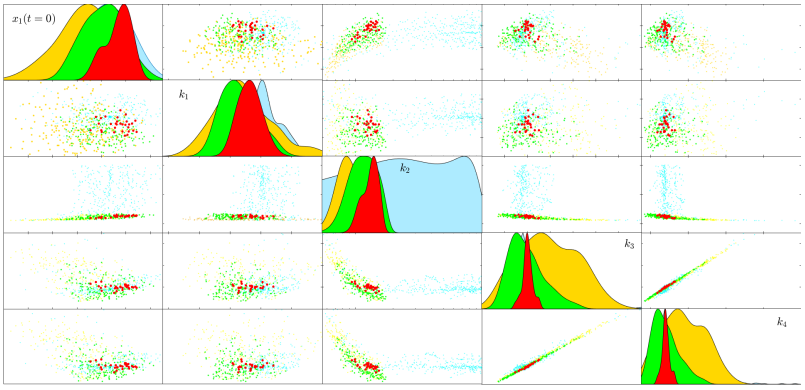

FIGURE 7.8 – Répartition des solutions de l'espace des versions à différentes étapes de l'algorithme EDEN. Chaque couleur correspond à une étape, c'est-à-dire à l'ajout des données d'une expérience supplémentaire. Le graphe de la case (1, 2) correspond par exemple à la projection de l'espace des versions sur les dimensions correspondant aux paramètres $x_1(t = 0)$ et k_2. Les éléments diagonaux représentent les densités normalisées estimées à partir des solutions de l'espace des versions selon chaque composante $x_1(0), k_1, k_2, k_3, k_4$.

étant définie par l'ajout d'une expérience supplémentaire. La diagonale contient des fonctions de densités estimées à partir de la population de l'espace des versions : on peut noter la diminution de leur support, qui représente la meilleure précision avec laquelle ces paramètres sont estimés. On peut ainsi identifier l'effet de certaines expériences : par exemple, l'expérience 4 est surtout utile pour réduire l'incertitude sur l'estimation des paramètres k_3 et k_4, tandis que l'expérience 2 a un fort effet sur l'estimation du paramètre k_2. On peut remarquer aussi que les projections sur les dimensions (k_3, k_4) génèrent des nuages de points diagonaux : ceci est cohérent avec l'article initial de Swameye et al. (2003) qui remarque qu'il faut imposer que $k_3 = k_4$ pour ajuster le modèle aux données expérimentales. On retrouve ici cette condition sans l'avoir imposée.

Enfin, la figure 7.9 représente la diminution du logarithme de l'erreur moyenne d'estimation, au fur et à mesure de l'ajout de données provenant de nouvelles expériences. On voit que les barres d'incertitudes diminuent de largeur, ce qui signifie que l'espace des versions se réduit : on lève ainsi progressivement (mais en partie seulement, même après ces 4 expériences) les non identifiabilités pratiques liées au manque de données.

7.2.5 Comparaison entre les approches parallèle et séquentielle

Bien qu'EDEN soit présenté comme un algorithme de planification séquentielle d'expériences, on peut également l'utiliser pour choisir un groupe d'expériences. Il suffit pour cela de sélectionner la liste des expériences correspondant au chemin le plus exploré de l'arbre, qui correspond à une solution approchée du problème (5.13). Là où, dans l'approche séquentielle, on ne recommande que la première expérience, on conservera ici la

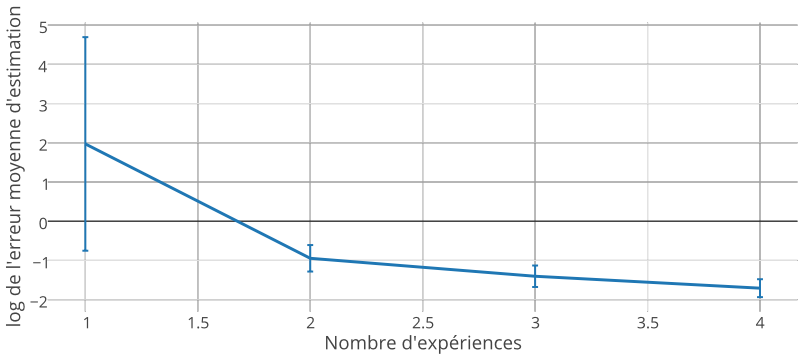

FIGURE 7.9 – Évolution de l'erreur d'estimation des variables d'intérêts du système JAK/STAT. Les barres d'incertitude correspondent aux intervalles $[log(m) - \sigma; log(m) + \sigma]$ où m est l'erreur moyenne d'estimation.

séquence complète : sachant que l'ordre n'a pas d'influence du fait de l'invariance par permutation, cela revient à proposer un groupe d'expériences qui pourront être réalisées en parallèle. Cependant, approcher cette solution demande plus d'itérations que d'approcher la meilleure expérience (5.14). Nous sommes donc partis de l'arbre calculé à partir des premières estimations, puis nous obligeons UCT à exploiter le meilleur bras en posant $C_{root} = 0$ (l'hyperparamètre d'UCB à la racine). On procède de la même manière pour les prochaines expériences jusqu'à approcher suffisamment la dernière expérience. Le tableau 7.6 montre les résultats obtenus par ces deux approches.

	RMSE			
Approche	Itération 0	Itérations		
Séquentielle	3.92±2.72	0.5192±0.34	0.38±0.27	0.31±0.23
Parallèle	3.92±2.72	0.45±0.30		

TABLE 7.6 – Comparaison entre les approches parallèle et séquentielle en terme d'erreurs d'estimation. La première ligne en montre la diminution au fur et à mesure que des expériences sont effectuées et apportent de l'information : elle correspond aux valeurs représentées sur la figure 7.9. La deuxième représente les résultats de l'approche parallèle : partant d'une même situation initiale (itération 0), on effectue cette fois-ci les 3 expériences en même temps.

Comme on pouvait s'y attendre, l'approche séquentielle offre de meilleurs résultats, puisqu'au bout de la deuxième expérience le plan séquentiel donne une meilleure estimation que les trois expériences de l'approche parallèle. Ceci est dû au fait que les expériences choisies par une approche séquentielle prennent en compte plus d'informations, puisque de nouvelles données sont acquises et analysées à chaque itération. Néanmoins, dans la réalité, on n'a pas nécessairement le choix entre ces deux stratégies : il est donc important de proposer une méthodologie pour ces deux cas d'application.

7.2.6 Étude de l'influence de l'horizon sur l'utilité des plans d'expériences séquentielles

Pour mettre en évidence l'intérêt que présente l'anticipation des futures expériences pour la planification séquentielle, nous avons réalisé une série de simulations de plan d'expériences en faisant varier l'horizon de 1 à 3 expériences. Les résultats indiquent dans deux cas sur trois un score maximal pour la stratégie avec un horizon 3 (voir le tableau 7.7). Bien entendu, le facteur chance peut jouer un rôle important car dans l'algorithme les expériences sont choisies de manière à maximiser l'espérance de l'utilité.

Utilité	$-log(-u_A)$			$-log(-u_D)$			$-log(-u_E)$		
Budget	3	2	1	3	2	1	3	2	1
Horizon 1	8.672	8.690	8.670	62.655	63.241	63.726	9.205	9.212	**9.205**
Horizon 2	8.690	8.677	8.685	62.623	62.934	63.724	9.219	9.205	9.193
Horizon 3	8.686	8.685	**8.698**	62.965	63.457	**63.876**	9.210	9.204	9.199

TABLE 7.7 – Comparaison de l'influence de l'horizon du MDP sur l'utilité du plan d'expériences. Un score élevé indique un jeu d'expériences très utile. Les utilités sont basées les critères d'optimalités A,D et E. Chaque sous-tableau montre l'évolution des scores en fonction du budget restant (c'est-à-dire du nombre d'expériences restant, puisque les coûts sont unitaires

7.3 Présentation challenge DREAM 7 - *Network Topology and Parameter Inference Challenge*

Les compétitions **DREAM** [7] (**D**ialogue for **R**everse **E**ngineering, **A**ssessments and **M**ethods (Stolovitzky et al., 2007)) forment une série de compétitions internationales visant à résoudre un certain nombre de problèmes ayant trait à la biologie des systèmes. Depuis 2002, plusieurs sujets sont proposés chaque année avec pour objectif d'amener les participants à tester et combiner des méthodes existantes ou à en créer de nouvelles pour ensuite les comparer et établir ainsi des références pour la communauté scientifique (Marbach et al., 2010, 2012, Prill et al., 2011, Meyer et al., 2014). Les compétitions DREAM6 (2011) et DREAM7 (2012) abordent la question de la planification d'expériences pour l'estimation de paramètres d'équations différentielles ordinaires (Meyer et al., 2014).

7. voir le site dreamchallenges.org

7.3.1 Description du modèle

On considère l'un des modèles proposés dans le cadre de DREAM7. La figure 7.10 donne la représentation de sa structure sous forme de réseau de régulation génique. Il s'agit d'un réseau ARNm-protéines à 9 gènes. La dynamique du réseau de régulation est régie par le système d'équations différentielles ordinaires décrit en (7.4). Le vecteur d'état $\mathbf{x}$ du modèle contient les variables de concentration des protéines et des ARNm associés à chaque gène $[r_1, \ldots, r_9, p_1, \ldots, p_9]$.

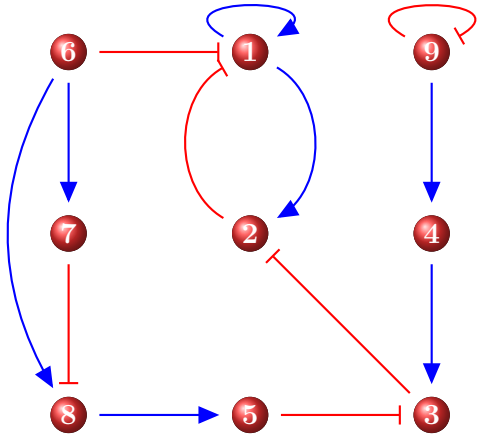

FIGURE 7.10 – Réseau de régulation de DREAM7, Modèle 1. Les flèches bleues (resp. rouges) représentent les régulations activatrices (resp. inhibitrices).

$$
\begin{cases}
\dot{r_1} &= \gamma_1 . \left[\frac{{p_1}^{h11}}{{K_{11}}^{h11} + {p_1}^{h11}} \right] . \left[\frac{{p_2}^{h21}}{{K_{21}}^{h21} + {p_2}^{h21}} \right] . \left[\frac{{K_{61}}^{h61}}{{K_{61}}^{h61} + {p_6}^{h61}} \right] - r_1 \\
\dot{r_2} &= \gamma_2 . \left[\frac{{p_1}^{h12}}{{K_{12}}^{h12} + {p_1}^{h12}} \right] . \left[\frac{{K_{32}}^{h32}}{{K_{32}}^{h32} + {p_3}^{h32}} \right] - r_2 \\
\dot{r_3} &= \gamma_3 . \left[\frac{{p_4}^{h43}}{{K_{43}}^{h43} + {p_4}^{h43}} \right] . \left[\frac{{K_{53}}^{h53}}{{K_{53}}^{h53} + {p_5}^{h53}} \right] - r_3 \\
\dot{r_4} &= \gamma_4 . \left[\frac{{p_9}^{h94}}{{K_{94}}^{h94} + {p_9}^{h94}} \right] - r_4 \\
\dot{r_5} &= \gamma_5 . \left[\frac{{p_8}^{h85}}{{K_{85}}^{h85} + {p_8}^{h85}} \right] - r_5 \\
\dot{r_6} &= \gamma_6 - r_6 \\
\dot{r_7} &= \gamma_7 . \left[\frac{{p_6}^{h67}}{{K_{67}}^{h67} + {p_6}^{h67}} \right] - r_7 \\
\dot{r_8} &= \gamma_8 . \left[\frac{{p_6}^{h68}}{{K_{68}}^{h68} + {p_6}^{h68}} \right] . \left[\frac{{K_{78}}^{h78}}{{K_{78}}^{h78} + {p_7}^{h78}} \right] - r_8 \\
\dot{r_9} &= \gamma_9 . \left[\frac{{K_{99}}^{h99}}{{K_{99}}^{h99} + {p_9}^{h99}} \right] - r_9 \\
\dot{p_i} &= \rho_i . r_i - k_p . p_i, \ \forall i \in \{1, \ldots, 9\} \\
r_i(0) &= 0, \ \forall i \in \{1, \ldots, 9\} \\
p_i(0) &= 1, \ \forall i \in \{1, \ldots, 9\}
\end{cases}
\tag{7.4}
$$

Le système d'équations (7.4) fait intervenir 45 paramètres inconnus contenus dans le vecteur $\boldsymbol{\theta} = [\gamma_1 \dots \gamma_9, \rho_1 \dots \rho_9, K_{11} \dots K_{99}, h_{11} \dots h_{99}, k_p]$. L'ensemble des paramètres à estimer avec les valeurs réelles correspondantes (« vraies » valeurs) est représenté dans le tableau (7.8). Lors du challenge, seule la valeur de la constante de dégradation k_r en ARN messager est donnée. Celle-ci vaut 1 $[s^{-1}]$ pour l'ensemble des espèces chimiques mises en jeu.

Taux de dégradation de l'ARNm (s^{-1})		
0	k_r	1
Force du répresseur ($nM.s^{-1}$)		
1	ρ_1	2.1
2	ρ_2	1.5
3	ρ_3	5
4	ρ_4	1.1
5	ρ_5	0.8
6	ρ_6	5
7	ρ_7	5
8	ρ_8	5
9	ρ_9	5
coefficient de Hill (sans unité)		
10	h_{11}	2
11	h_{12}	2
12	h_{32}	2
13	h_{21}	2
14	h_{43}	2
15	h_{78}	4
16	h_{53}	2
17	h_{61}	2
18	h_{94}	2
19	h_{68}	2
20	h_{99}	2
21	h_{67}	2
22	h_{85}	4

Taux de dégradation des protéines (s^{-1})		
23	k_p	0.1
Force du promoteur ($nM.s^{-1}$)		
24	γ_1	0.67
25	γ_2	1.42
26	γ_3	5
27	γ_4	1.5
28	γ_5	1.3
29	γ_6	1
30	γ_7	0.8
31	γ_8	1.5
32	γ_9	3.77
Constante de dissociation (nM)		
33	K_{11}	2.3432
34	K_{12}	1.342
35	K_{32}	1.774
36	K_{21}	0.262
37	K_{43}	0.568
38	K_{78}	2.6
39	K_{53}	1.54
40	K_{61}	0.12
41	K_{94}	1.13
42	K_{68}	0.2
43	K_{99}	1.4
44	K_{67}	2
45	K_{85}	0.5

TABLE 7.8 – Tableau des paramètres à estimer pour le modèle 1 de DREAM et leurs vraies valeurs.

La figure 7.11 montre l'évolution temporelle des niveaux d'expression de chaque gène et des concentrations de protéines qui leurs sont associées obtenues en simulant le modèle 1

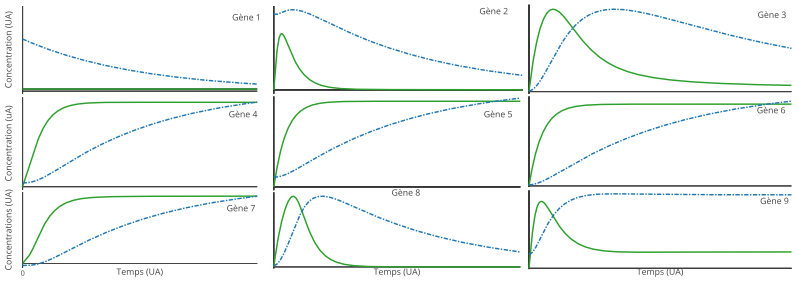

FIGURE 7.11 – Allure des niveaux d'expression simulés de chaque gène du réseau 1 du challenge DREAM 7 et dynamiques des concentrations en protéines associées. Les courbes en vert représentent les concentrations en ARN messagers et les courbes en pointillés bleu représentent les concentrations en protéines.

sans perturbation à partir des vrais paramètres. On voit par exemple que le gène 1 n'est jamais activé, ce qui fait que la concentration de sa protéine ne peut que diminuer en se dégradant.

7.3.2 Description et coût des expériences réalisables

En plus du jeu de données initial, fourni gratuitement et correspondant à l'observation des concentrations en ARN messager du système dit « *wildtype* » (non perturbé), on dispose d'un budget de 10000 crédits, utilisable pour l'achat d'expériences. Une expérience consiste ici en un couple (perturbation, mesures) formé d'une perturbation à appliquer au réseau et un type de variable à mesurer avec une résolution donnée.

a. Mesures possibles.

Les différentes mesures sont rendues possibles par les techniques présentées en section 7.3.2 et sont rassemblées dans le tableau 7.9. Les mesures d'expression des gènes par puces à ADN (« Micro-array ») fournissent les dynamiques des concentrations en ARNm pour tous les gènes : une haute résolution peut être demandée à un coût double de celui d'une basse résolution. L'expérience la moins coûteuse est la mesure des concentrations de protéines (« Fluorescent protein fusion »), mais elle ne fournit les profils dynamiques de concentrations que pour 2 protéines seulement. L'expérience la plus chère est largement celle fondée sur le retard à la migration sur gel (« Gel-shift assay »). Dans le cadre d'une modélisation de l'interaction ADN-protéine par des fonctions Hill (équations 1.3 et 1.4), cette expérience est supposée pouvoir être effectuée pour n'importe laquelle des interactions du réseau (mais une seule d'entre elles à la fois) et permet concrètement de révéler la valeur des paramètres K et h associés à la régulation ciblée.

Mesure	Coût	Description
Puces à ADN	500	Trajectoires en basse résolution de tous les ARNm
	1000	Trajectoires en haute résolution de tous les ARNm
Protéine de fusion fluorescente	400	Trajectoires d'un couple en protéines
Retard de migration sur gel	1600	Coefficient de dissociation (K_{ij}) et coefficient de (h_{ij}) d'un facteur de transcription

TABLE 7.9 – Tableau des mesures réalisables sur le modèle 1 de DREAM 7

b. Perturbations possibles du réseau et leur modélisation.

L'ensemble des perturbations qu'il est possible d'appliquer est présenté dans le tableau 7.10. Il peut s'agir (i) de l'élimination totale du gène ('knock-out' ou KO), ce qui supprime à la fois la production d'ARNm and et des protéines associés, (ii) d'une réduction de l'expression d'un gène par l'intermédiaire de petits ARN interférents (pARNi ou 'small interfering RNA' (siRNA)), ici supposée correspondre à une diminution de la production d'ARNm d'un facteur 10, ou (iii) une réduction de l'activité des ribosomes pour la synthèse de protéines, là encore d'un facteur 10. Chacune de ces perturbations a un coût fixé, supposé indépendant du gène auquel elles sont appliquées.

Perturbation	Coût	Description
Knock-Out	800	Supprime les termes de production des ARNm et protéines
Knock-Down	350	Utilise les pARNi pour décupler la dégradation de l'ARNm
Rbs-Down	450	Réduit l'activité des ribosomes pour la synthèse de protéines

TABLE 7.10 – Tableau des perturbations réalisables sur le modèle 1 de DREAM 7

Comme chacun de ces trois types de perturbations peut cibler l'ensemble des 9 gènes, on dénombre 27 perturbations réalisables. Le système d'équations différentielles en (7.4) est modifié en incorporant des variables de contrôle, de sorte à pouvoir utiliser un unique modèle pour simuler l'ensemble de ces perturbations. Le système (7.5) montre le nouveau modèle, obtenu en ajoutant des coefficients multiplicateurs qui permettent de perturber le modèle. On note $\mathbf{u}$ la variable de contrôle globale $\mathbf{u} = [u_{kd_1} \dots u_{kd_i} .. u_{ko_i} .. u_{rbs_i} \dots u_{rbs_9}]$ de taille 27. Cette variable est toujours constante à travers le temps mais prend 27 valeurs

possibles.

$$\begin{cases} \dot{r_1} &= u_{ko_1}.\gamma_1.g^+_{1,1}(x_{p_1}).g^-_{1,2}(x_{p_1}).g^-_{1,6}(x_{p_1}) - u_{kd_1}.r_1 \\ \dot{r_2} &= u_{ko_2}.\gamma_2.g^+_{2,1}(x_{p_2}).g^-_{2,3}(x_{p_2}) - u_{kd_2}.r_2 \\ \dot{r_3} &= u_{ko_3}.\gamma_3.g^+_{3,4}(x_{p_3}).g^-_{3,5}(x_{p_3}) - u_{kd_3}.r_3 \\ \dot{r_4} &= u_{ko_4}.\gamma_4.g^+_{4,9}(x_{p_4}) - u_{kd_4}.r_4 \\ \dot{r_5} &= u_{ko_5}.\gamma_5.g^+_{5,8}(x_{p_5}) - u_{kd_5}.r_5 \\ \dot{r_6} &= u_{ko_6}.\gamma_6 - u_{kd_6}.r_6 \\ \dot{r_7} &= u_{ko_7}.\gamma_7.g^+_{7,6}(x_{p_7}) - u_{kd_7}.r_7 \\ \dot{r_8} &= u_{ko_8}.\gamma_8.g^+_{8,6}(x_{p_8}).g^-_{8,7}(x_{p_8}) - u_{kd_8}.r_8 \\ \dot{r_9} &= u_{ko_9}.\gamma_9.g^-_{9,9}(x_{p_9}) - u_{kd_9}.r_9 \\ \dot{p_i} &= u_{rbs_i}.u_{ko_i}\rho_i.r_i - k_p.p_i,\ \forall i \in \{1,\dots,9\} \\ r_i(0) &= 0,\ \forall i \in \{1,\dots,9\} \\ p_i(0) &= 1,\ \forall i \in \{1,\dots,9\} \end{cases} \tag{7.5}$$

Le comportement du système *wildtype* (non perturbé) est obtenu en fixant tous ces coefficients à 1 ($\forall i, \mathbf{u}[i] = 1$). Un *knock-out* sur le gène i est simulé en affectant la valeur 0 au coefficient u_{ko_i}. Un *knock-down* sur le gène i est simulé en affectant la valeur 10 au coefficient u_{kd_i}. Un *rbs-down* sur le gène i est simulé en affectant la valeur $\frac{1}{10}$ au coefficient u_{rbs_i}.

c. Critères d'évaluation

Dans DREAM7, les compétiteurs sont évalués selon deux critères. Le premier critère mesure la qualité des paramètres estimés $\hat{\boldsymbol{\theta}}$ par rapport aux vrais paramètres $\boldsymbol{\theta}^*$:

$$D_\Theta(\hat{\boldsymbol{\theta}}, \boldsymbol{\theta}^*) = \frac{1}{n_{\boldsymbol{\theta}}} \sum_{i=1}^{n_{\boldsymbol{\theta}}} \left(\ln \left(\frac{\hat{\theta}_i}{\theta^*_i} \right) \right)^2 \tag{7.6}$$

où $n_{\boldsymbol{\theta}} = 45$ est la dimension du vecteur de paramètres.

Le second critère évalue le pouvoir de prédiction du modèle obtenu une fois les paramètres estimés :

$$D_\mathcal{Y}(\hat{\boldsymbol{\theta}}, \boldsymbol{\theta}^*) = \frac{1}{3(|T| - 11)} \sum_{k=3,5,8} \sum_{t_i \in T} \frac{(p_k(\hat{\boldsymbol{\theta}}, t_i) - p_k(\boldsymbol{\theta}^*, t_i))^2}{\sigma_b^2 + \sigma_s^2 \cdot p_k(\boldsymbol{\theta}^*, t_i)^2} \tag{7.7}$$

Ce dernier critère évalue la performance du modèle pour la prédiction de trois protéines indexées par $k = 3, 5, 8$ et en appliquant 3 perturbations simultanées, qu'on modélise par les transformations sur les paramètres suivantes :

- taux de production en protéine p_5 multiplié par 5 ($\gamma_5 \leftarrow 5 \cdot \gamma_5$).
- taux de production en protéine p_3 multiplié par 2 ($\gamma_3 \leftarrow 2 \cdot \gamma_3$).
- taux de dissociation de l'auto-inhibition du gène 9 divisé par 10 ($K_{99} \leftarrow 0.1 \cdot K_{99}$).

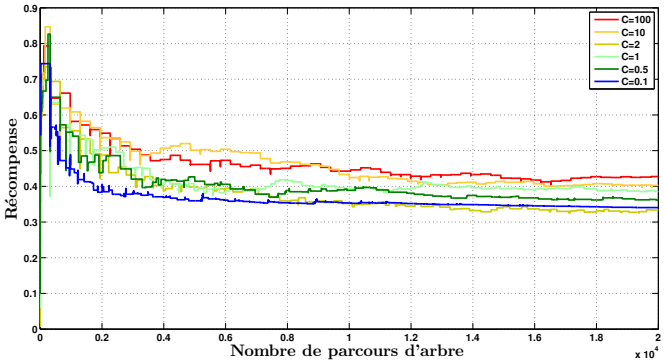

FIGURE 7.12 – Évolution des récompenses associées aux expériences recommandées en fonction de différentes valeurs de compromis *exploration-exploitation* pour un horizon de 1

Le terme $\sigma_b = 0.1$ est l'écart-type de la composante additive du bruit de mesure. Le terme $\sigma_s = 0.2$ est le facteur multiplicatif de la composante du bruit dépendante du signal. Les prédictions sont calculées sur l'intervalle de temps de $t_0 = 0$ à $20s$ par pas de $\Delta t = 0.5s$, soit $|T| = 41$. Cependant, les conditions initiales étant fournies aux concurrents, les 10 premiers instants ne sont pas pris en compte dans la prédiction : c'est la raison pour laquelle la somme est indexée à partir de la 11ème itération.

$$R(\mathbf{c}, \hat{\boldsymbol{\theta}}|_{\mathbf{Y}_{\mathbf{s}_n}}, \boldsymbol{\theta}^*) = \left[1 - \frac{D_\Theta(\hat{\boldsymbol{\theta}}|_{\mathbf{Y}_{\mathbf{s}_n \cup \mathbf{c}}, \boldsymbol{\theta}^*})}{D_\Theta(\hat{\boldsymbol{\theta}}|_{\mathbf{Y}_{\mathbf{s}_n}}, \boldsymbol{\theta}^*)}\right] \cdot \left[1 - \frac{D_\mathcal{Y}(\hat{\boldsymbol{\theta}}|_{\mathbf{s}_n \cup \mathbf{c}}, \boldsymbol{\theta}^*)}{D_\mathcal{Y}(\hat{\boldsymbol{\theta}}|_{\mathbf{Y}_{\mathbf{s}_n}}, \boldsymbol{\theta}^*)}\right] \tag{7.8}$$

7.3.3 Configuration de l'algorithme UCT

Afin d'adapter l'algorithme UCT au problème abordé, nous ajustons la constante C du critère UCB sur le premier jeu de données (les données du système *wildtype* fourni gratuitement) de façon à rendre plus efficace l'exploration de l'espace des expériences. On fixe ensuite la constante pour les prochaines expériences. Comme on se place dans le cadre de l'exploration pure (i.e. ce n'est pas le cumul des récompenses à la fin du jeu qui nous intéresse mais uniquement d'identifier le nœud donnant la meilleure récompense, cf b.), il est intéressant d'observer l'évolution des récompenses associées aux recommandations. Comme, de plus, la profondeur de l'arbre a une certaine influence sur le temps nécessaire à l'exploration, on va observer cette évolution pour différents horizons.

Tout d'abord, considérons l'évolution des récompenses pour un horizon de 1 (figure 7.12). L'horizon 1 correspond au cas particulier où l'arbre construit par UCT a une profondeur de 1 : on s'attache à choisir seulement la prochaine expérience à réaliser sans prendre en compte le fait qu'elle puisse être suivie d'autres. Dans ce cas, UCT est équivalent à un simple algorithme bandit. On teste des valeurs de la constante C du critère du UCB

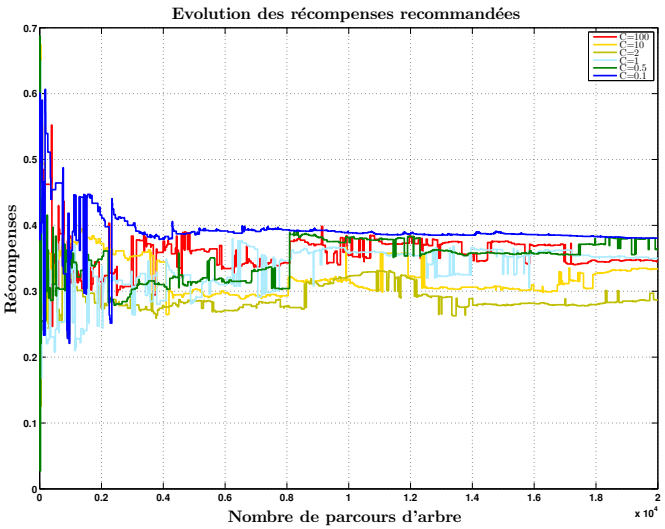

FIGURE 7.13 – Évolution des récompenses associées aux expériences recommandées en fonction de différents valeurs de compromis *exploration-exploitation* pour un horizon de 2.

variant de 0.1 à 100. Les meilleurs résultats sont obtenus avec $C = 100$, c'est-à-dire que l'exploration est privilégiée par rapport à l'exploitation lors du choix du bras du bandit à jouer. Cela signifie que l'algorithme a besoin de passer plus de temps à explorer pour fournir une recommandation pertinente dans ce cas. En revanche, si on augmente l'horizon à 2, comme en figure 7.13, il est plus intéressant d'exploiter assez tôt car une expérience pouvant être suivie de plusieurs autres, il ne suffit plus d'explorer une fois cette expérience pour juger de son utilité ; il faut aussi explorer au moins une fois toutes les combinaisons de 2 expériences possibles.

a. Réduction du nombre d'expériences

Afin d'améliorer les performances de l'algorithme EDEN dans la phase de sélection d'expérience, il peut être judicieux de réduire la taille de l'espace d'expériences en éliminant les expériences *a priori* non pertinentes. Dans le cas du problème posé dans DREAM, la structure du réseau de régulation est connue et est donnée sous la forme d'un graphe orienté. En exploitant cette connaissance on peut dans certain cas réduire le nombre totale nombre d'expériences. L'idée est d'éliminer les expériences dans lesquelles le système subit une perturbation sans que cela n'affecte les mesures. En effet, étant donné qu'une perturbation est coûteuse, il semble pertinent de privilégier les expériences qui en mesurent les effets.

Le réseau de DREAM (figure 7.10) présente une structure particulière puisqu'il s'agit d'un graphe acyclique. En ré-agençant les nœuds du graphe de régulation selon un ordre topologique on peut mettre en exergue la structure hiérarchique. Sur le réseau DREAM cela se traduit par une décomposition du graphe en 5 niveaux (voir la 7.14). Les numérotations des niveaux traduisent l'ordre topologique du graphe.

Le nombre total d'expérience réalisable dans le challenge DREAM est de 1368. Ce nombre est réduit à 325 en appliquant la stratégie de réduction d'expérience. Bien sur on pourrait laisser algorithme UCT pourrait se rendre compte par lui-même que ces expériences sont inintéressante, cependant cela nécessiterai un précieux temps de calcul.

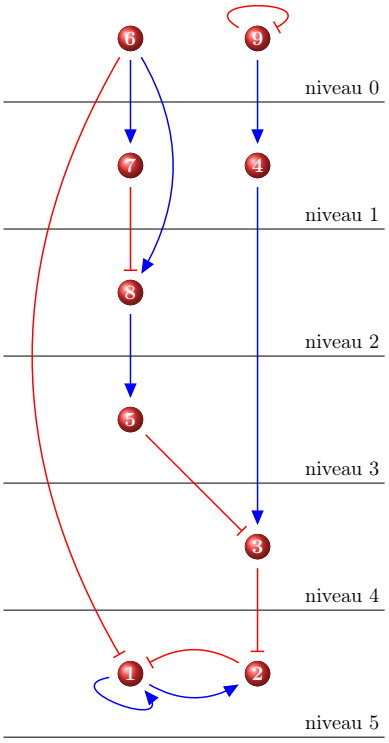

FIGURE 7.14 – Représentation hiérarchique du modèle 1 de DREAM7.

Le tableau 7.11 récapitule la configuration finale de l'algorithme obtenue pour la résolution du challenge DREAM

Hyperparamètre		Valeur
Facteur de ramification :	$\lvert\mathcal{E}\rvert$	325
Taille de l'espace des version :	N_{VS}	5000
Seuil de rejet des solutions :	λ	1.1
Budget initial :	B	10000
Budget intermédiaire :	β	1800
profondeur d'arbre :	H	3
Critère d'arrêt (parcours d'arbre) :	N	10^6
Stratégie de sélection :	ϕ	$UCB1(C = \frac{1}{\sqrt{2}})$
Fonction de recommandation	ψ	Robust Child
Fonction de récompense	R	Equation (7.8)

TABLE 7.11 – Configuration de l'algorithme EDEN pour le problème DREAM.

7.3.4 Comparaison des résultats avec ceux des autres compétiteurs.

Les résultats des différents participants de la compétition DREAM7 sont publiés dans Meyer et al. (2014). La plupart des participants ont également publié leurs résultats indépendamment à la suite de cette compétition Pauwels et al. (2014), Steiert et al. (2012) et les nôtres peuvent ainsi être trouvés dans Llamosi et al. (2014) et Mezine et al. (2015). Le tableau 7.12 montre les performances obtenues par les différents concurrents, incluant les nôtres qui ont été réalisées avant le développement d'EDEN.

Afin de pouvoir évaluer l'algorithme EDEN en le confrontant aux autres algorithmes, nous avons ajouté à ce tableau les scores obtenus en l'appliquant sur les données d'un challenge simulé. Il faut noter que nous ne pouvons plus accéder aux données de la compétition elle-même, maintenant qu'elle est clôturée. Nous les avons donc simulées à partir des valeurs de paramètres utilisées par les organisateurs et en utilisant exactement le même modèle de bruit. Les conditions sont donc tout à fait comparables même si les réalisations des bruits d'observations sont différentes dans notre application d'EDEN de celles de la compétition.

Les performances sont évaluées en fonction des deux critères de la compétition, présentés en paragraphe c., i.e. l'erreur sur les paramètres D_Θ et l'erreur sur les prédictions $D_\mathcal{Y}$. Les estimations finales de nos valeurs de paramètres sont obtenues en choisissant la meilleure des solutions contenues dans l'espace des versions discret obtenu à la dernière étape de l'algorithme, c'est-à-dire ici celle qui maximise la fonction de vraisemblance basée sur la statistique de Kolmogorov-Smirnov.

# **Équipe**	D_Θ $D_\mathcal{Y}$	Méthode de planification	Méthode d'échantillonage
1 Orangeballs	**0.0229** 0.0024	Games Tree	Sequential local search
2 TBP	0.8404 0.0160	Manual based on parameter uncertainty	Global method
3 Crux	0.1592 0.0354	Manual	Latin Hypercube
4 ForeCinHD	0.0899 0.0475	Manual	Levenberg-Marquardt Particle Swarm
5 Amis2011	0.1683 0.0980	Manual Simulation of estimation	Multistart Unscented Kalman Filter
6 Synmikro	0.0453 0.1988	A-optimality criterion	Local Levenberg-Marquardt
7 Biometris	0.1702 0.3625	Sensitivity analysis	Hybrid Local + Global
8 Reinhardt	0.8128 0.3564	Estimation of improved uncertainty	Global Metropolis Hastings
9 $\theta\sigma\beta$	0.3766 0.8180	Maximum Mutual Information	Approximate Bayesian Computation with Sequential Monte Carlo
10 BCB	0.0699 19.323	Minimize variance based on Fischer Information	Multistart local search
11 2pac	0.1883 3.2228	Iterative steps of training on data and simulation	Latin Hypercube Differential Evolution
12 Ntu	14.774 5.0278	Manuel	Local method
Algorithme EDEN	0.0371 **0.0016**	Automatique basé sur UCT=MCTS+Bandit	Local + Global ESS combiné à Nelder Mead

TABLE 7.12 – Performances de l'ensemble des participants de la compétition DREAM7-Parameter Inference Challenge. (Données extraites de Meyer et al. (2014)). La ligne colorée en rouge représente le score que nous avons obtenu pendant la compétition avant le développement de l'algorithme EDEN. La ligne colorée en vert représente notre score hors compétition en appliquant l'algorithme EDEN. Les scores D_Θ et $D_\mathcal{Y}$ correspondent respectivement aux erreurs sur les paramètres et sur la prédiction définies en paragraphe 7.3.2.c.. Le tableau donne pour chaque concurrent une description par mots-clés des méthodes employées pour, d'une part, sélectionner les expériences à soumettre et, d'autre part, pour l'estimation.

Les performances de l'algorithme EDEN approchent celles de la meilleure équipe du challenge Orangeballs. Nous parvenons à obtenir de meilleures performances en termes de prédiction, alors que l'équipe *orangeballs* réalise la meilleure performance pour l'estimation des paramètres. Ces différences de score s'expliquent en partie par la stratégie adoptée par l'équipe (décrite par Meyer et al. (2014)). *Orangeball* a commencé par acheter l'ensemble

des trajectoires des protéines du système *wildtype* en avançant comme argument le faible coût de ces expériences. EDEN en revanche a besoin de tester plusieurs combinaisons d'expériences pour ensuite les choisir une à une. Les tableaux 7.13 et 7.14 montrent l'ensemble des expériences sélectionnées. EDEN n'a produit qu'une seule expérience *wildtype*, estimant que la configuration wildtype apporterait probablement moins d'informations que le système perturbé. La stratégie d'EDEN au début du jeu est plus prudente, ce qui peut expliquer la meilleure qualité de nos prédictions. On remarque également que contrairement à EDEN, *Orangeballs* a réalisé deux expériences de type Gel-shift. Ces deux expériences, qui révèlent les valeurs exactes des paramètres $K_{21,}h_{21}, K_{61,}h_{61}$, sont très coûteuses mais permettent de réduire grandement l'erreur sur les paramètres. Cependant compte-tenu de la topologie du réseau, ces paramètres n'ont strictement aucune influence sur la prédiction du comportement des protéines p_3, p_5 et p_8, les rendent moins intéressants pour EDEN qui tente de résoudre le problème multi-objectifs.

Concernant les méthodes de planification employées, ce sont celles à base d'arbres (EDEN et Orangeballs) qui fournissent les meilleurs résultats, ce qui vient appuyer la thèse que l'anticipation peut être bénéfique pour la planification d'expériences. Cependant, alors qu'EDEN construit cet arbre automatiquement dans une structure de données numériques, l'équipe Orangeballs a construit ses arbres sur papier manuellement et a employé des heuristiques pour réduire l'espace de recherche des expériences.

Algorithme EDEN			
Expériences réalisés	Mesures	Taille des données	Crédits dépensés
wildtype	$(r_i)_{i=1...9}$	162	0
wildtype	p_3, p_4	402	400
knock-down gene 3	$(r_i)_{i=1...9}$	369	850
knock-down gene 7	p_3, p_5	402	750
knock-down gene 7	p_1, p_8	402	750
knock-down gene 5	p_1, p_2	402	750
knock-out gene 6	p_7, p_8	402	1200
knock-out gene 8	p_2, p_8	402	1200
rbs-down gene 7	p_1, p_8	402	850
rbs-down gene 9	p_1, p_2	402	750
knock-down gene 7	p_7, p_8	402	750
knock-down gene 8	p_5, p_8	402	750
knock-down gene 6	p_1, p_7	402	750
13 expériences	40 mesures	4931 points	9850

TABLE 7.13 – Liste ordonnée des expériences réalisées par l'algorithme EDEN

Equipe Orangeballs			
Expériences réalisés	Mesures	Taille des données	Crédits dépensés
wildtype	$(r_i)_{i=1...9}$	162 (9×18)	0
wildtype	p_1, p_2	402 (2×201)	400
wildtype	p_3, p_4	402	400
wildtype	p_4, p_7	402	400
wildtype	p_8, p_9	402	400
knock-down gene 6	$(r_i)_{i=1...9}$	369 (9×41)	850
knock-down gene 3	p_2, p_3	402	750
knock-down gene 4	p_3, p_6	402	750
knock-down gene 9	p_4, p_9	402	750
rbs-down gene 7	p_3, p_5	402	850
knock-out gene 6	p_1, p_2	402	1200
Gel-shift 61	k_{61}, h_{61}	2 (sans bruit)	1600
Gel-shift 21	k_{21}, h_{21}	2 (sans bruit)	1600
13 expériences	40 mesures	4555 points	9950

TABLE 7.14 – Liste désordonnée des expériences réalisées par l'équipe orangeballs.

Algorithme *NextExperiment*			
Expériences réalisés	Mesures	Taille des données	Crédits dépensés
wildtype	$(r_i)_{i=1...9}$	162	0
knock-out gene 7	p_3, p_8	402	1200
knock-out gene 7	p_8, p_9	402	1200
knock-out gene 9	p_3, p_8	402	1200
rbs-down gene 7	p_3, p_8	402	850
rbs-down gene 7	p_3, p_7	402	850
rbs-down gene 7	p_7, p_8	402	850
rbs-down gene 7	p_8, p_9	402	850
rbs-down gene 7	p_5, p_8	402	850
knock-down gene 7	p_3, p_8	402	750
knock-down gene 7	p_8, p_9	402	750
11 expériences	29 mesures	4182 points	9350

TABLE 7.15 – Liste des expériences réalisées à l'aide de l'algorithme *NextExperiment* Pauwels et al. (2014)

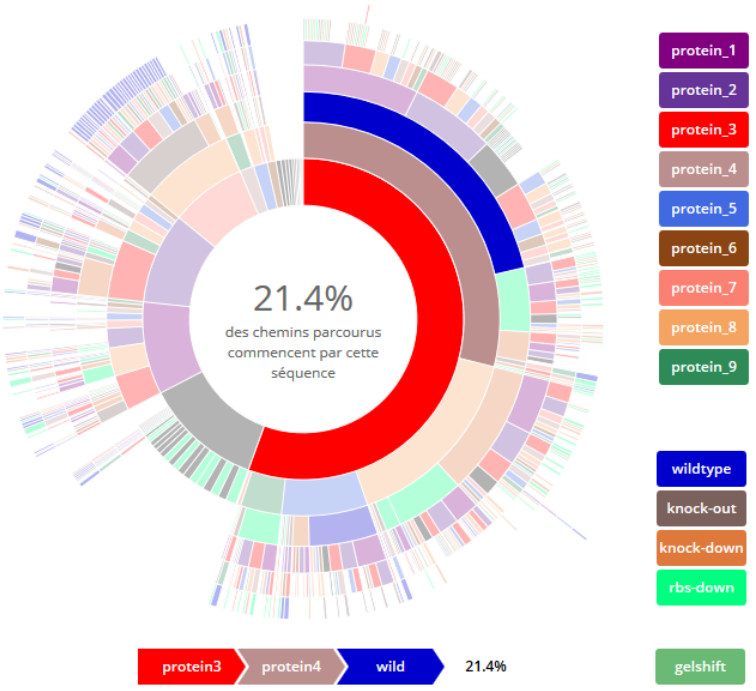

FIGURE 7.15 – Visualisation des statistiques de MCTS.

Afin de rendre plus conviviale l'utilisation de notre méthode, nous avons aussi développé un outil de visualisation des résultats de l'algorithme EDEN. La figure 7.15 en présente un exemple : on peut voir sous la forme de sections de cercles concentriques les différentes expériences qui ont été explorées lors d'une exécution. Ainsi, on voit que la première expérience, correspondant au système *wildtype* avec comme observation le couple de protéines (p3, p4), a été présente dans 21.4% des séquences testées. Les cercles concentriques de rayons plus élevés représentent les enchaînements d'observations et de perturbations suivants qui ont été explorés. L'horizon était ici fixé à 3 mais selon les séquences, le budget pouvait dans certains cas ne permettre que la réalisation de 2 expériences. Ce type de visualisation pourrait ainsi permettre d'identifier des expériences intéressantes même si elles ne sont pas recommandées par l'algorithme ou bien au contraire d'identifier des zones peu explorées, correspondant *a priori* à des expériences moins intéressantes.

Conclusion et perspectives

« Ce n'est pas la fin. Ce n'est même pas le commencement de la fin. Mais, c'est peut-être la fin du commencement. »

Winston Churchill

7.4 Synthèse

7.4.1 Les principales contributions

Cette thèse est née de notre participation aux compétitions DREAM [8] qui proposent, à la communauté internationale, une tâche à résoudre issue des problématiques soulevées par la biologie des systèmes. Notre motivation était l'aide à la planification d'expériences pour l'identification de modèles dynamiques de réseaux de régulation.

Notre première contribution consiste à formuler ce problème comme un problème à un joueur, ce qui ouvre la voie à l'utilisation de méthodes de recherche arborescente de Monte-Carlo (MCTS) développées en intelligence artificielle. L'outil MCTS offre une grande souplesse, nous permettant de traiter une grande diversité de problèmes de planification d'expérience, en prenant en compte les contraintes liées aux coût d'expériences et au budget limité, et prenant en compte la variété des expériences réalisables que ce soit en terme de perturbations (*knock-out*, *knock-down*, etc.) ou en terme de mesures (test de décalage du gel, fluorescence de protéines, puces à ADN, etc.). D'importants efforts ont été réalisés pour permettre une automatisation du processus de recommandation ainsi que l'approche globale EDEN dans le cas des problèmes *in silico* sur lesquelles nous avons testé le méta-algorithme.

Nous avons proposé des définitions de récompenses inédites qui intègrent les notions d'optimalité A, D et E provenant de la théorie de la planification optimale.

À travers le module d'estimation d'EDEN, nous avons également apporté une contribution importante à l'identification de systèmes dynamiques par UKF à partir de données issues de plusieurs expériences de perturbation, en proposant un algorithme d'estimation par fusion de modèles qui permet de mettre à jour la connaissance sur les paramètres par des corrections prenant en compte plusieurs expériences simultanées.

Les tests que nous avons réalisés sur le problème DREAM nous ont permis de mettre en évidence l'intérêt d'exploiter la topologie d'un réseau de régulation pour simplifier la structure des arbres de recherche en réduisant leur facteur de ramification en tenant compte du lien de causalité entre la perturbation d'un système et la mesure de ses effets.

Nous avons parlé ici principalement d'estimation de paramètres mais il pourra s'agir également de l'estimation des états cachés ou bien de réaliser des prédictions précises comme le montrent nos résultats sur le challenge DREAM.

8. http://dreamchallenges.org/

7.4.2 Les limitations

On peut évoquer quelques limitations de nos travaux. D'une part, même si quelques optimisations *ad hoc* ont permis sa réduction, le temps de calcul pour les réseaux de grande taille reste l'une des problématiques principales. C'est le calcul des récompenses qui est le plus chronophage car il requiert un grande nombre de simulations destinées à estimer l'utilité des combinaisons d'expériences candidates.

D'autre part, l'application de cette méthode au cas de l'estimation paramétrique de réseaux de régulation repose sur l'hypothèse que les paramètres du modèle restent inchangés lorsque le réseau est perturbé : c'est une hypothèse forte dont la validité peut être remise en cause selon les systèmes biologiques considérés.

Enfin, l'algorithme contient quelques hyper-paramètres qu'il est nécessaire de calibrer : λ, C, etc. Leurs valeurs sont à fixer pour chaque type de problème. Le paramètre λ détermine la qualité de l'espace des versions et aussi sa taille. Nous avons choisi de manière purement arbitraire, sachant que si λ est trop grand l'espace des versions risque d'inclure des comportements quasi invraisemblable, alors qu'un λ trop petit risque d'exclure des hypothèses potentiellement très proche de la vérité, voire la vérité elle-même. Une solution qu'on pourrait envisager serait de stratifier l'espaces des versions et affecter des poids sur chaque strates. Même si cela introduit de nouveaux hyper-paramètres comme le nombre de strates et les poids, c'est, semble-t-il, un début de piste intéressant.

7.4.3 Les extensions envisageables

a. Planification d'expériences pour l'inférence de réseaux

Dans cette thèse, nous nous sommes placé volontairement dans le cadre idéal dans lequel la structure de modèle est donnée. Cela nous a permis de nous concentrer sur les tâches d'estimation de paramètres et de recommandation d'expériences. Cependant, en pratique, il est rare que des systèmes dynamiques biologiques soient suffisamment maîtrisés au point d'en connaître complètement la structure sous-jacente. Nous avons donné à EDEN une structure suffisamment modulaire et générique pour qu'il soit immédiat de modifier l'algorithme d'apprentissage et les fonctions d'utilité et de cohérence pour pouvoir l'appliquer à un problème d'inférence de réseaux. Nous avons pu identifier un certain nombre de méthodes qui nous semblent éligibles comme les méthodes d'inférence de réseaux à base d'arbres de Irrthum et al. (2010), celles basées sur la notion d'information mutuelle avec réduction d'entropie de Villaverde et al. (2014), ou encore les méthodes à base noyaux à valeur opérateur développées par Lim (2015).

b. Contrôle des expériences en temps réel

Au cours d'une expérience, plusieurs mesures peuvent être effectuées pendant la durée de l'expérience. Or, dès les premières mesures, on dispose de nouvelles informations qui sont susceptibles de remettre en question l'utilité des prochaines mesures. Jusqu'à présent, nous nous sommes focalisés sur la planification hors-ligne d'expériences. Un algorithme de planification en ligne permettrait d'interrompre ou de réviser les prochaines mesures. Cette perspective est toutefois moins envisageable pour certains problèmes dans lesquels la durée des expériences peut s'étendre sur un grand intervalle de temps (horloge circadienne, croissance de plantes). Pour d'autres problèmes, l'enjeu sera de parvenir à réaliser des calculs dans un temps significativement plus court que la durée totale de l'expérience. Pour un robot scientifique comme Adam (King et al., 2004), la planification d'expériences en temps réel serait tout à fait pertinente. Par ailleurs, combiner les approches de programmation logique inductive d'Adam et les approches bandits pourrait être intéressant.

c. Proposer une approche constructive

On distingue deux types d'approches en apprentissage actif : l'approche sélective et l'approche constructive (voir section 3.2.2). Pour rappel, dans le cas d'une approche sélective, l'apprenant sélectionne les exemples d'apprentissage à soumettre à l'oracle parmi une base de données non-étiquetées prédéfinie. Au contraire, dans une approche constructive, c'est l'apprenant qui construit les exemples d'apprentissages ce qui lui confère une plus grande marge de manœuvre. Dans cette thèse, nous nous sommes placés dans le cadre d'une approche sélective, puisque les exemples d'apprentissage sont choisis par EDEN parmi une collection d'expériences imposées par l'utilisateur (expert humain). Par exemple, dans le challenge DREAM, les participants ont la possibilité d'appliquer les différentes perturbations à un gène seulement, de manière **indépendante** d'une expérience à une autre. Avec une approche constructive, il s'agirait d'extrapoler la recherche à de nouvelles expériences, par exemple en combinant ou en modifiant des expériences suggérées par un expert humain, ce qui reviendrait par exemple dans notre application à perturber **simultanément** deux ou plusieurs gènes. Le principal obstacle à l'élaboration d'une telle approche constructive reste la difficulté de construire de manière artificielle des expériences réalisables en pratique. Cela nécessiterait d'être en mesure de concevoir les contraintes à imposer à l'algorithme pour que celui-ci ne puisse proposer que des expériences réalistes.

d. Exploration de domaines d'expériences continus

L'implémentation actuelle de l'algorithme EDEN permet uniquement d'explorer les séquences d'expériences dans un domaine discret. Cependant explorer un domaine continu serait très avantageux pour agir finement sur les valeurs de certaines variables de contrôle. EDEN pourrait non seulement décider de la perturbation à réaliser (Knock-Down, Knock-Out, etc.) mais aussi de l'intensité de la perturbation. Pour atteindre cet objectif, une solution consiste à combiner MCTS avec le χ-armed bandit (Bubeck et al., 2011). Les χ-armed bandits permettent en effet de considérer un espace de bras continu.

7.5 Perspectives

7.5.1 Perspectives méthodologiques

Le temps de calcul du module de recommandation reste l'une des problématiques importantes pour la mise en œuvre pratique de nos résultats. Nous avons identifié de nombreuses pistes prometteuses qui méritent d'être explorées. Il y a en effet plusieurs marges de manœuvre pour optimiser ce temps de calcul, en agissant par exemple sur la mise en œuvre du calcul de récompense ou sur la stratégie de sélection.

a. Estimation d'utilité à l'aide de la notion de similarité

Lors de la phase de calcul de récompense intervenant à chaque simulation de l'algorithme MCTS, une procédure d'estimation est appliquée systématiquement afin de mesurer l'utilité de chaque réalisation de données associées à la séquence d'expériences sélectionnée. L'inconvénient est que cette opération est relativement coûteuse alors qu'elle est répétée un très grand nombre de fois (chaque fois qu'une séquence d'expériences est parcourue). Afin de contourner ce problème, nous avons envisagé une autre manière d'estimer l'utilité d'une combinaison d'expériences simulées. L'idée-clé repose sur l'exploitation des similarités des comportements dynamiques des différents modèles candidats pour les expériences testées. En utilisant la similarité comme un poids affecté à chaque candidat, on est alors capable d'approcher la plupart des utilités que l'on a défini dans cette thèse.

Soient deux vecteurs de paramètres candidats $\boldsymbol{\theta}_p$ et $\boldsymbol{\theta}_q$ et une expérience e. Dans un souci de clarté, nous introduisons les notations $\mathbf{y}^e_{k,p} = h^e(f(\mathbf{x}^e_0, \mathbf{u}^e, \boldsymbol{\theta}_p, t^e_k), t^e_k)$ et $\mathbf{y}^e_{k,q} = h^e(f(\mathbf{x}^e_0, \mathbf{u}^e, \boldsymbol{\theta}_q, t^e_k), t^e_k)$ pour exprimer les prédictions selon les hypothèses $\boldsymbol{\theta}_p$ et $\boldsymbol{\theta}_q$ du comportement du système au temps t^e_k dans les conditions expérimentales déterminer par

e. On propose de définir la fonction de similarité suivante :

$$\omega(\boldsymbol{\theta}_p, \boldsymbol{\theta}_q, e) = \prod_{i=1}^{m_e} 1 - \text{ks2test}\left(\left\{\mathbf{y}^e_{k,p}[i]\right\}_{k=0:K^e}, \left\{\mathbf{y}^e_{k,q}[i]\right\}_{k=0:K^e}\right) \tag{7.9}$$

On suppose ici une fonction de similarité binaire retournant le résultat du test de Kolmogorov-Smirnov donné par la fonction ks2test, qui renvoie 0 si le test détermine que les deux échantillons sont issus de la même distribution ; sinon le test renvoie 1. L'intérêt de définir une mesure de similarité de la sorte est que l'on peut en déduire les similarités de n'importe quelle combinaison d'entre elles simplement en multipliant les similarités des prédictions d'expériences individuelles. Étant donné une combinaison d'expériences $\mathbf{c}$ on a alors la relation :

$$\omega(\boldsymbol{\theta}_p, \boldsymbol{\theta}_q, \mathbf{c}) = \prod_{e \in \mathbf{c}} \omega(\boldsymbol{\theta}_p, \boldsymbol{\theta}_q, e) \tag{7.10}$$

Nous proposons d'exploiter cette mesure de similarité dans le calcul d'utilité. Rappelons la formule d'utilité u_A qui évalue l'intérêt d'une combinaison d'expérience selon le critère d'A-optimalité :

$$u_A(\mathbf{c}, \mathbf{Y}_{\mathbf{s}_n \cup \mathbf{c}}) = -\text{trace}\left(\text{Var}\left[\hat{\boldsymbol{\theta}} | \mathbf{Y}_{\mathbf{s}_n \cup \mathbf{c}}\right]\right) \tag{7.11}$$

Comme on a pu le voir précédemment, la stratégie actuelle pour calculer cette utilité nécessite l'estimation de la loi *a posteriori* $\boldsymbol{\theta}|_{\mathbf{Y}_{\mathbf{s}_n \cup \mathbf{c}}}$. Plutôt que de réaliser cette opération coûteuse, on peut pondérer les vecteurs de paramètres candidats afin d'approcher l'utilité sans avoir à estimer la connaissance *a posteriori*.

$$u_A(\mathbf{c}, \boldsymbol{\theta}^*, \mathbf{Y}_{\mathbf{s}_n \cup \mathbf{c}}) \approx -\frac{\sum_{\boldsymbol{\theta} \in \mathcal{V}(\mathbf{Y}_{\mathbf{s}_n})} \omega(\boldsymbol{\theta}, \boldsymbol{\theta}^*, \mathbf{c}) \cdot \|\boldsymbol{\theta} - \overline{\boldsymbol{\theta}}\|_2^2}{\sum_{\boldsymbol{\theta} \in \mathcal{V}(\mathbf{Y}_{\mathbf{s}_n})} \omega(\boldsymbol{\theta}, \boldsymbol{\theta}^*, \mathbf{c})} \tag{7.12}$$

Dans cette expression, on voit apparaître l'espace des versions $\mathcal{V}(\mathbf{Y}_{\mathbf{s}_n})$ qui est, on le rappelle, l'ensemble des hypothèses cohérentes avec les données collectées jusqu'à l'itération courante. On calcule de manière approchée la trace de covariance en affectant à chaque terme un poids correspondant à la similarité $\omega(\boldsymbol{\theta}, \boldsymbol{\theta}^*, \mathbf{c})$ entre les dynamiques générées respectivement par les vecteurs de paramètres $\boldsymbol{\theta}^*$ et $\boldsymbol{\theta}$. Ainsi, l'utilité d'une combinaison d'expériences sera peu élevée si, avec les données qu'elle génère, on obtient un grand nombre de vecteurs de paramètres de l'espace des versions qui, bien qu'éloignés de $\boldsymbol{\theta}^*$ (i.e. valeur élevée du terme $\|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|_2$), produisent des dynamiques de modèles indiscernables ($\omega = 1$) : c'est en effet ce qui se produit typiquement dans les situations de non-identifiabilités que l'on cherche à éviter. Autrement dit, on va chercher à réduire autant que possible la dispersion, autour de $\boldsymbol{\theta}^*$, des points générant des trajectoires similaires à celles qu'il génère lui-même. On normalise ensuite par la somme de tous les poids, d'une part, et par la dispersion non pondérée d'autre part, afin de se ramener à une valeur entre 0 et 1. De cette manière, nous nous passons des estimations à partir des données simulées.

En effet, seul l'espace des versions courant $\mathcal{V}(\mathbf{Y}_{\mathbf{s}_n})$, ainsi que les relations de similarités entre les membres de l'espace des versions et le paramètre $\boldsymbol{\theta}^*$ sont nécessaires pour calculer l'utilité d'une combinaison d'expériences $\mathbf{c}$ associée au jeu de données $\mathbf{Y}_\mathbf{c}$ sachant $\boldsymbol{\theta}^*$.

Remarquons que l'approche développée par Pauwels (2013) (décrite dans la section 4.4.2) a recours à une stratégie de pondération assez proche afin de calculer de manière approchée l'utilité d'une expérience pour réduire le risque empirique, à la différence que notre définition du poids s'applique aisément pour une combinaison d'expériences. Toutefois, un des obstacles que nous devons encore surmonter pour pouvoir appliquer cette méthodologie dans de bonnes conditions est lié à la qualité de la représentation de l'espace des versions. En effet, il faut que l'espace des versions soit correctement représenté, si possible en tenant compte de la sensibilité du modèle dans les conditions expérimentales explorées.

b. Stratégie de sélection adaptative.

La stratégie de sélection doit permettre de résoudre le compromis entre exploration et exploitation. Lorsque nous appliquons un algorithme comme UCB, ce compromis est réalisé à l'aide d'un hyper-paramètre fixé dès le début. Intuitivement, on peut penser qu'une méthode de sélection adaptative permettrait d'allouer plus efficacement les ressources et ainsi trouver plus rapidement les ressources de calculs. Dans (Tokic, 2010, Tokic and Palm, 2011) les auteurs décrivent un algorithme adaptatif qui peut s'appliquer dans le cadre des MDP. Il s'agit de l'algorithme VDBE (*Value-Difference Based Exploration*) qui s'appuie sur les fluctuations de la fonction de valeur que nous avons décrite dans la section sur les processus de décision de Markov (3.3.1).

c. Recours à des méta-modèles pour accélérer les phases de simulations

Le calcul d'une récompense nécessite de simuler le comportement pour l'expérience testée. La complexité de certains modèles peut alors peser sur le temps de simulation. Pour contourner ce problème, il est possible d'utiliser un méta-modèle qui approche le comportement du vrai système tout en demandant moins de ressources. On peut envisager de revenir ensuite au vrai modèle pour départager un nombre plus restreint d'expériences.

7.5.2 Perspectives d'application

a. Chronobiologie et chronothérapeutique

La chronobiologie, cette une discipline consistant à étudier les rythmes biologiques qui caractérisent la régulation les fonctions biologiques. Par exemple, certaines fonctions biologiques n'opère pas de manière constante au cours de la journée, mais fluctuent à un rythmes orchestré par l'horloge circadienne (figure 7.16).

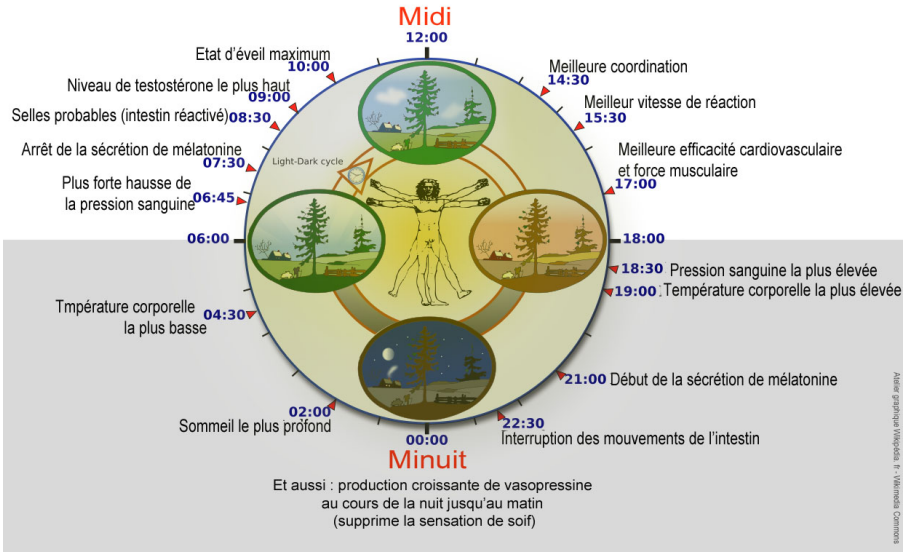

FIGURE 7.16 – Illustration du rythme circadien chez l'homme (source : Wikipedia [9])

La compréhension des mécanismes sous-jacents à ces rythmes peut permettre de fournir de meilleurs traitements médicamenteux (chronothérapeutique, Reinberg (1974)), en prescrivant la dose adéquate de médicament au moment idéal, on peut ainsi augmenter leur efficacité tout en réduisant les effets secondaires (citations). L'influence des rythmes biologiques sur l'effet de traitement médicaux n'est plus à démontrer comme en attestent les travaux de Touitou and Haus (2012).

Les expérimentations déjà réalisées sur les plantes et sur les mammifères ont permis d'identifier une quinzaine de gènes intervenant dans le processus mécanisme d'horloge circadienne ainsi qu'un certain nombre de régulations. Mais il reste encore du travail pour percer à jour toute la complexité de ce système. La modélisation de l'horloge circadienne par des systèmes d'équations différentielles ordinaires ou stochastiques a déjà contribué à améliorer notre compréhension de ces systèmes. L'enjeu de la chronothérapeutique est de relier ces modèles aux modèles prédictifs associés aux traitements médicamenteux. La

9. auteur : Lamiot. À partir d'une réalisation de YassineMrabet sur la base d'informations extraites de Smolensky and Lamberg (2015)

planification d'expériences peut certainement jouer un rôle important dans la sélection du meilleur rythme auquel les médicaments doivent être administrés.

b. Biologie synthétique.

Dans cette thèse, nous nous sommes focalisé sur les applications en biologie systémique. Dans ce cadre d'étude, nous employons des méthodes de rétro-ingénierie pour construire des modèles mathématiques représentatifs des systèmes dynamiques biologiques observés. Or on peut tout à fait envisager la démarche inverse, c'est-à-dire concevoir des modèles mathématiques dans le but de produire artificiellement des systèmes biologiques répondant à des besoins particuliers : en ingénierie, on parle alors de bio-ingénierie ou plus précisément de biologie synthétique (Andrianantoandro et al., 2006). La biologie synthétique remplit deux objectifs : d'une part améliorer notre compréhension des principes inhérents à la biologie via une approche complémentaire à la biologie systémique, d'autre part développer des organismes génétiquement reprogrammés pour remplir des fonctions biologiques bien précises, dans le but de répondre à des problématiques industrielles (énergie, médecine, cosmétique, etc.). Outre les aspects éthiques qui font débat, la biologie synthétique suscite l'intérêt des plus grands acteurs du monde industriel et bénéficie d'importants investissements. Dans ce contexte, il devient indispensable de rentabiliser le processus de production de ces organismes synthétiques en calibrant de manière optimale les interactions génétiques des systèmes conçus (Batt et al., 2007, Batt, 2014). La planification d'expériences par apprentissage actif pourrait être adaptée pour accroître la productivité des organismes employés et la qualité des produits synthétisés.

c. Médecine personnalisée, avatar virtuel et essai clinique *in silico*

Les progrès de la biologie moléculaire et de la génétique ouvrent la voie à la médecine personnalisée [10]. Contrairement à la médecine classique, la médecine personnalisée prend en compte le profil génétique des individus pour déterminer le traitement adapté et la posologie pour favoriser le processus de guérison. Dans ce contexte, l'amélioration des modèles quantitatifs de systèmes dynamiques biologiques permet d'entrevoir, dans un futur sans doute relativement lointain, le développement de la virtualisation des systèmes vivants en vue d'application dans le domaine médical. À terme, on pourrait envisager que des traitements présentant des risques pour le patient soient d'abord administrés à un avatar virtuel du patient avant d'être appliqués au patient lui-même.

10. aussi appelée médecine de précision ou 4**P**(**P**rédiction, **P**ersonalisation, **P**réemption, **P**lanification (anticipation)

Dans un autre registre, notre approche de planification d'expériences par apprentissage actif pourrait également contribuer au développement des essais cliniques *in silico*. En effet, les essais cliniques classiques, bien qu'indispensables aux progrès de la médecine, présentent certains risques pour les volontaires qui s'y soumettent. Les essais cliniques *in silico* visent à anticiper les complications liées à ces essais, en utilisant différents types de modèles (Boissel et al., 2013) destinés à prédire les effets néfastes des nouveaux traitements et les risques potentiels sur une population donnée.

d. Adam et Eve dans le jardin d'EDEN.

Les avancées dans les domaines de la robotique et des approches d'inférence logique ont permis de mettre au point Adam et Eve (King et al., 2004, 2009b), présentés la figure 7.17), deux robots capables de réaliser certaines tâches habituellement réalisées par un scientifique comme modéliser un phénomène biologique, planifier et réaliser des expérimentations, interpréter les résultats. En plus d'être totalement autonomes, ces robots possèdent des aptitudes à réaliser des expériences qui demandent des manipulations impossibles à reproduire par un humain.

(A) Adam (B) Eve

FIGURE 7.17 – Adam & Eve, les robots scientifiques (King et al., 2004)

Adam applique les outils classiques de la planification d'expériences optimale, afin de déterminer les expériences discriminant le plus grand nombre d'hypothèses à moindre coût. Comme pour notre problème, Adam est amené à construire séquentiellement une série d'expériences en prenant en compte le coût des expériences. Dans ces conditions il semble tout à fait envisageable de reproduire un schéma équivalent à EDEN, permettant à un robot comme Adam de choisir les expériences plus efficacement en conservant son autonomie.

Avec l'avènement de machines telles que Adam et Eve, une perspective intéressante de cette thèse pourrait être le développement d'une interface entre l'humain et l'intelligence artificielle permettant d'interagir plus efficacement dans le but de dépasser nos limites respectives.

FIGURE 7.18 – Adam et Eve dans le jardin d'Eden [11]

« *La science a fait de nous des dieux avant même que nous méritions d'être des hommes.* »

Jean Rostand

11. Auteur : Johann Wenzel Peter. License : Domaine public

Annexe A

Inférence bayésienne

> « *Le vrai génie réside dans l'aptitude à évaluer l'incertain, le hasardeux, les informations conflictuelles.* »
>
> Winston Churchill

Par opposition à l'approche fréquentiste où seules les données sont modélisées comme des réalisations de variables aléatoires, dans une approche bayésienne, le paramètre inconnu est également vu comme une variable aléatoire dont on souhaite caractériser la loi.

Comme son nom le suggère, l'approche bayésienne emploie la formule de Bayes pour lier ces différentes quantités. Introduite par le révérend Thomas Bayes (Bayes and Price, 1763) et indépendamment par Pierre Simon de Laplace, la formule de Bayes sert de socle à l'inférence bayésienne pour la mise à jour de la distribution de probabilité sur le paramètre.

Theorem A.1. Théorème de Bayes
Étant donnée une observation y et une valeur θ que peut prendre le paramètre, le théorème de Bayes permet d'exprimer la probabilité a posteriori en fonction de la probabilité a priori via la formule de Bayes (A.1)*. En supposant que la distribution de probabilité de chaque variable aléatoire possède une densité de probabilité, nous exprimons ici ce théorème à travers les densités de probabilité :*

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} \tag{A.1}$$

$$= \frac{p(y|\theta)p(\theta)}{\int_\theta p(y|\theta)d\theta}, \tag{A.2}$$

où $p(\theta)$ est la densité de probabilité de la variable aléatoire $\boldsymbol{\theta}$ sans tenir compte de l'observation de $\mathbf{y}$, appelée densité de probabilité *a priori* et $p(\theta|y)$ est la densité de probabilité de $\boldsymbol{\theta}$ ayant observé y. La densité de probabilité $p(\theta)$ résume l'information disponible sur $\boldsymbol{\theta}$ avant la prise en compte de la donnée y. Le théorème de Bayes dit que la densité de probabilité *a posteriori* $p(\theta|y)$ est proportionnelle au produit de la densité de probabilité *a priori* et de la vraisemblance.

$$\begin{gathered} p(\theta|y) \propto p(y|\theta) \times p(\theta) \\ \text{(densité a posteriori)} \propto \text{(vraisemblance)} \times \text{(densité a priori)} \end{gathered} \tag{A.3}$$

Lorsqu'on souhaite estimer $\boldsymbol{\theta}$, un des principaux avantages de l'approche bayésienne est de pouvoir incorporer de la connaissance *a priori* sur ce paramètre. On peut en effet fournir un *a priori* plus ou moins informatif (Box and Tiao, 2011) en s'appuyant sur les connaissances des experts, la littérature ou les résultats d'expériences passées, comme l'illustre la figure ci-dessous. Un autre avantage qu'offre ce type d'approche est de pouvoir traiter les cas où les données sont partiellement observables ou manquantes (Kong et al., 1994, Gelman and Rubin, 1996). La figure (A.1) montre l'influence que peut avoir l'*a priori* sur la mise à jour bayésienne de l'*a posteriori*. D'un autre côté, un inconvénient majeur de l'inférence bayésienne est lié aux aspects calculatoires, surtout dans le cas où l'approximation des intégrales nécessite un très grand nombre de simulations.

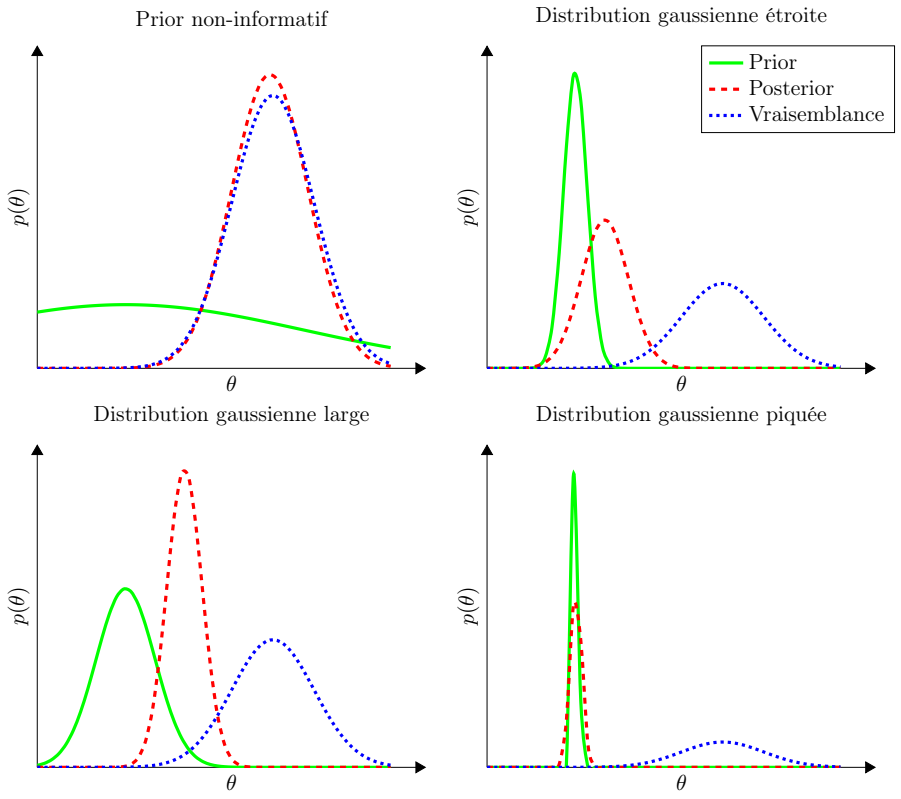

TABLE A.1 – Influence de la distribution a priori sur les paramètres estimés *a posteriori* (Van De Schoot et al., 2013). Illustration réalisée à partir de distributions gaussiennes, la vraisemblance est identique dans chaque cas de figure

Pour une vision plus approfondie de l'approche bayésienne, les ouvrages de Congdon (2007) et Parent and Bernier (2007) devraient satisfaire la curiosité des lecteurs, notamment en ce qui concerne les implémentations via les méthodes de type Monte-Carlo par chaîne de Markov.

Annexe B

Origine des équations cinétiques de Michaelis-Menten et de Hill

« *La politique c'est éphémère mais une équation est éternelle* »

Albert Einstein

Les équations de Michaelis-Menten et de Hill ont d'abord été proposées pour modéliser des réactions enzymatiques en biochimie. Une réaction enzymatique est une réaction chimique catalysée par une *enzyme* (une protéine particulière), c'est-à-dire que la présence de l'enzyme permet de déclencher ou d'accélérer la réaction par laquelle un réactif appelé *substrat* S est transformé en un *produit* P. Cette réaction se décompose en deux réactions (figure B.1) : la formation d'un complexe enzyme-substrat ES, et la catalyse qui transforme le substrat en produit P.

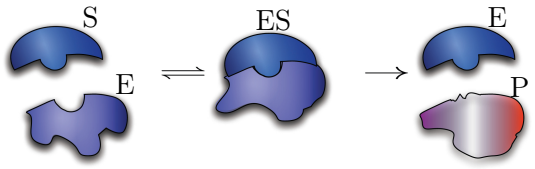

FIGURE B.1 – Schéma d'une réaction enzymatique (Guy-Bart, 2015)

B.1 Modèle de Michaelis-Menten

La forme canonique d'une réaction enzymatique est donnée par l'équation suivante (Chen et al., 2010) :

$$E + S \underset{k_r}{\overset{k_f}{\rightleftharpoons}} ES \overset{k_{cat}}{\rightarrow} E + P \tag{B.1}$$

La loi d'action de masse permet de modéliser l'ensemble de ces réactions sous la forme d'un système d'équations différentielles. La loi d'action de masse stipule que dans une réaction mettant en jeu au moins deux réactants, la vitesse de réactions pour une température donnée est proportionnelle au produit des concentrations des réactants. En appliquant ce principe à la réaction enzymatique (B.1), on obtient le système d'équations différentielles ordinaires suivant :

$$\begin{aligned}
\frac{d[ES]}{dt} &= k_f [E][S] - k_r [ES] - k_{cat} [ES] \\
\frac{d[E]}{dt} &= -k_f [E][S] + k_r [ES] + k_{cat} [ES] \\
\frac{d[S]}{dt} &= -k_f [E][S] + k_r [ES] \\
\frac{d[P]}{dt} &= k_{cat} [ES]
\end{aligned} \tag{B.2}$$

En pratique, on travaille plutôt avec le modèle de Michaelis Menten (B.3) obtenu en réduisant le nombre d'équations du système (B.2). Cette réduction est obtenue en utilisant deux arguments : la loi de conservation de la masse et l'approximation de quasi-stationnarité (Michaelis and Menten, 1913).
L'équation classique de Michaelis-Menten s'écrit sous la forme :

$$\frac{d[P]}{dt} \simeq -\frac{d[S]}{dt} \simeq V_{max} \frac{[S]}{[S] + K_M} \tag{B.3}$$

Où K_M est la constante de Michaelis spécifique à l'enzyme et V_{max} est la vitesse maximale initiale :

$$K_M \equiv \frac{k_r + k_{cat}}{k_f} \text{ et } V_{max} = k_{cat} E(0)$$

B.2 Formalisme de Hill

Le modèle de Michaelis-Menten n'est pas suffisamment général pour couvrir l'ensemble des réactions enzymatiques. Pour les réactions plus complexes comme les réactions enzymatiques coopératives (figure B.2), on utilise un modèle plus riche, fourni par les équations de Hill.

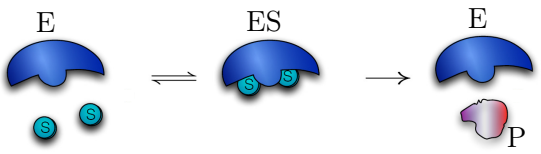

FIGURE B.2 – Schéma d'une réaction enzymatique avec coopérativité (Guy-Bart, 2015)

Dans ce type de réaction, plusieurs substrats coopèrent pour amorcer la réaction. Le nombre h de substrats devant coopérer est appelé *degré de coopérativité* ou coefficient de Hill. L'équation suivante présente la forme canonique d'une réaction enzymatique coopérative.

$$E + h.S \underset{k_r}{\overset{k_f}{\rightleftharpoons}} ES \overset{k_{cat}}{\rightarrow} E + P \tag{B.4}$$

La loi d'action de masse permet alors d'écrire le système d'équations différentielles ordinaires suivant :

$$\begin{aligned}
\frac{d[ES]}{dt} &= k_f [E] [S]^h - k_r [ES] - k_{cat} [ES] \\
\frac{d[E]}{dt} &= -k_f [E] [S]^h + k_r [ES] + k_{cat} [ES] \\
\frac{d[S]}{dt} &= h.(-k_f [E] [S]^h + k_r [ES]) \\
\frac{d[P]}{dt} &= k_{cat} [ES]
\end{aligned} \tag{B.5}$$

En utilisant les mêmes arguments que pour la construction du modèle de Michaelis-Menten, on obtient le modèle de Hill suivant :

$$\frac{d[P]}{dt} \simeq -\frac{d[S]}{dt} \simeq V_{max} \frac{[S]^h}{[S]^h + K_M} \tag{B.6}$$

On retrouve K_M la constante de Michaelis spécifique à l'enzyme et V_{max} la vitesse maximale initiale :

$$K_M \equiv \frac{k_r + k_{cat}}{k_f} \text{ et } V_{max} = h.k_{cat} E(0)$$

Annexe C

Expérimentation biologique

« *Observer, c'est perturber* »

Hubert Reeves

C.1 Techniques de mesures

C.1.1 Mesure d'expression de gènes : puces à ADN

Les puces à ADN, ou biopuces (micro-arrays), ont été imaginées en 1983 par Tse-Wen Chang (Chang, 1983), un chercheur d'une entreprise de biotechnologies, Centocor, et développées principalement à partir de 1995 suite à la publication dans *Science* d'un article présentant leur utilisation pour l'analyse différentielle de l'expression de 45 gènes d'Arabidopsis par des chercheurs de l'université de Stanford (Schena et al., 1995).

Ces puces permettent aujourd'hui de mesurer simultanément le niveau d'expression de plusieurs milliers à plusieurs centaines de milliers de gènes dans un échantillon biologique. Ce sont des lames, en verre ou en plastique, de quelques centimètres carrés sur lesquelles on dépose les ARNs messager d'un échantillon biologique, préalablement extraits, amplifiés et marqués par un colorant, les fluorochromes (souvent rouges ou verts comme l'illustre la figure C.1).

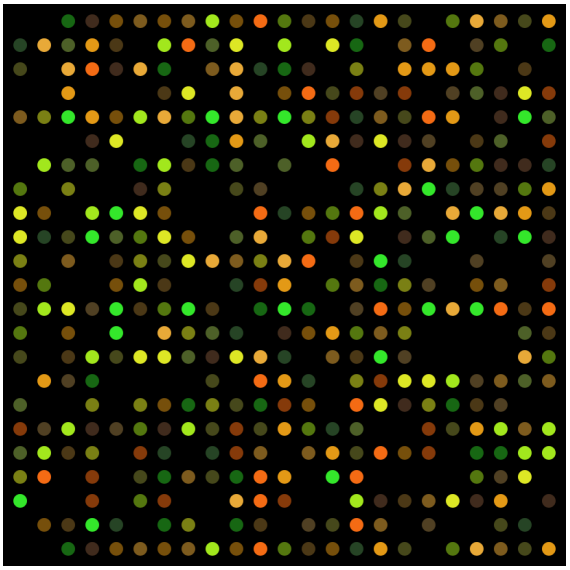

FIGURE C.1 – Fluorescence observable sur une puce à ADN.

Sur ces lames sont fixées des sondes d'ADN : ce sont des séquences synthétiques qui peuvent être soit des séquences longues, de l'ordre de 500 à 5000 bases, soit, plus fréquemment, des séquences courtes, appelées oligonucléotides, de l'ordre de 20 à 80 bases. À ces sondes vont venir s'attacher les séquences d'ARN messager complémentaires correspondantes qui sont présentes dans l'échantillon, grâce à la propriété que possède l'ADN dénaturé de reformer spontanément sa double hélice lorsqu'il est mis en présence d'un brin complémentaire : c'est la réaction d'hybridation, qui peut durer ici 15 à 20 heures selon l'organisme considéré. La lame est ensuite lue par un scanner à très haute résolution pour créer une image de la puce : plus il y a eu hybridation sur une sonde donnée, plus en ce point l'intensité du signal généré par le fluorochrome fixé sur l'ARNm est élevée. Les résultats sont normalisés pour ensuite être analysés et ainsi fournir le profil d'expression de chaque gène d'intérêt (Cotillard, 2009).

On utilise souvent l'analyse différentielle qui consiste à hybrider sur la même lame un échantillon contrôle de référence mélangé à l'échantillon à tester, chacun marqué par un fluorochrome différent. Ce type d'approche permet de s'affranchir des différences d'affinité des sondes (toutes les sondes ne fixent pas de la même manière leur séquence complémentaire) et ainsi d'obtenir des résultats quantitatifs interprétables. La précision toutefois limitée qu'apporte cette technique fait qu'elle est en pratique utilisée pour identifier une liste de gènes potentiels agissant sur la fonction d'intérêt et que leur implication est ensuite testée à l'aide d'autres techniques comme la PCR quantitative (qPCR, quantitative Polymerase Chain Reaction).

C.1.2 Mesures de concentrations de protéines : protéine de fusion fluorescente.

Cette méthode exploite la curieuse propriété d'une méduse, l'*Aequorea victoria*, à produire une protéine émettant une fluorescence de couleur verte. Son gène peut être fusionné *in vitro* au gène d'une protéine que l'on souhaite étudier et le gène recombinant ainsi obtenu est réintroduit dans des cellules, qui vont alors synthétiser la protéine de fusion fluorescente. Cela permet, à l'aide d'un microscope à fluorescence par exemple, de suivre la protéine ciblée dans son environnement naturel (la cellule vivante) (Snapp, 2005).

C.1.3 Mesures de la valeur de paramètres : retard à la migration sur gel

Le retard sur gel (EMSA ou electrophoretic mobility shift assay) est une technique de biologie moléculaire permettant d'étudier une interaction entre une protéine et de l'ADN ou de l'ARN. L'une de ses premières utilisations dans ce cadre remonte à 1981 (Fried and Crothers, 1981). Pour cela, on fait migrer par électrophorèse sur gel polyacrylamide le fragment d'ADN seul d'une part et le fragment d'ADN mis en contact avec la protéine ciblée d'autre part. On compare ensuite leurs vitesses de migration, en mesurant les positions atteintes par les « bandes » associées sur chacune des pistes de l'électrophorèse : comme la motilité d'un complexe ADN-protéine décroît avec le nombre de protéines qui y sont attachées, on peut en déduire les rapports stœchiométriques de cette réaction. Des répétitions de cette expérience à différents intervalles de temps permettent également d'estimer la constante du premier ordre de la cinétique de dissociation du complexe.

C.2 Techniques de perturbations

C.2.1 Invalidation génétique ou knock-out.

Ce terme a été initialement introduit pour parler de souris transgéniques, chez qui un gène particulier est « éteint », généralement par l'insertion ou le remplacement aléatoire d'un tronçon d'ADN, de sorte à invalider la production de la protéine associée. Pour réaliser cela, on isole des cellules souches à partir d'embryons de souris, que l'on met en contact avec de l'ADN contenant le gène modifié. On a alors échange de matériel génétique par recombinaison homologue, ce qui permet d'introduire cette séquence défectueuse dans les cellules embryonnaires. En se développant, celles-ci donneront naissance à un organisme porteur de la modification recherchée Hall et al. (2009).

C.2.2 Réduction de l'expression d'un gène (knock-down) par siRNA.

Sans l'« éteindre » complètement, il est possible de réduire l'expression d'un gène soit de manière permanente par une modification génétique, soit de manière transitoire par un traitement avec une séquence courte de nucléotides pouvant se lier spécifiquement à une séquence d'ARN messagers et ainsi empêcher l'expression de gènes en clivant cet ARN. Dans le second cas (qui est celui qui nous intéresse dans notre cadre d'application), de petits ARN interfèrent (ou siRNA pour small interfering RNA) sont introduits dans le cytoplasme de la cellule. Par complémentarité des bases nucléiques, ceux-ci vont reconnaître leur transcrit cible, l'ARN messager considéré, et s'y cliver. Les complexes ainsi formés sont ensuite rapidement dégradés par une protéine spécifique, *Argonaute*, empêchant toute nouvelle traduction de la protéine codée par ces ARN messagers et les rendant ainsi plus « silencieux ».

C.2.3 Réduction de l'activité ribosomale (RBS-down).

On peut enfin faire varier la production de protéines en agissant sur l'activité des ribosomes qui en contrôlent la synthèse. Les ribosomes sont des complexes formés de protéines et d'ARN capables de lire l'ARN messager et de polymériser des acides aminés pour former la protéine correspondante.

Annexe D

Publications

« L'importance d'un travail scientifique peut se mesurer au nombre des publications rendues obsolètes par celui-ci. »

David Hilbert

La plupart des travaux exposés dans cette thèse ont fait l'objet de publications dans des conférences et dans des workshops.

Articles

- Llamosi, A., Mezine, A., d'Alché-Buc, F., Letort, V., & Sebag, M. (2014, September). Experimental design in dynamical system identification : A bandit-based active learning approach. *In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 306-321). Springer Berlin Heidelberg.*
- Meyer, P., Cokelaer, T., Chandran, D., Kim, K. H., Loh, P. R., Tucker, G. ... DREAM 6&7 Parameter Estimation Consortium [1] *et al.* (2014). Network topology and parameter estimation : from experimental design methods to gene regulatory network kinetics using a community based approach. *BMC systems biology.*
- Mezine, A., Llamosi, A., Letort, V., Sebag, M., & d'Alché-Buc, F. (2015, July). Autonomous learning of parameters in differential equations. *In 32nd International Conference on Machine Learning (ICML)-AutoML workshop.*

Posters

- Llamosi, A., Mezine, A., d'Alché-Buc, F., Letort, V., & Sebag, M. (2014, September). EDEN : Experimental Design for the Estimation of Network. *In 13th European Conference on Computational Biology (ECCB) - MLSB Workshop.*
- Mezine, A., Llamosi, A., Letort, V., Sebag, M., & d'Alché-Buc, F. (2015, July). Autonomous learning of parameters in differential equations. *In 32nd International Conference on Machine Learning (ICML)-AutoML workshop.*

1. Dont la *team amis2011* : Mezine, A., Llamosi, A., Letort, V., Sebag, M. et d'Alché-Buc, F.

Abréviations

ADN	**A**cide **D**ésoxyribo**N**ucléique
ARNm	**A**cide **R**ibo**N**ucléique **m**essager
BAAL	**BA**ndit-based **A**ctive **L**earner
BED	**B**ayesian **E**xperimental **D**esign
CESS	**C**ooperative **E**nhanced **S**catter **S**earch
DOE	**D**esign **O**f **E**xperiment
DREAM	**D**ialogue for **R**everse **E**ngineering **A**ssessments and **M**ethods
EDEN	**E**xperimental **De**sign for **N**etwork inference
EDO	**E**quation **D**ifferentielle **O**rdinaire
EKF	**E**xtended **K**alman **F**ilter
e**SS**	enhanced **S**catter **S**earch
JAK	**JA**nus **K**inase
KF	**K**alman **F**ilter
KS	**K**olmogorov **S**mirnov
MAB	**M**ulti-**A**rmed **B**andit
MBDOE	**M**odel-**B**ased **D**esign **O**f **E**xperiment
MCTS	**M**onte **C**arlo **T**ree **S**earch
MDP	**M**arkov **D**ecision **P**rocess
PLNE	**P**rogrammation **L**inéaire en **N**ombres **E**ntiers
MIF	**M**atrice d'**I**nformation de **F**isher
SS	**S**catter **S**earch
STAT	**S**ignal **T**ransducer and **A**ctivator **T**ranscription
UCB	**U**pper **C**onfidence **B**ounds
UCT	**U**pper **C**onfidence bounds for **T**rees
UKF	**U**nscented **K**alman **F**ilter

Bibliographie

Abe, N. and Hiroshi, M. (1998). Query learning strategies using boosting and bagging. In *Machine Learning : Proceedings of the Fifteenth International Conference (ICML'98)*, pages 1–10. 42

Anderson, B. D. and Moore, J. B. (2012). *Optimal filtering*. Courier Corporation. 23

Anderson, B. D., Moore, J. B., Di Franco Rmm, L. S., and Sage, M. (1979). *Optimal Filtering*. Citeseer. 23

Andrianantoandro, E., Basu, S., Karig, D. K., and Weiss, R. (2006). Synthetic biology : new engineering rules for an emerging discipline. *Molecular systems biology*, 2(1). 143

Asprey, S. and Macchietto, S. (2000). Statistical tools for optimal dynamic model building. *Computers & Chemical Engineering*, 24(2–7) :1261 – 1267. 103

Atkinson, A. (1996). The usefulness of optimum experimental designs. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 59–76. 59, 59

Atkinson, A., Donev, A., and Tobias, R. (2007). *Optimum experimental designs, with SAS*. Oxford Univ. Press, UK. 59

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3) :235–256. 48, 48

Baker, S. M., Poskar, C. H., and Junker, B. H. (2011). Unscented kalman filter with parameter identifiability analysis for the estimation of multiple parameters in kinetic models. *EURASIP Journal on Bioinformatics and Systems Biology*, 2011(1) :1–8. 87

Batt, G. (2014). *Design, optimization and control in systems and synthetic biology*. PhD thesis, Université Paris-Diderot-Paris VII. 143

Batt, G., Yordanov, B., Weiss, R., and Belta, C. (2007). Robustness analysis and tuning of synthetic gene networks. *Bioinformatics*, 23(18) :2415–2422. 143

Baum, E. B. and Lang, K. (1992). Query learning can work poorly when a human oracle is used. In *International Joint Conference on Neural Networks*, volume 8, page 8. 39

Bayes, M. and Price, M. (1763). An essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfrs. *Philosophical Transactions (1683-1775)*, pages 370–418. 147

Bellman, R. E. (1952). On the theory of dynamic programming. *Proceedings of the National Academy of Sciences of the United States of America*, 38(8) :716–719. 44

Bellman, R. E. (1957). Dynamic programming. *Princeton University Press*. 44

Berry, D. A., F. B. (1985). *Bandit Problems : Sequential Allocation of Experiments*. Chapman and Hall, London. 66

Boissel, J.-P., Kahoul, R., Marin, D., and Boissel, F.-H. (2013). Effect model law : an approach for the implementation of personalized medicine. *Journal of personalized medicine*, 3(3) :177–190. 144

Box, G. E. and Tiao, G. C. (2011). *Bayesian inference in statistical analysis*, volume 40. John Wiley & Sons. 148

Box, G. E. and Wilson, K. (1951). On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 13(1) :1–45. 62

Box, M. (1971). Bias in nonlinear estimation. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 171–201. 66

Brazhnik, P., de la Fuente, A., and Mendes, P. (2002). Gene networks : how to put the function in genomics. *TRENDS in Biotechnology*, 20(11) :467–472. 11

Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2) :123–140. 42

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S., et al. (2012). A survey of monte carlo tree search methods. *Computational Intelligence and AI in Games, IEEE Transactions on*, 4(1) :1–43. 51, 79

Brügmann, B. (1993). Monte carlo go. Technical report, Citeseer. 50

Bubeck, S., Munos, R., and Stoltz, G. (2009). Pure exploration in multi-armed bandits problems. In *Algorithmic Learning Theory*, pages 23–37. Springer. 49, 49, 50, 50, 52

Bubeck, S., Munos, R., and Stoltz, G. (2011). Pure exploration in finitely-armed and continuous-armed bandits. *Theoretical Computer Science*, 412(19) :1832–1852. 49, 139

Campbell, M., Hoane, A. J., and Hsu, F.-h. (2002). Deep blue. *Artificial intelligence*, 134(1) :57–83. 51

Chaloner, K. and Verdinelli, I. (1995). Bayesian experimental design : A review. *Statistical Science*, pages 273–304. 64, 64

Chang, T.-W. (1983). Binding of cells to matrixes of distinct antibodies coated on solid surface. *Journal of Immunological Methods*, 65 :217–223. 155

Chen, T., He, H., and Church, G. (1999). Modeling gene expression with differential equations. In *Proceeedings of Pacific Symposium of Biocomputing 1999*. 12, 13, 14

Chen, W. W., Niepel, M., and Sorger, P. K. (2010). Classic and contemporary approaches to modeling biochemical reactions. *Genes & development*, 24(17) :1861–1875. 14, 151

Chernoff, H. (1959). Sequential design of experiments. *The Annals of Mathematical Statistics*, pages 755–770. 65

Chernoff, H. (1968). *Sequential designs*. Department of Statistics, Stanford University.

Chernoff, H. (1972). *Sequential analysis and optimal design*, volume 8. Society of Industrial and Applied Mathematics. 65

Chernoff, H. (1975). Approaches in sequential design of experiments. Technical report, DTIC Document.

Chernoff, H. (1981). *Lectures on Optimal Design and Sequential Analyses*. Citeseer. 65

Chis, O.-T., Banga, J., and E, B.-C. (2011). Structural identifiability of systems biology models : A critical comparison of methods. *PLoS ONE*, 6(11) :e27755. 31

Cohn, D., Atlas, L., and Ladner, R. (1994). Improving generalization with active learning. *Machine learning*, 15(2) :201–221. 38, 42, 42, 42

Congdon, P. (2007). *Bayesian statistical modelling*, volume 704. John Wiley & Sons. 149

Cornuéjols, A. and Miclet, L. (2011). *Apprentissage artificiel : concepts et algorithmes*. Editions Eyrolles. 36, 38, 40

Cotillard, A. (2009). *Développement d'une méthodologie robuste de sélection de gènes dans le cadre d'une activation pharmacologique de la voie PPAR*. PhD thesis, Ecole Centrale Paris. 156

Crick, F. (1970). Central dogma of molecular biology. *Nature*. 9

Culotta, A. and McCallum, A. (2005). Reducing labeling effort for structured prediction tasks. In *AAAI*, pages 746–751. 41

Dagan, I. and Engelson, S. P. (1995). Committee-based sampling for training probabilistic classifiers. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 150–157. 43

d'Alché Buc, F. and Brunel, N. (2009). Estimation of parametric nonlinear odes for biological networks identification. *Learning and Inference in Computational Systems Biology*, pages 61–96. 25

Dasgupta, S., Monteleoni, C., and Hsu, D. J. (2007). A general agnostic active learning algorithm. In *Advances in neural information processing systems*, pages 353–360. 42

De Jong, H. (2002). Modeling and simulation of genetic regulatory systems : a literature review. *Journal of computational biology*, 9(1) :67–103. 12, 13

DeGroot, M. H. (1970). *Optimal Statistical Decisions*. McGraw-Hill. 64

Dobre, S. (2010). *Analyses de sensibilité et d'identifiabilité globales. Application à l'estimation de paramètres photophysiques en thérapie photodynamique*. PhD thesis, Université Henri Poincaré-Nancy I. 57

Eberhart, R. C. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, volume 1, pages 39–43. New York, NY. 48

Egea, J. A., Henriques, D., Cokelaer, T., Villaverde, A. F., MacNamara, A., Danciu, D.-P., Banga, J. R., and Saez-Rodriguez, J. (2014). Meigo : an open-source software suite

based on metaheuristics for global optimization in systems biology and bioinformatics. *BMC bioinformatics*, 15(1) :136. 110

Egea, J. A., Martí, R., and Banga, J. R. (2010). An evolutionary method for complex-process optimization. *Computers & Operations Research*, 37(2) :315–324. 29

Fariza, T. (2014). *Bioinformatique des ARNs non-codants : Algorithmes pour leur identification et la prédiction de leur structure*. PhD thesis, Université d'Evry-Val d'Essonne. 10

Feldman, Z. and Domshlak, C. (2014). On mabs and separation of concerns in monte-carlo planning for mdps. In *Twenty-Fourth International Conference on Automated Planning and Scheduling*. 47, 50

Fouchet, A., Delosme, J.-M., and D'Alché-Buc, F. (2013). Gene regulatory network inference using ensembles of local multiple kernel models. In *Seventh international workshop on Machine Learning in Systems Biology, satellite meeting of ISMB'2013*. 110

Franceschini, G. and Macchietto, S. (2008). Model-based design of experiments for parameter precision : State of the art. *Chemical Engineering Science*, 63(19) :4846–4872. 56, 56, 57, 58, 62, 103

Freund, Y., Seung, H. S., Shamir, E., and Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine learning*, 28(2-3) :133–168. 42

Fried, M. and Crothers, D. M. (1981). Equilibria and kinetics of lac repressor-operator interactions by polyacrylamide gel electrophoresis. *Nucleic Acids Research*, 9(23) :6505–6525. 157

Galvanin, F., Barolo, M., Bezzo, F., and Macchietto, S. (2010). A backoff strategy for model-based experiment design under parametric uncertainty. *AIChE journal*, 56(8) :2088–2102. 56

Galvanin, F., Macchietto, S., and Bezzo, F. (2007). Model-based design of parallel experiments. *Industrial & engineering chemistry research*, 46(3) :871–882. 56, 103

Garivier, A. and Cappé, O. (2011). The kl-ucb algorithm for bounded stochastic bandits and beyond. In *COLT*, pages 359–376. 110

Gelly, S. and Silver, D. (2007). Combining online and offline knowledge in uct. In *Proceedings of the 24th international conference on Machine learning*, pages 273–280. ACM. 50

Gelman, A. and Rubin, D. B. (1996). Markov chain monte carlo methods in biostatistics. *Statistical Methods in Medical Research*, 5(4) :339–355. 148

Ghahramani, Z. (1998). Learning dynamic bayesian networks. In *Adaptive processing of sequences and data structures*, pages 168–197. Springer. 20

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1) :156–166. 28, 29

Golberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. *Addion wesley*, 1989. 48

Guy-Bart, S. (2015). Modelling in biology. *Imperial College London*. 151, 152

Hall, B., Limaye, A., and Kulkarni, A. B. (2009). Overview : Generation of gene knockout mice. *Current protocols in cell biology*, 19 :1217. 157

Hanneke, S. (2009). Theoretical foundations of active learning. *Science*. 38

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2) :100–107. 51

Hartikainen, J., Solin, A., and Särkkä, S. (2011). *Optimal Filtering with Kalman Filters and Smoothers a Manual for the Matlab toolbox EKF/UKF Version 1.3*. 110

Hecker, M., Lambeck, S., Toepfer, S., Van Someren, E., and Guthke, R. (2009). Gene regulatory network inference : data integration in dynamic models—a review. *Biosystems*, 96(1) :86–103. 87, 87

Hengl, S., Kreutz, C., Timmer, J., and Maiwald, T. (2007). Data-based identifiability analysis of non-linear dynamical models. *Bioinformatics*, 23(19) :2612–2618. 31, 34

Hindmarsh, A. C., Brown, P. N., Grant, K. E., Lee, S. L., Serban, R., Shumaker, D. E., and Woodward, C. S. (2005). Sundials : Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, 31(3) :363–396. 110

Ho, C.-H., Tsai, M.-H., and Lin, C.-J. (2011). Active learning and experimental design with svms. In *Active Learning and Experimental Design@ AISTATS*, pages 71–84. 66

Howard, R. A. (1960). Dynamic programming and markov processes. 44

Irrthum, A., Wehenkel, L., Geurts, P., et al. (2010). Inferring regulatory networks from expression data using tree-based methods. *PloS one*, 5(9) :e12776. 137

Julier, S. J. and Uhlmann, J. K. (1997). New extension of the kalman filter to nonlinear systems. In *AeroSense'97*, pages 182–193. International Society for Optics and Photonics. 23

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1) :35–45. 21

Kaufmann, E., Cappé, O., and Garivier, A. (2012). On bayesian upper confidence bounds for bandit problems. In *AISTATS*, pages 592–600. 110

Kiefer, J. (1959). Optimum experimental designs. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 272–319. 59

Kiefer, J. (1975). Construction and optimality of generalized youden designs ii. *A Survey of Statistical Designs and Linear Models*, pages 333–353. 62

King, R. D., Rowland, J., Oliver, S. G., Pir, P., Aubrey, W., Liakata, M., Markham, M., Soldatova, L. N., Whelan, K. E., Clare, A., et al. (2009a). The robot scientist adam. *Computer*, 42(8) :46–54. 39

King, R. D., Rowland, J., Oliver, S. G., Young, M., Aubrey, W., Byrne, E., Liakata, M., Markham, M., Pir, P., Soldatova, L. N., et al. (2009b). The automation of science. *Science*, 324(5923) :85–89. 39, 144

King, R. D., Whelan, K. E., Jones, F. M., Reiser, P. G., Bryant, C. H., Muggleton, S. H., Kell, D. B., and Oliver, S. G. (2004). Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971) :247–252. 39, 138, 144, 144

Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., et al. (1983). Optimization by simulated annealing. *science*, 220(4598) :671–680. 48

Knuth, D. E. and Moore, R. W. (1976). An analysis of alpha-beta pruning. *Artificial intelligence*, 6(4) :293–326. 51

Kocsis, L. and Szepesvári, C. (2006). Bandit based monte-carlo planning. In *Machine Learning : ECML 2006*, pages 282–293. Springer. 52, 52

Kong, A., Liu, J. S., and Wong, W. H. (1994). Sequential imputations and bayesian missing data problems. *Journal of the American statistical association*, 89(425) :278–288. 148

Kreutz, C. and Timmer, J. (2009). Systems biology : experimental design. *FEBS journal*, 276(4) :923–942. 63, 103, 103

Laguna, M. and Martí, R. (2005). Experimental testing of advanced scatter search designs for global optimization of multimodal functions. *Journal of Global Optimization*, 33(2) :235–255. 29

Laguna, M. and Marti, R. (2012). *Scatter search : methodology and implementations in C*, volume 24. Springer Science & Business Media. 29

Lehmann, E. L. (2007). *Reminiscences of a statistician : The company I kept*. Springer Science & Business Media. 59

Lewis, D. D. and Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the eleventh international conference on machine learning*, pages 148–156. 41, 42

Lim, N. (2015). *Estimation de modèles autorégressifs vectoriels à noyaux à valeur opérateur : application à l'inférence de réseaux*. PhD thesis, Université d'Evry Val-d'Essonne. 110, 137

Lindley, D. (1956). On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, 27(4) :986–1005. 65

Lindley, D. V. (1972). *Bayesian Statistics : A Review*. Society of Industrial and Applied Mathematics. 64

Llamosi, A., Mezine, A., d'Alché Buc, F., Letort, V., and Sebag, M. (2014). Experimental design in dynamical system identification : A bandit-based active learning approach. In *Machine Learning and Knowledge Discovery in Databases*, pages 306–321. Springer. 3, 68, 72, 129

Luersen, M. A. and Le Riche, R. (2004). Globalized nelder–mead method for engineering optimization. *Computers & structures*, 82(23) :2251–2260. 28

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, pages 281–297. Oakland, CA, USA. 95

Marbach, D., Costello, J. C., Küffner, R., Vega, N. M., Prill, R. J., Camacho, D. M., Allison, K. R., Kellis, M., Collins, J. J., Stolovitzky, G., et al. (2012). Wisdom of crowds for robust gene network inference. *Nature methods*, 9(8) :796–804. 120

Marbach, D., Prill, R. J., Schaffter, T., Mattiussi, C., Floreano, D., and Stolovitzky, G. (2010). Revealing strengths and weaknesses of methods for gene network inference. *Proceedings of the National Academy of Sciences*, 107(14) :6286–6291. 120

Mary, J. (2005). *Étude de l'Apprentissage Actif, Application à la Conduite d'Expériences*. PhD thesis, Univeristé de Paris XI. 66

Mazur, J. (2012). *Bayesian Inference of Gene Regulatory Networks : From parameter estimation to experimental design*. PhD thesis, Ruprecht-Karls-Universität. 62, 63, 64

McCallum, A. K. and Nigamy, K. (1998). Employing em and pool-based active learning for text classification. In *Proc. International Conference on Machine Learning (ICML)*, pages 359–367. Citeseer. 43

Meyer, P., Cokelaer, T., Chandran, D., Kim, K. H., Loh, P.-R., Tucker, G., Lipson, M., Berger, B., Kreutz, C., Raue, A., et al. (2014). Network topology and parameter estimation : from experimental design methods to gene regulatory network kinetics using a community based approach. *BMC systems biology*, 8(1) :13. 3, 120, 129, 130

Mezine, A., Llamosi, A., Letort, V., Sebag, M., and d'Alché Buc, F. (2015). Autonomous learning of parameters in differential equations. 3, 68, 129

Michaelis, L. and Menten, M. L. (1913). Die kinetik der invertinwirkung. *Biochem. z*, 49(333-369) :352. 152

Michalski, R. S., Carbonell, J. G., and Mitchell, T. M. (2013). *Machine learning : An artificial intelligence approach*. Springer Science & Business Media. 36

Mitchell, T. M. (1982). Generalization as search. *Artificial intelligence*, 18(2) :203–226. 41

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of machine learning*. MIT press. 36, 44

Murphy, K. P. (2012). *Machine learning : a probabilistic perspective*. MIT press. 36

Muslea, I. A. (2002). *Active learning with multiple views*. PhD thesis, Citeseer. 40

Neil D. Lawrence, Mark Girolami, M. R. G. S. (2010). *Learning and Inference in Computational Systems Biology*. MIT Press. 2

Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *The computer journal*, 7(4) :308–313. 26

Parent, É. and Bernier, J. (2007). *Le raisonnement bayésien : modélisation et inférence*. Springer Science & Business Media. 149

Pauwels, E. (2013). *Applications de l'apprentissage statistique à la biologie computationnelle*. PhD thesis, Paris, ENMP. 66, 141

Pauwels, E., Lajaunie, C., and Vert, J.-P. (2014). A bayesian active learning strategy for sequential experimental design in systems biology. *BMC Systems Biology*, 8(1). 66, 67, 129, 132

Perrin, B.-E., Ralaivola, L., Mazurie, A., Bottani, S., Mallet, J., and d'Alché Buc, F. (2003). Gene networks inference using dynamic bayesian networks. *Bioinformatics*, 19(suppl 2) :ii138–ii148. 14

Prill, R. J., Saez-Rodriguez, J., Alexopoulos, L. G., Sorger, P. K., and Stolovitzky, G. (2011). Crowdsourcing network inference : the dream predictive signaling network challenge. *Science signaling*, 4(189). 120

Pronzato, L. (2006). Sélection séquentielle de conditions expérimentales non contrôlées. *Journal européen des systèmes automatisés*, 40(2) :197–209. 2

Quach, M., Brunel, N., and d'Alché Buc, F. (2007). Estimating parameters and hidden variables in non-linear state-space models based on odes for biological networks inference. *Bioinformatics*, 23(23) :3209–3216. 21, 25, 73, 87

Raue, A., Kreutz, C., Maiwald, T., Bachmann, J., Schilling, M., Klingmüller, U., and Timmer, J. (2009). Structural and practical identifiability analysis of partially observed

dynamical models by exploiting the profile likelihood. *Bioinformatics*, 25(15) :1923–1929. 32

Raue, A., Kreutz, C., Maiwald, T., Klingmüller, U., and Timmer, J. (2011). Addressing parameter identifiability by model-based experimentation. *Systems Biology, IET*, 5(2) :120–130. 34

Reinberg, A. (1974). *Des rythmes biologiques à la chronobiologie*. Gauthier-Villars. 142

Resende, M. G., Ribeiro, C. C., Glover, F., and Martí, R. (2010). Scatter search and path-relinking : Fundamentals, advances, and applications. In *Handbook of metaheuristics*, pages 87–107. Springer. 29

Ridout, M. (1995). Three-stage designs for seed testing experiments. *Applied statistics*, pages 153–162. 66

Ristevski, B. (2013). A survey of models for inference of gene regulatory networks. *Nonlinear Anal. Model. Control*, 18(4) :444–465. 12

Robbins, H. (1985). Some aspects of the sequential design of experiments. In *Herbert Robbins Selected Papers*, pages 169–177. Springer. 47, 52

Rodriguez-Fernandez, M., Egea, J. A., and Banga, J. R. (2006). Novel metaheuristic for parameter estimation in nonlinear dynamic biological systems. *BMC bioinformatics*, 7(1) :1. 29, 30

Rolet, P. (2011). *Eléments pour l'Apprentissage et l'Optimisation de Fonctions Chères*. PhD thesis, Université Paris-Sud. 53, 79

Rolet, P., Sebag, M., and Teytaud, O. (2009a). Boosting active learning to optimality : A tractable monte-carlo, billiard-based algorithm. *Machine Learning and Knowledge Discovery in Databases*. 53

Rolet, P., Sebag, M., and Teytaud, O. (2009b). Optimal robust expensive optimization is tractable. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1951–1956. ACM. 53

Satz, H. S., K. T. H. (2001). Comparison of batch and kalman filtering for radar tracking. *Proceedings of 10th Annual AIAA/BMDO Conference*. 88

Schadd, F. C. (2009). *Monte-carlo search techniques in the modern board game thurn and taxis*. PhD thesis, Maastricht University. 52

Schena, M., Shalon, D., Davis, R. W., and Brown, P. O. (1995). Quantitative monitoring of gene expression patterns with a complementary dna microarray. *Science*, 270 :467–470. 155

Settles, B. (2010). Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66) :11. 40, 41, 42

Settles, B. (2012). Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1) :1–114. 41

Seung, H. S., Opper, M., and Sompolinsky, H. (1992). Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. 42, 42, 42

Shirt, R. W., Harris, T. J., and Bacon, D. W. (1994). Experimental design considerations for dynamic systems. *Industrial & engineering chemistry research*, 33(11) :2656–2667. 57

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587) :484–489. 3, 51

Singh, R., Palmer, N., Gifford, D., Berger, B., and Bar-Joseph, Z. (2005). Active learning for sampling in time-series experiments with application to gene expression analysis. In *Proceedings of the 22nd international conference on Machine learning*, pages 832–839. ACM. 66

Smolensky, M. and Lamberg, L. (2015). *The Body Clock Guide to Better Health : How to Use Your Body's Natural Clock to Fight Illness and Achieve Maximum Health*. Henry Holt and Company. 142

Snapp, E. (2005). Design and use of fluorescent fusion proteins in cell biology. *Current protocols in cell biology*, 21 :4. 157

Sontag, E. D. (2002). For differential equations with r parameters, 2r+ 1 experiments are enough for identification. *Journal of Nonlinear Science*, 12(6) :553–583. 1

Steiert, B., Raue, A., Timmer, J., and Kreutz, C. (2012). Experimental design for parameter estimation of gene regulatory networks. *PloS one*, 7(7) :e40052. 129

Steinhaus, H. (1956). Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci*, 1 :801–804. 95

Stolovitzky, G., Monroe, D., and Califano, A. (2007). Dialogue on reverse-engineering assessment and methods. *Annals of the New York Academy of Sciences*, 1115(1) :1–22. 120

Swameye, I., Müller, T., Timmer, J., Sandra, O., and Klingmüller, U. (2003). Identification of nucleocytoplasmic cycling as a remote sensor in cellular signaling by databased modeling. *Proceedings of the National Academy of Sciences*, 100(3) :1028–1033. 111, 111, 112, 118

Tinsson, W. (2010). *Plans d'expérience : constructions et analyses statistiques*, volume 67. Springer Science & Business Media. 56, 62

Tokic, M. (2010). Adaptive ε-greedy exploration in reinforcement learning based on value differences. In *KI 2010 : Advances in Artificial Intelligence*, pages 203–210. Springer. 141

Tokic, M. and Palm, G. (2011). Value-difference based exploration : adaptive control between epsilon-greedy and softmax. In *KI 2011 : Advances in Artificial Intelligence*, pages 335–346. Springer. 141

Touitou, Y. and Haus, E. (2012). *Biologic rhythms in clinical and laboratory medicine*. Springer Science & Business Media. 142

Vajda, S., Godfrey, K. R., and Rabitz, H. (1989). Similarity transformation approach to identifiability analysis of nonlinear compartmental models. *Mathematical biosciences*, 93(2) :217–248. 31

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, 27(11) :1134–1142. 38

Van De Schoot, R., Kluytmans, A., Tummers, L., Lugtig, P., Hox, J., and Muthén, B. (2013). Facing off with scylla and charybdis : a comparison of scalar, partial, and the novel possibility of approximate measurement invariance. *Frontiers in psychology*, 4. 149

Vanlier, J., Tiemann, C. A., Hilbers, P. A., and van Riel, N. A. (2012). An integrated strategy for prediction uncertainty analysis. *Bioinformatics*, 28(8) :1130–1135. 110

Villaverde, A. F., Henriques, D., Smallbone, K., Bongard, S., Schmid, J., Cicin-Sain, D., Crombach, A., Saez-Rodriguez, J., Mauch, K., Balsa-Canto, E., et al. (2015). Biopredyn-bench : a suite of benchmark problems for dynamic modelling in systems biology. *BMC systems biology*, 9(1) :1. 28

Villaverde, A. F., Ross, J., Morán, F., and Banga, J. R. (2014). Mider : network inference with mutual information distance and entropy reduction. *PLoS computational biology*. 137

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17(4) :395–416. 96

Wald, A. (1945). Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2) :117–186. 65

Walter, E. and Pronzato, L. (1997). *Identification of parametric models from experimental data*. Communications and control engineering. Springer. 31, 56, 56, 61, 66

Wan, E., Van Der Merwe, R., et al. (2000). The unscented kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158. IEEE. 25

Zacks, S. (1977). Problems and approaches in design of experiments for estimation and testing in non-linear models. *Multivariate Analysis 4 (P. R. Krishnaiah)*, pages 209–223. 66

Zamora-Sillero, E., Hafner, M., Ibig, A., Stelling, J., and Wagner, A. (2011). Efficient characterization of high-dimensional parameter spaces for systems biology. *BMC systems biology*, 5(1) :142. 94

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Résumé

Titre : Conduite d'expériences par apprentissage actif pour l'identification de systèmes dynamiques biologiques

Nous présentons EDEN, un algorithme d'intelligence artificielle (IA) dédié à l'estimation paramétrique d'équations différentielles ordinaires et à la planifications d'expériences pour l'identification de systèmes dynamiques biologiques. L'approche proposée est déclinée dans le cadre de l'inférence de réseaux de régulations géniques (GRN) à partir de données expérimentales. Contrairement aux méthodes classiques, la contrainte budgétaire est prise en compte, notamment grâce à un approche originale inspirée d'outils d'IA employés avec succès dans les jeux comme le GO. Les résultats numériques obtenues sur des problèmes *in silico* confortent la pertinence de notre approche. Une version autonome de EDEN parvient à rivaliser avec le meilleur des concurrents de la compétition DREAM7. Dans cette compétition, les participants disposent d'un budget limité et d'un large choix d'expériences afin d'estimer précisément les paramètres d'EDO associés à un GRN et également prédire son comportement dans des conditions expérimentales données.

Abstract

Title : Design of experiments by active learning for the identification of dynamical biological systems

We present EDEN, an artificial intelligence (AI) algorithm dedicated to the parameter estimation of ordinary differential equations and the design of experiments for the identification of biologicals dynamicals systems. The proposed approach is declined in the case of gene regulation networks (GRN) identification from experimental data. Unlike conventional methods, the budget constraint is taken into account, thanks to an approach inspired by AI tools succesfully employed in games such as GO. Many numerical results confirm the relevance of our approach through examples of *in silico* biological networks. A autonomous version of EDEN manages to compete the best performer of the DREAM7 challenge. In this challenge, participants are given a limited budget and a wide range of experiences, with which the competitors are challenged to accurately estimate the parameters of an EDO system associated to a GRN and predict the behavior of the latter for a given experimental setting.