Analyse multi-niveaux en biologie systémique computationnelle : le cas des cellules HeLa sous traitement apoptotique

Matthieu Pichené

ANNÉE 2018

UNIVERSITÉ DE RENNES 1

UNIVERSITE BRETAGNE LOIRE

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Bretagne Loire

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

Ecole doctorale MathSTIC

Matthieu Pichené

Préparée à l'unité de recherche UMR6074 IRISA
Institut de recherche en informatique et systèmes aléatoires

Analyse multi-niveaux en biologie systémique computationnelle : le cas des cellules HeLa sous traitement apoptotique.

Thèse soutenue à Rennes le 25 juin 2018

devant le jury composé de :

Denis THIEFFRY
Professeur ENS Ulm / *examinateur (président du jury)*

Fabien CRAUSTE
CR CNRS HdR, Institut des Mathématiques de Bordeaux / *rapporteur*

P.S. THIAGARAJAN
Professeur Harvard Medical School, USA. d'exercice / *rapporteur*

Olivier ROUX
Professeur Centrale Nantes / *examinateur*

Jakob RUESS
CR INRIA Saclay / *examinateur*

Anne SIEGEL
DR CNRS IRISA, Rennes 1 / *examinateur*

Blaise GENEST
CR CNRS HdR, IRISA, Rennes 1 / *directeur de thèse*

Remerciements

Je tiens tout d'abord à remercier mon directeur de thèse, Blaise Genest qui m'a encadré avec grande patience, et il en a fallu par moments. J'ai beaucoup appris grâce a lui et il a toujours su me pousser à aller plus loin.

Merci à toute l'équipe SUMO pour ces trois années passées, particulièrement à Karim Kecir qui a été un excellent soutien dans les moments difficiles de la thèse.

Merci à Fabien Crauste et P.S. Thiagarajan pour le temps qu'ils ont pris et leurs remarques sur le manuscrit.

Merci à mon jury de thèse, Fabien Crauste, Olivier Roux, Jakob Ruess, Anne Siegel et Denis Thieffry pour avoir la patience de lire ce manuscrit et évaluer ma défense.

Merci à ma famille pour me soutenir en toutes circonstances, particulièrement dans les moments durs. Et merci à mes amis pour le support moral.

Résumé

Tumeurs et cancer

Le terme cancer est utilisé pour décrire un grand nombre de maladies qui ont en commun la croissance anormale d'un groupe de cellules qui envahissent d'autres parties du corps appelé tumeur maligne [HW00]. Pour qu'une tumeur devienne maligne, les cellules doivent se reproduire anormalement et sans contrôle par l'organisme, elle doivent être immortelles et ne plus répondre aux signaux envoyés par le reste du corps. Dans cette thèse nous utiliserons le terme tumeur au lieu de tumeur maligne par soucis de concision.

TRAIL comme traitement tumoral

L'apoptose ou mort cellulaire programmée est un mechanisme controlé génétiquement, nécessaire au developpement tissulaire car il élimine les cellules superflues. Ce mecanisme est notamment nécéssaire à l'élimination des cellules dont l'ADN est endommagé et qui pourraient devenir cancéreuses [Won11].

TRAIL (Tumour Necrosis Factor-α-Related Apoptosis-Inducing Ligand) est une protéine de la famille TNF qui peut induire l'apoptose en se liant à deux recepteurs membranaires, les "death receptors" DR4 et DR5 [POC+97, CEJ+97].

La liaison de TRAIL aux death receptors induit la formation de DISC (death-inducing signaling complex) et l'activation de Caspase 8, qui elle même active deux réseaux de signalements se terminant par l'apoptose. Une étape critique de ce processus est la perméabilisation de la membrane mitochondriale externe (MOMP) qui déclenche le clivage de la protéine PARP (voir figure 1.1). La présence de PARP clivée dans la cellule est un bon indicateur de l'apoptose [MF91, NDB+97, DNM+97].

Les cellules normales ont démontrées une résistance à TRAIL élevée, contrairement a certains types de cellules cancéreuses [Sri01], faisant de celui-ci une cible intéressante pour certains traitements anti-cancer. Tous les cancers ne répondent pas positivement à TRAIL mais pour ceux qui le font les résultats sont encourageants [MLM+07].

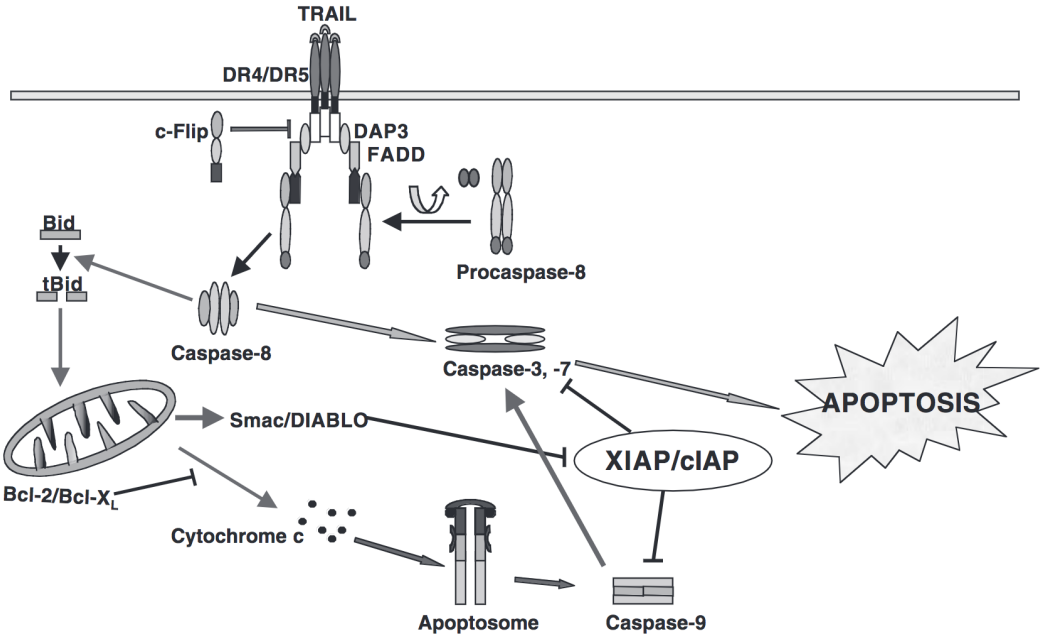


Figure 1: Représentation simplifiée de la voie biologique TRAIL

Dans cette thèse, nous examinerons spécifiquement le cas particulier de la réaction des cellules HeLa au traitement. Nous savons que, bien que la plupart des cellules HeLa soient affectées par TRAIL, certaines sont résistantes et survivent au traitement indépendamment de la concentration de TRAIL. Cette résistance à TRAIL s'est montrée transitoire et permet à une fraction des cellules de survivre à la saturation de TRAIL [FRSS13]. Des études sur le devenir des cellules sœurs ont montré que cette survie provient principalement de différences naturelles dans les concentrations de protéines régulant l'apoptose [SGA+09, RHH+09]. La résistance des cellules survivantes retourne progressivement en quelques jours à la valeur trouvée pour les cellules aléatoires en raison de la synthèse naturelle des protéines.

Modèles in silico du traitement

En raison de l'existence d'une résistance transitoire, enchaîner des traitements à grande fréquence peut être sous-optimal. D'autre part, le traitement doit être suffisamment fréquent pour dépasser la croissance naturelle de la

tumeur. Aussi, alors que TRAIL cible spécifiquement les cellules cancéreuses, il peut aussi déclencher l'apoptose dans les cellules normales [Ber16]. Pour ces raisons, il y a un intérêt à trouver la quantité et la fréquence de traitement optimales pour maximiser la mort de la lignée de cellules tumorales tout en minimisant la quantité totale de TRAIL utilisée.

Au lieu de tester directement les protocoles *in vivo*, la simulation *in silico* peut être utilisée pour tester un grand nombre de protocoles. Un petit nombre des protocoles les plus prometteurs testés *in silico* peut ensuite être testé *in vivo*. Cela permettrait d'économiser du temps et de l'argent en réduisant le nombre d'expériences *in vivo*.

Modélisation Multi-échelle

Notre objectif est de faire un modèle *in silico* qui représente fidèlement l'effet des concentrations protéiques internes sur des traitements tumoraux successifs. Deux niveaux différents doivent être considérés: Au niveau tissulaire, le modèle doit prendre en compte la croissance naturelle de la tumeur et le traitement détruisant une fraction de la tumeur. Au niveau cellulaire, le modèle doit représenter avec précision les concentrations des protéines dans la cellule et le devenir de la cellule après traitement. Il doit y avoir une identification claire de la mort de la cellule. Il a été montré que le clivage de PARP est un bon identificateur du processus d'apoptose [MF91, NDB+97, DNM+97]. Nous utiliserons donc une forte concentration de PARP clivé comme indicateur que la cellule est morte, comme utilisé par [BSDB14].

Pour étudier l'évolution tumorale sous traitement de TRAIL, nous avons donc besoin d'un modèle prenant en compte à la fois les niveaux tissulaires et cellulaires tout en évitant une explosion calculatoire due au nombre élevé de cellules individuelles à l'intérieur d'une tumeur.

Au niveau cellulaire, nous avons besoin d'un modèle qui puisse à la fois représenter l'état de la voie biologique et être facilement utilisé au niveau tissulaire. La méthode que nous développons utilise des distributions multivariées de probabilités sur les protéines. Cela permet de garder une trace de l'influence des protéines du système à l'intérieur de chaque cellule sans énumérer toutes les cellules.

Le modèle doit également être capable de simuler rapidement l'évolution du système. Pour cela nous proposons de représenter l'évolution de la dis-

tribution multivariée en utilisant une abstraction appelée réseau dynamique bayésien (DBN). On peut alors utiliser soit des simulations du DBN soit des interférences approchées, c'est-à-dire le calcul de la distribution multivariée.

Au niveau tissulaire, nous devons considérer les différences de conditions environnementales qui existent entre les différentes cellules de la tumeur (principalement leur accès au traitement et leurs contraintes de croissance). Pour les prendre en compte, nous proposons de subdiviser la population en sous-populations avec leurs propres distributions (protéiques). La croissance de ces sous-populations doit être modélisée en prenant en compte les décès dus au traitement.

Contributions

Nous avons développé une méthode utilisant des distributions multivariées pour représenter une population de cellules soumises à la voie de signalisation de TRAIL. Une représentation exacte étant intraitable, nous utilisons une représentation où les espèces biologiques sont groupées en "clusters" non disjoints. Pour ces clusters, la distribution exacte est calculée. La distribution de deux espèces liées par une chaine de clusters peut être approximée avec une précision raisonnable. Nous montrons que l'information mutuelle que partagent les variables est en grande partie conservée par cette approximation, de manière théorique aussi bien qu'expérimentale.

Nous avons developpé une méthode d'abstraction automatique d'une voie de signalisation par des réseaux bayésiens dynamiques (DBN), afin de modéliser l'évolution d'un groupe de cellules soumises à l'influence de cette voie dans le temps. Le modèle permet de modéliser efficacement l'impact d'une voie de signalisation à partir d'un sous ensemble réduit de variables (passant de 92 à 10). Ce modèle est la fondation pour un modèle multi-niveau prenant en compte l'évolution de la tumeur et l'évolution des composants d'une voie de signalisation à l'intérieur des cellules qui la composent.

Nous avons enfin développé une technique d'inférence pour les DBNs. Cette technique se distingue par son utilisation des clusters pour conserver au mieux les corrélations entre espèces biologiques d'un pas de temps à l'autre. Nous montrons que cette technique est efficace comparée aux techniques antérieures.

Nos prototypes sont à la disposition de la communauté scientifique et sont téléchargeables sur: https://perso.crans.org/~genest/D22.zip.

Abstract

This thesis examines a new way to study the impact of a given pathway on the dynamics of a tissue through Multi-Level Analysis. The analysis is split in two main parts: The first part considers models describing the pathway at the cellular level. Using these models, one can compute in a tractable manner the dynamics of a group of cells, representing it by a multivariate distribution over concentrations of key molecules. The second part proposes a 3d model of tissular growth that considers the population of cell as a set of subpopulations, partitionned such as each subpopulation shares the same external conditions. For each subpopulation, the tractable model presented in the first part can be used. This thesis focuses mainly on the first part, whereas a chapter covers a draft of a model for the second part.

Contents

Chapter 1

Introduction

1.1 Tumours and cancer

Cancer is a term defining a large range of diseases that have in common the abnormal growth of a group of cells that invade other parts of the body called malignant tumour [HW00]. For a tumour to become malignant, the cells have to reproduce abnormally and without possible control, they must be immortal and stop responding to apoptotic signals of the rest of the body. In this thesis we will use the term tumour instead of malignant tumour for short.

1.2 TRAIL as a tumour treatment

Apoptosis or programmed cell death is a genetically controlled mechanism necessary for tissue developpement through the elimination of unwanted cells. This mechanism is notably used naturally to eliminate cells with damaged DNA that could otherwise possibly lead to cancer [Won11].

Tumour Necrosis Factor-α-Related Apoptosis-Inducing Ligand (TRAIL) is a protein from the TNF family that can induce the apoptosis process by binding to two transmembrane receptors, the death receptors DR4 and DR5 [POC+97, CEJ+97]. TRAIL's binding to the death receptors leads to the formation of the death-inducing signaling complex (DISC) and the activation of Caspase 8, that activates two different signaling pathways leading to apoptosis. A critical state in this process is the mitochondrial outer membrane permeabilization (MOMP) that triggers the cleavage of the PARP protein

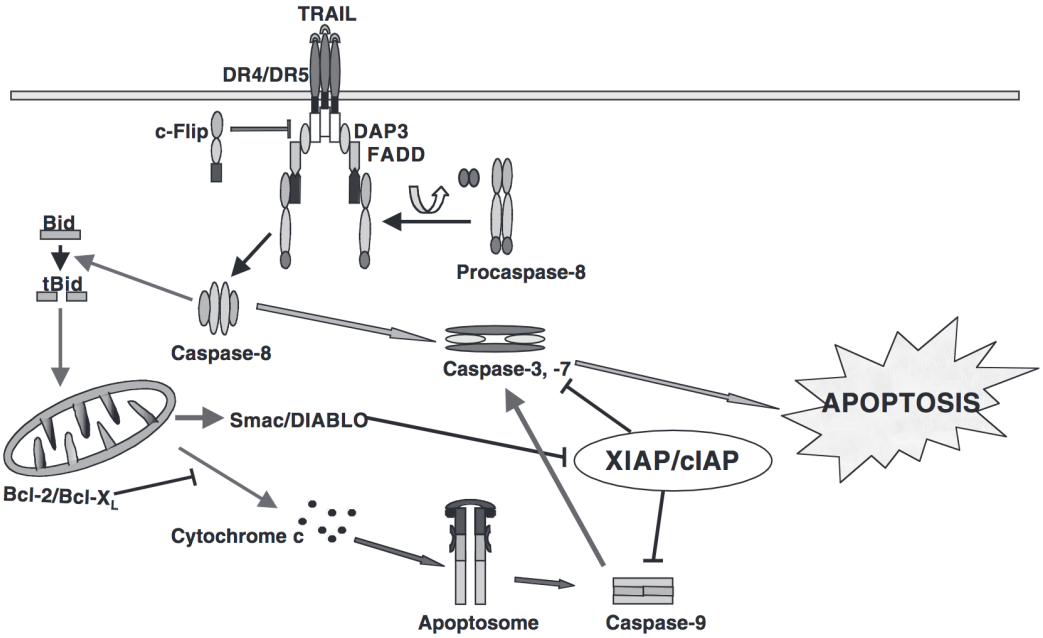


Figure 1.1: Simplified representation of the TRAIL pathway

(see figure 1.1). The presence of cleaved PARP in the cell is a good indicator that apoptosis is ongoing [MF91, NDB+97, DNM+97].

Normal cells were shown to be highly resistant to TRAIL while some tumour cells are vulnerable to its action [Sri01], making it an interesting target for some cancer treatments. Not all cancers respond positively to TRAIL but for those that do, results are encouraging [MLM+07].

In this thesis we will consider specifically the particular case of HeLa cells response to treatment. We know that while most HeLa cells are affected by TRAIL, some are resistant and survive the treatment regardless of TRAIL concentration. This resistance to TRAIL has been shown to be transient and allows a fraction of the cells to survive at TRAIL saturation [FRSS13]. Studies on the fate of sister cells have shown that this survival comes mostly from naturally occuring differences in the concentration levels of proteins regulating apoptosis [SGA+09, RHH+09]. While almost all surviving cells are resistant right after the treatment, some of them gradually lose this resistance over time through random changes in their protein concentrations. The average number of resistant cells return to it's initial value after a few days.

1.3 In silico model of the treatment

Because of the existence of a transient resistance, successive treatments with a high frequency can be suboptimal. On the other hand the treatment must be frequent enough to outpace the tumour's natural growth. Also while TRAIL targets specifically cancer cells, it can trigger apoptosis in normal cells [Ber16]. For those reasons, there is an interest in finding the optimal quantity and frequency of treatment to maximize the tumour cells line's death while also minimizing the total quantity of TRAIL used.

Instead of testing *in vivo* protocols directly, *in silico* simulations can be used to test a large number of protocols. A small number of the most promising protocols tested *in silico* can then be tested *in vivo*. This would allow to save both time and money by reducing the number of *in vivo* experiments.

In this thesis, for simplicity, we will consider unvascularized tumours. Without vascularizations, tumours can grow up to a million cells without having the core necrotizing too much due to the lack of oxygenation and nutrients. Larger tumours require vascularization to continue their growth [FH73].

1.4 Multi scale modelling

Our goal is to make an *in silico* model that faithfully represents the effect of the internal protein concentrations arising from TRAIL treatments and their effect on the resistance rate for subsequent treatments. Two different levels must be considered : At the tissular level, the model must take into account the natural growth of the tumour and the treatment killing a fraction of the tumor. At the cellular level, the model must represent accurately the concentrations of the proteins in the cell and the cell's fate after treatment. There needs to be a clear identification of the cell's death. It was shown that the cleavage of Parp is a good indicator of the apoptosis process [MF91, NDB+97, DNM+97]. We will thus use a high concentration of cleaved PARP as an indicator that the cell is dead, as in [BSDB14].

To study the tumour evolution under TRAIL treatment, we thus need a model taking into account both the tissular and cellular levels while avoiding a computationnal explosion due to the high number of individual cells inside a tumour.

At the cellular level we need a model that can both represent the state of

the pathway at any time and be easily used at the tissular level. The method we will study uses a multivariate distribution of probabilities over all the proteins. This will allow us to keep track of the influence of the proteins of the system inside every cells at once without enumerating every single cell.

The model must also be able to quickly simulate the evolution of the system. For that we propose to represent the evolution of the multivariate distribution using a dynamic bayesian network (DBN). We can then use either simulations of the DBN or approximated interence, that is computation of the multivariate distribution.

At the tissular level, we need to consider the differences in environmental conditions that exist between different cells in the tumour (mostly their access to treatment and their growth constraints). To take them into account we propose to subdivide the population in sub-populations with their own (protein) distributions. The growth of those sub-populations accounting for the deaths due to treatment has to be modeled.

1.5 Contributions

The main contribution of this thesis is a new approach to simulate the evolution of biological systems containing multiple correlated variables, such as led by the dynamics of pathways. This approach contains two main steps. First, the pathway is represented as a multivariate distribution of all the biological components considered. To avoid a combinatorial explosion in this representation, we developped a cluster based representation that allows to keep perfectly some of the main correlations and have reduced errors on other highly correlated components.

We developped an automated method of abstraction of a signaling pathway using DBNs to model the evolution of cells subject to the influence of the pathway over time. This model allows to reduce the numer of variables significantly (from 92 to 10) as well as accelerating simulations by an order of magnitude compared to ODEs.

Second, we developped an inference method for DBNs to predict the evolution of the system. DBNs have already been used to model and study biological systems leading to novel finding in immune system regulation [LZT+11]. Our approach goes further by using clusters in those DBNs to preserve the correlations between important variables ans thus increase the fidelity of the model compared to simple DBNs. The algorithm for inference

of clustered DBNs as well as the data used are available at:
https://perso.crans.org/~genest/D22.zip.

The second contribution is still a work in progress and consists in a new model for the representation of a tumor and its growth. While there exist good models of tumor growth taking into account its topology such as [WBP+15], our model has the advantage of being fast and that it considers cells not individually but as group sharing the same external conditions. This lays the ground for a future multi-level representation of a tumor where the evolution of a pathway that influence cell death like the apoptosis pathway can be modeled for each sub-population sharing the same external conditions instead of individual cells making it more tractable.

1.6 Outline

This thesis is organized as follows :

In chapter 2 we will look at the possible representations of the distribution of a large number of variables and propose a new representation allowing to preserve correlations while being tractable for a large number of variables.

In chapter 3 and 4 we will study the use of DBNs to represent the evolution of a pathway over time and new methods to increase the accuracy of the inference at each time step.

In chapter 5 we will look at the use of DBNs to model the evolution of a tumor over time that allows to decouple the complexity of the computations from the size of the tumor.

Finally in chapter 6 we will summarize our results and propose some perspectives derived from this work.

Chapter 2

Representing multivariate distributions

2.1 Introduction

When studying a process like apoptosis over a large group of cells, it is crucial to understand how the population as a whole will evolve. Slight differences in cells of the population can have a large impact in the outcome of those processes. For instance the overexpression of a single protein can entirely block a pathway. In the case of TRAIL, the pathway ultimately results in the apoptosis of the cell and we know that the overexpression of antiapoptotic proteins can prevent the apoptosis. Due to naturally occuring differences in protein production and degradations, cells inside a same population differ in protein concentrations and, in the specific case of TRAIL treatment, those variations can change the cells fate [SGA+09]. Because of that, we need to take into account this variability among cells:

Representing the particular configurations of a population can be complicated. One could consider the means of the concentrations for proteins of interest in the population of cells but this method would result in considering all cells as identical and hence with the same outcome. As an example, let's consider two cell populations A and B under TRAIL treatment shown in figure 2.1. Both have a different distribution of concentration of an antiapoptotic protein among cells but with the same average value. The antiapoptotic proteins prevent death if they are at high concentration. Populations A and B both have the same mean concentration (medium) but we can easily see

Antiapop	Pop A	Pop B
low	0%	50%
med	100%	0%
high	0%	50%
death	100%	50%

Figure 2.1: Example of the outcome of two populations with the same average concentration of an antiapoptotic protein that prevent death when at high concentration

that the outcome for A (every cell dies) is vastly different from B (half the cells survive).

The other extreme would be to consider the state of every individual cells, but this would lead to very large models hence very intensive computations. We propose instead to represent the particular configuration of a population of cells in a more tractable manner, using multivariate probability distribution. Each specie's concentration would be one random variable. A concern that arises with this approach is that, because of the large number of variables in the system, representing the exact joint distribution is still intractable.

To make this approach amenable to large systems, we explore different techniques to *approximate* the exact joint distribution over a large set of variables. In this chapter, we consider the problem of (approximately) *representing* a probability distribution, as those that appear in populations of cells governed by the same biological pathway. Beyond classical approximations, we propose to use the Chow-Liu tree representation [CL68], based on *non-disjoint* clusters of two variables.

We compare these approximate stochastic representations on different models of increasing complexities. Our experiments show that the Chow-Liu based approximation scheme is more accurate than existing ones to model probability distributions deriving from biopathways, while requiring a minimal complexity overhead.

This chapter also introduces the notations, concepts and tools that will be used throughout this thesis, including information theory concepts, as well as the pathways we will use for our *in silico* experiments.

The work presented through this chapter comes mainly from [PPFG18, PPFG17], in which I was first author.

2.2 Different representations of a multivariate distribution

As a first step, we will assume that species are described by discrete variables. We refer the reader to section 2.6 to obtain discrete values from continuous variables. Let's consider a system constituted of n different variables, each able to take s possible values. A distribution associates a probability to each possible state of the system. Exactly representing this distribution requires s^n different values. For instance, consider a three variable distribution with binary values Δ_0, with exact representation:

$$P(X_1 = 0, X_2 = 0, X_3 = 0) = 0.1$$
$$P(X_1 = 0, X_2 = 1, X_3 = 0) = 0.2$$
$$P(X_1 = 1, X_2 = 0, X_3 = 0) = 0.1$$
$$P(X_1 = 1, X_2 = 1, X_3 = 0) = 0.15$$
$$P(X_1 = 0, X_2 = 0, X_3 = 1) = 0.05$$
$$P(X_1 = 0, X_2 = 1, X_3 = 1) = 0.1$$
$$P(X_1 = 1, X_2 = 0, X_3 = 1) = 0.1$$
$$P(X_1 = 1, X_2 = 1, X_3 = 1) = 0.2$$

The exact representation becomes impossible to use realistically for large values of n. The most naive and simple approach to represent the distribution in a tractable manner is when all variables are independent. All variables are thus represented independently. Using the Δ_0 example, we would obtain :

$$P(X_1 = 0) = 0.45$$
$$P(X_1 = 1) = 0.55$$
$$P(X_2 = 0) = 0.35$$
$$P(X_2 = 1) = 0.65$$
$$P(X_3 = 0) = 0.55$$
$$P(X_3 = 1) = 0.45$$

This representation is detailed in subsection 2.2.2.

Another solution we explore consists in considering *disjoint* clusters of variables. This can be seen as an extension of the previous method where we consider joint distributions of c variables instead of single variables. The representation of the system consists in the exact representation of the correlations of the variables inside each cluster and those clusters are independent : A variable inside a cluster is independent to any variable inside any other cluster.

For Δ_0, we would obtain with clusters $\{X_1, X_2\}$ and $\{X_3\}$:

$$P(X_1 = 0, X_2 = 0) = 0.15$$
$$P(X_1 = 1, X_2 = 0) = 0.2$$
$$P(X_1 = 0, X_2 = 1) = 0.3$$
$$P(X_1 = 1, X_2 = 1) = 0.35$$
$$P(X_3 = 0) = 0.55$$
$$P(X_3 = 1) = 0.45$$

This approximation is detailed in subsection 2.2.3.

Last, it is possible to allow variables to be in multiple clusters. This can allow to partially keep the correlations between the species inside the clusters that share the protein.

For Δ_0, we would obtain with clusters $\{X_1, X_2\}$ and $\{X_1, X_3\}$:

$$P(X_1 = 0, X_2 = 0) = 0.15$$
$$P(X_1 = 1, X_2 = 0) = 0.2$$
$$P(X_1 = 0, X_2 = 1) = 0.3$$
$$P(X_1 = 1, X_2 = 1) = 0.35$$
$$P(X_1 = 0, X_3 = 0) = 0.3$$
$$P(X_1 = 1, X_3 = 0) = 0.25$$
$$P(X_1 = 0, X_3 = 1) = 0.15$$
$$P(X_1 = 1, X_3 = 1) = 0.3$$

This approximation is detailed in subsection 2.2.4.

2.2.1 Formalism

We now formalize the different representations : We assume a set $X = \{X_1, ..., X_n\}$ of random variables over the indices $I = \{1, ..., n\}$ (for instance concentrations of molecules). We assume that these variables take discrete values in the same set V of values (for instance, $V = \{$very low,low,medium, high,very high$\}$). In our case, the size $|V|$ of V would be small, typically around 5, while the number of variables n would be larger, typically around 30. For a subset $J \subseteq I$ of indices, we denote by $\vec{X}_J$ the tuple of variables $\{X_j | j \in J\}$, and by $\vec{x}_J = (x_k)_{k \in J}$ a tuple in V^J. A *distribution* P over X is a function $P : V^I \to [0, 1]$ such as $\sum_{\vec{x} \leq V^X} P(\vec{x}) = 1$. We denote by $P(\vec{x}_J) = P(\vec{X}_J = \vec{x}_J)$ the probability that the tuple of variable $\vec{X}_J$ takes the tuples of values $\vec{x}_J$. By definition, we have $P(\vec{x}_J) = \sum_{\{x_i | i \notin J\}} P(\vec{X} = \vec{x})$ with $\vec{x}_i = x_i$ for all $i \notin J$ and $\vec{x}_i = \vec{x}_{Y,i}$ for $i \in J$. This operation is called *marginalization*.

The exact representation of a probabilistic distribution over the set of variables X with values in V involves $|V|^{|X|}$ values (each encoding a probability in $[0, 1]$). We call such values *joint probabilities*.

2.2.2 Fully factored representation

The first approximated representation assumes that the joint probability is equal to the product of individual probabilities. In this way, it suffices to keep only the marginal probability for each species, that is $|V| \cdot |X|$ values. More precisely, the representation is a function $Q : I * V \to [0, 1]$ that follows the rule: $Q(i, x_i) = P(X_i = x_i)$.

We call such an approximation *fully factored*, and denote it by P_{FF} It represent the distribution:

$$P_{FF}(\vec{x}_X) = \prod_i P(X_i = x_i) \tag{2.1}$$

The main interest of this approximation is its really low complexity.

Obviously, on the other hand, with such a scheme, correlations between variables are lost. It is not too hard to understand that such representation will not represent faithfully the original distribution in cases where correlations are not negligeable. For most pathways, as we will see in the section 2.8 the concentrations of proteins can be highly correlated. We now propose representations that allows to consider some of these correlations.

2.2.3 Disjoint cluster representation

We describe now an approximation taking into account the most important correlations: The disjoint cluster representation. In this representation, we group variables that share information into disjoint clusters. For each cluster, the joint distribution of the variables in the cluster is represented. A variable inside a given cluster is considered independent to every variable outside of the cluster.

One could thus think of disjoint clusters of correlated values, relations between disjoint clusters being handled in the succint fully-factored form. Assuming that each cluster is of size m and c is the number of clusters ($c = \frac{|X|}{m}$), the representation Q is a function $Q : c * V^m \to [0, 1]$ that follow the rule: $Q(c, \vec{x}_{K_c}) = P(\vec{X}_c = \vec{x}_{K_c})$ We call such approximation the *disjoint clustered* approximation.

Assuming a known clusterisation $(K_j)_{1 \leq j \leq c}$ with c clusters we obtain

$$P_{\text{cluster}}(\vec{x}_X) = \prod_{j \leq c} P(\vec{x}_{K_j}) \tag{2.2}$$

This gives a representation using $\frac{|X|}{m} \cdot |V|^m$ values.

It can be noted that increasing the cluster size quickly cause the representation to be intractable. Also the limit cases of this representation are Fully Factored for $m = 1$ and the exact distribution for $m = |X|$. A major problem of the representation is that even if the strongest correlations are preserved, most correlations are lost. In cases where there are chains of correlations, every possible clusterisation (with clusters smaller than the chains) would break those chains.

2.2.4 Non-disjoint cluster representation

In general, there are correlation between almost each species involved in a biological pathway (which we will confirm in Section 2.8). As the fully-factored and disjoint clustered approximated representations impose no correlation between most of the species, a lot of information is lost using these approximated representations. We thus propose to use c *non-disjoint clusters* $(K_j)_{1 \leq j \leq c}$, allowing to keep some correlations between each species. The representation Q is the same as disjoint clusters: $Q : c * V^m \to [0, 1]$ that follow the rule: $Q(c, \vec{x}_{K_c}) = P(\vec{X}_c = \vec{x}_{K_c})$.This representation keeps $c \cdot |V|^m$

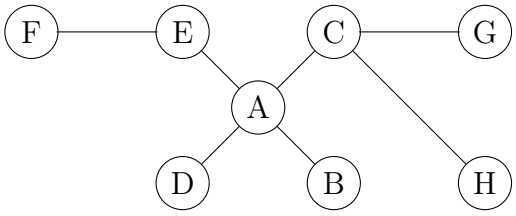


Figure 2.2: Graph forming an unrooted tree. There is only at most a single path linking two given nodes.

values, that is $|V|^m$ values for each of the c clusters. Notice that the number c of clusters may be more than $\frac{N}{m}$ as clusters are non-disjoint.

One cannot reuse the formula of $P_{cluster}$, as variables can be used in different clusters, and their contribution would be counted several times. Instead, one has to discount the contribution of a variable of a cluster when it has already be counted for a previous cluster. For simplicity, throughout this chapter we will only consider clusters of size 2. We will not allow loops between clusters, meaning there will be at most one path to link two species through clusters. If we consider a graph with species as nodes and clusters as edges, because loops are forbidden, this graph forms a tree (or a forest).

Formally, the undirected graph can be written as $G = (I, E)$ in the following manner: vertices are defined by I with $I = \{1, ..., n\}$, and the edge $\{j, k\}$ is present in E iff there exists some cluster K_i such that $\{j, k\} \in K_i$.

Graph G is a *cluster tree* (see Figure 2.2) iff there is no subgraph with a sequence $S = i_1, ..., i_n \in I$, s.t. $\forall j \in S$, $\{j, j+1\} \in E$ and $\{i_n, i_1\} \in E$

The approximated probability distribution, denoted P_{NDC} (for non-disjoint clusters forming a tree) is the following:

$$P_{\text{NDC}}(X = x) \quad = \quad \prod_{i=1}^{c} \frac{P(\overrightarrow{X_{K_i}} = \overrightarrow{x_{K_i}})}{\prod_{j \in K_i^{old}} P(X_j = x_j)} \tag{2.3}$$

where $K_i^{old} = K_i \cap (\cup_{j<i} K_j)$ represents variables of K_i already present in $K_1 \cup ... \cup K_{i-1}$.

Equivalently, denoting C_i the number of clusters containing i, we have:

$$P_{\text{NDC}}(X = x) \quad = \quad \frac{\prod_{i=1}^{c} P(\overrightarrow{X_{K_i}} = \overrightarrow{x_{K_i}})}{\prod_{i=1}^{N} P(X_i = x_i)^{C_i - 1}} \tag{2.4}$$

Notice that (2.3) generalizes (2.2) that already generalizes (2.1). What is true on P_{NDC} thus also applies for P_{FF} and P_{cluster}.

Proposition 1 *P_{NDC} defined in (2.3) for a cluster tree is a proper probability distribution over X, i.e. $\sum_{x \in V^X} P_{NDC}(X = x) = 1$. Moreover, P_{NDC} does not depend on the ordering of clusters used in (2.3).*

Proof We first prove that any permutation does not affect the value of P_{NDC}. By induction, we can derive that the second statement of the proposition hold true for any order of clusters.

Let i the index at which the permutation occurs. Let's consider the two possible cases for permutations :

-First case: the clusters, that we will call K_i and K_{i+1}, don't have a variable in common. We can assume without loss of generality that $K_i = \{1, 2\}$ and $K_{i+1} = \{3, 4\}$.

$P_{\text{NDC}}(X = x) = P1 * \frac{P(X_1=x_1, X_2=x_2)}{Q_1 * Q_2} * \frac{P(X_3=x_3, X_4=x_4)}{Q_3 * Q_4} * P2$, with

$P1$ the part of the product before those two clusters, and $P2$ the part after.

$$Q_k = \begin{cases} P(X_k = x_k), & \text{if } k \in \cup_{j<i} K_j \\ 1, & \text{otherwise} \end{cases}$$

Q_1 and Q_2 depend on $\cup_{j<i} K_j$. It is important to note that because $K_i \cap K_{i+1} = \emptyset$, Q_3 and Q_4 also depend only on $\cup_{j<i} K_j$.

After permutation we get :

$P1 * \frac{P(X_3=x_3, X_4=x_4)}{Q3 * Q4} * \frac{P(X_1=x_1, X_2=x_2)}{Q_1 * Q_2} * P2 = P_{\text{NDC}}(X = x)$

because $K_i \cap K_{i+1} = \emptyset$, Q_1 and Q_2 don't change.

-Second case: the clusters, that we will call $C1$ and $C2$, have a variable in common. We can assume without loss of generality that $K_i = \{1, 2\}$ and $K_{i+1} = \{2, 3\}$:

$P_{\text{NDC}}(X = x) = P1 * \frac{P(X_1=x_1, X_2=x_2)}{Q_1 * Q_2} * \frac{P(X_2=x_2, X_3=x_3)}{P(X_2=x_2) * Q_3} * P2$,

because X_2 is present in K_i.Also because X_1 and X_3 ar only present in one of those clusters, $Q1$ and $Q3$ are not changed by the order.

After permutation we get :

$P1 * \frac{P(X_2=x_2, X_3=x_3)}{Q_2 * Q_3} * \frac{P(X_1=x_1, X_2=x_2)}{Q_1 * P(X_2=x_2)} * P2 = P_{\text{NDC}}(X = x)$

Which is equal to the first equation.

Let's now consider a particular ordering of nodes: the one obtained using breadth first search starting from the root of the tree. This gives an ordering

for clusters, starting with $K_1 = \{1, 2\}$, etc. Because of the previous proof, the first statement beeing true with this ordering is also true for any order. Now, simply observe that each generic term in (2.3) is the conditional distribution of X_{K_i} given $X_{K_i^{old}}$. Indeed, each K_i^{old} is made of one indice, but for $K_1\{1, 2\}$ which have K_1^{old} empty. Hence $P_{\text{NDC}}(X = x) \in [0, 1]$ for all x. Now we can easily prove by recursion on the number $n \geq 2$ of variables that $\sum_x P_{\text{NDC}}(X = x) = 1$: For $n = 2$, we have $\sum_{x_1,x_2} P_{\text{NDC}_1}(X_1 = x_1, X_2 = x_2) = \sum_{x_1,x_2} P(X_1 = x_1, X_2 = x_2) = 1$

Else, consider $n+1$ variables. X_{n+1} is a leaf, with parent some k. Let $J = \{1, ..., n\}$. Thus $P_{\text{NDC}_{n+1}}(\vec{X} = \vec{x}) = P_{\text{NDC}_n}(\vec{X}_J = \vec{x}_J) * \frac{P(X_k=x_k, X_{n+1}=x_{n+1})}{P(X_k=x_k)}$.

We can easily see that if we marginalize on X_{n+1} : $\sum_{x_1,...,x_{n+1}} P_{\text{NDC}_{n+1}}(\vec{X} = \vec{x}) = \sum_{x_1,...,x_n} P_{\text{NDC}_n}(\vec{X}_J = \vec{x}_J) * \sum_{x_{n+1}} \frac{P(X_k=x_k; X_{n+1}=x_{n+1})}{P(X_k=x_k)}$. Now observe that $\sum_{x_{n+1}} P(X_k = x_k; X_{n+1} = x_{n+1}) = P(X_k = x_k)$ (this is a marginalization), and hence the sum is equal to $\sum_{x_1,...,x_n} P_{\text{NDC}_n}(\vec{X}_J = \vec{x}_J)$ which is 1 using the induction hypothesis.

We can conclude that $P_{\text{NDC}_{n+1}}(\vec{X} = \vec{x})$ is a distribution of all n. □

2.3 Handling P_{NDC} with low complexity

A priori (2.3) is not straightforward to use. We show here that it can however be used with low complexity when careful algorithms are used. More precisely, we show that it's possible to compute the distribution of a subset of variables in a reasonable amount of time, without computing the whole variables distribution.

Consider a subset $J \subset I$ of indices. Formally, given $x_s \in V$ for all $s \in J$, we define $P_{\text{NDC}}(\vec{x}_J) = \sum_{x_t | t \notin J} P_{\text{NDC}}(\vec{x}_X)$, the marginalisation of $P_{\text{NDC}}(\vec{x}_X)$ on J.

To compute $P_{\text{NDC}}(\vec{x}_J)$ with low complexity, we proceed as follows: for all node s at depth i, we denote J_s the descendants of s that are in J, plus $\{s\}$ if it is not in J. We compute $P_{\text{NDC}}(\vec{x}_{J_s})$ inductively: given t a child of s, from the computation of $P_{\text{NDC}}(\vec{x}_{J_t})$, one can simply obtain $P_{\text{NDC}}(\vec{x}_{J_t \cup \{s\}})$ by applying the formula (2.3):

$$P_{\text{NDC}}(\vec{x}_{J_t \cup \{s\}}) = \frac{P_{\text{NDC}}(\vec{x}_{J_t}) \cdot P(X_s = x_s, X_t = x_t)}{P(X_t = x_t)}$$

We can do the same for every child t of s. Let $T_t = J_t \cup \{s\}$ if $t \in S$, and

$T_t = J_t \cup \{s\} \setminus \{t\}$ otherwise. We can compute $P_{\text{NDC}}(\vec{x}_{T_t})$ from $P_{\text{NDC}}(\vec{x}_{J_t \cup \{s\}})$ by marginalizing out t when t is not in S.

From there, one can compute $P_{\text{NDC}}(\vec{x}_{J_s})$, by joining together all the $P_{\text{NDC}}(\vec{x}_{T_t})$ and dividing $d-1$ times by $P(X_s = x_s)$, where d is the number of children of s in the tree using (2.4):

$$P_{\text{NDC}}(\vec{x}_{J_s}) = \frac{\prod_{t \text{ child of } s} P_{\text{NDC}}(\vec{x}_{T_t})}{P(X_s = x_s)^{d-1}}$$

Hence, we can compute all probability values over a subset J of n variables. This computation can thus be done in a reasonable amount of time, $|X|$ times the number $|V|^n$ of values to compute.

Proposition 2 *Let $J \subseteq I$. Then one can compute $P_{NDC}(\vec{x}_J)$ in time $O(|X| \cdot |V|^{|J|})$ all possible tuples $\vec{x}_J \in V^J$* [1].

2.4 Information Theory

In order to quantify correlation strength to better select clusters plus related tasks we will perform in chapter 3, we now recall some basic tools from information theory.

2.4.1 Entropy

First, we introduce Shannon's Entropy [SW49]. Entropy is an obvious candidate to mesure the information contained in a distribution. It is a good tool to mesure the quantity of information of a variable.

The entropy $H(X)$ of a K-valued discrete random variable X is:

$$H(X) = -\sum_{x \in X} p(x) log_K(p(x))$$

For $p(x) = 0$, we use the convention $p(x)log_K(p(x)) = 0$.

For instance, for a variable X, if the discretization scheme perfectly splits the data such that each of the K valuations has equal probability, then its entropy $H(X) = -K * (1/K) * log_K(1/K) = 1$. In the worst case, all the data points are concentrated in a single interval of the random variable, and the entropy is 0.

[1] To obtain this complexity, we initially reroot the tree with a variale of J at the root. This ensures that the root is not marginalized out.

2.4.2 Mutual information

We often want to quantify the strength of correlations of pairs of variables. To that effect we will use *Mutual Information* (*MI* for short), which is commonly used in information and probability theory. It gives a quantitative measure of the correlation between two variables. Mathematically, mutual information evaluates how similar the joint distribution between two random variables compared to the product of the marginal probabilities of the individual variables. This metric has attractive information-theoretic interpretations and can be used to measure non-linear associations.

In formal terms, given two discrete K-valued random variables X and Y, the mutual information $MI(X;Y)$ between X and Y is given by:

$$\mathrm{MI}(X,Y) = \sum_{y \in V_Y} \sum_{x \in V_X} p(x,y) log_K \left(\frac{p(x,y)}{p(x)p(y)} \right)$$

When a variable Z is known, it can be helpful to know how much information is shared between two variables X and Y on top of the information already given by knowing Z. Formally, given three discrete random variables X,Y and Z, the mutual information $MI(X;Y|Z)$ between X and Y conditioned on Z is given by:

$$MI(X;Y|Z) = \sum_{z \in V_Z} \sum_{y \in V_Y} \sum_{x \in V_X} p(x,y,z) log_K \left(\frac{p(x,y,z)p(z)}{p(x,z)p(y,z)} \right)$$

where V_R is the set of values of variable $R \in \{X, Y, Z\}$, and $p(x,y,z)$ is the joint probability that $X = x$, $Y = y$ and $Z = z$. We will use this in chapter 3.

2.4.3 Kulback-Leibler divergence

To mesure the variation between a distribution and its approximation we can use the Kulback-Leibler divergence.

Given two distributions P and Q of K-valued variables, we quantify their differences using the Kulback-Leibler divergence with the following formula :

$$KL(P,Q) = \sum_{\vec{x} \in V^X} P(\vec{X} = \vec{x}) \log_K \frac{P(\vec{X} = \vec{x})}{Q(\vec{X} = \vec{x})}$$

We consider a group of clusters optimal if using those clusters gives a lower or equal Kulback-Leibler divergence with the original distribution than any other cluster group.

2.5 Obtaining optimal clusters

A main problem with the clustered representation (disjoint or not) is to obtain optimal clusters, that is clusters for which P_{NDC} will approximate as well as possible the real probability distribution P. In that respect, the Chow Liu algorithm [CL68] allows to compute an optimal tree (actually, a forest in general, that is with possibly several roots).

The Chow-Liu algorithm uses mutual information in order to select the clusters, as described in Algo. 1. A tree is considered optimal if the sum of the mutual information shared inside all clusters is superior or equal to that sum for any other tree. This will imply that, if the Kulback-Leibler divergence of those clusters with the original distribution is lower of equal than the Kulback-Leibler divergence for any other group of clusters forming a tree.

More formally, let $T = (X, E)$ an undirected acyclic graph, with X the set of variables. Let $S = \{x \mid \neg\exists y, \{x, y\} \in E\}$, that is the set of nodes with no neighbours. The clusters associated with T are the pairs $\{x, y\} \in E$, plus the set S of singletons.

Let K_T be the clusters associated with the tree T. We can compare the approximation P_T of distribution P obtained using K_T and the approximation P_S obtained with the set of clusters associated with any another tree S. Prop. 3 states the optimality of the tree built using the Chow-Liu algorithm: its Kulback-Leibler divergence KL is optimal.

Algorithm 1: Chow Liu Algorithm. Computes an optimal tree of clusters.

For each pair $\{X_i, X_j\}$ in X, compute $MI(X_i, X_j)$.
Sort edges $\{i, j\}$ by decreasing value of $MI(X_i, X_j)$.
Starting with an empty graph as tree T, repeat:
- consider the next edge $\{i, j\}$ in the list
- if $\{i, j\}$ does not close a cycle in T, add it to the tree

Proposition 3 *The probability distribution $P^*_{NDC,2}$ derived by (2.4) from a Chow-Liu tree $T = (X, E)$ satisfies*

$$KL(P, P^*_{NDC,2}) = \min_{P_{NDC,2}} KL(P, P_{NDC,2}) \tag{2.5}$$

Moreover, one has

$$KL(P, P_{FF}) = KL(P, P_{NDC,2}) + KL(P_{NDC,2}, P_{FF}) \tag{2.6}$$

The proof of the first point can be found in [CL68]. The second point derives from [Csi75]. Observe that it holds for all cluster tree approximations of P, which proves that $P_{\mathrm{NDC},2}$ is always a better approximation of P than P_{FF}.

Augmenting the size of clusters is always beneficial. For example, let $I = K_1 \cup ... \cup K_c$ be a cluster tree with clusters of size 3, let $G_3 = (I, E_3)$ be the triangulated graph defined by clusters (maximal cliques) K_i, and let us further assume that the K_i are adequately ordered. Let $P_{\mathrm{NDC},3}$ be the probability distribution associated to G_3 by (2.3). For any tree T forming a subgraph of G_3, and the associated distribution $P_{\mathrm{NDC},2}$, one has

$$\begin{aligned} & KL(P, P_{\mathrm{NDC},2}) \\ = \; & KL(P, P_{\mathrm{NDC},3}) + KL(P_{\mathrm{NDC},3}, P_{\mathrm{NDC},2}) \end{aligned} \tag{2.7}$$

which generalizes and complements (3). So adding edges to a Chow-Liu tree approximation of P can only improve the resulting approximation of P. However, there is no simple procedure that would give the optimal G_3 graph: this problem was proved to be NP-complete. Nevertheless, greedy procedures generalizing the Chow-Liu algorithm can perform well [Mal91].

2.6 Entropy-based discretization

Protein concentrations are often described with continuous variables. In this section we will discuss the ways to discretize the variables representing each specie into a limited number of states.

A simple and common strategy to discretize a variable is to get an estimate of the minimal (usually 0) and maximum value it can take, and to partition this range into equal sized intervals, henceforth called *uniform discretization* (see *e.g.* [LHT11]). In our case, each variable describes the concentration level of a biochemical species. The 5 ranges will be called very low,

low, medium, high and very high. This scheme has therefore the advantage of being easily interpretable biologically.

Alternatively we can use *entropy* as defined in section 2.4.1 to analyse the quality of the uniform discretization: If the entropy is close to 1, the uniform discretization is a good candidate. On the other hand, if the entropy is close to 0 it mean the uniform discretization is a bad representation. For cases where the uniform discretization represent badly the system we can choose to use an equal size discretization instead: that is using ranges that possess the exact same weight. This discretization garantee that all ranges contain information. It can result though in ranges of very different sizes and in some case some ranges can be disproportionately different.

Another way to make a trade off between optimizing size and optimizing weight is to use the Lloyd Max algorithm [Llo82, Max60].

2.7 Biological Pathways

In this section, we present the pathways on which we will perform our experiments on. Those are presented from the smallest ot the largest. The enzyme catalysis is a simple tool we used to test rapidly our model. The EGF-NGF pathway presents the advantages of being well known and is mainly used to verify that our algorithm work on multiple pathways and isn't overfitted for the apoptosis pathway.

2.7.1 Enzyme Catalysis

$$S + E \underset{k_2}{\overset{k_1}{\rightleftharpoons}} ES \xrightarrow{K_3} E + P$$

Figure 2.3: Enzyme catalytic reaction

The simple enzyme catalytic system is shown in Fig. 2.3. It describes a typical mass action based kinetics of the binding (ES) of enzyme (E) with substrate (S) and its subsequent catalysis to form the product (P). The value space of each species (variable) is divided into 5 equal intervals. The time scale of the system is 10 minutes which was divided into 100 time points.

This reaction possess the advantage of containing only 4 components meaning it's possible to compute its exact distribution for each time point in a reasonable amount of time. This allows to compare our methods to the exact computation which is not possible on larger pathways.

2.7.2 Abstracted Apoptosis Pathway

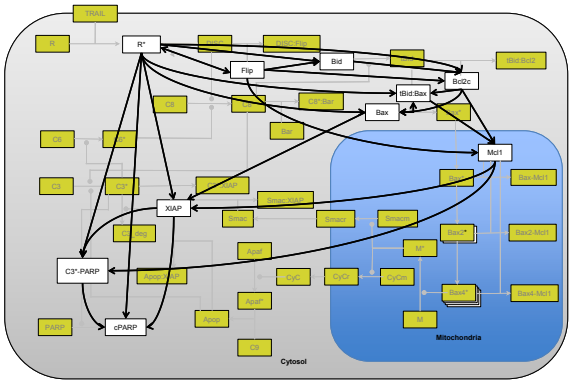


Figure 2.4: Apoptosis pathway. In white are shown the 10 proteins used for the abstracted pathway.

TNF-related apoptosis-inducing ligand (TRAIL), an apoptosis inducing protein in cancer cells, has been considered as a target for anti-cancer therapeutic strategies. Biological observations on HeLa cells suggest that in a population of cells, TRAIL application only leads to fractional killing of cells. Further, there is a time dependent evolution of cell resistance to TRAIL. We consider an abstraction, consisting of 10 (out of 58) protein variables (shown in white in fig. 2.4). The time horizon of the model is the first 90 minutes period after injection of TRAIL, which was divided into 22 time points. Each variable can take 5 possibles values discretized using the trade off shown in section 2.6. We will explain in chapter 3 how this abstraction is obtained automatically (see also [PBP+17]). One of the most important species to track in that pathway is cPARP, which is an indicator of the apoptosis [CAB].

2.7.3 EGF-NGF Pathway

The EGF-NGF pathway describes the behavior of cells to EGF or NGF stimulation [BHC+04]. EGF (Epidermal Growth Factor) is involved in cellular differenciation and proliferation while NGF (Nerve Growth Factor) is involved in the growth and proliferation of nerve cells specifically [TMC72, LC56]. It consists of 32 variables (one for each molecular species). The value domains of the 32 variables were divided into 5 uniform intervals as described in section 2.6. The time horizon of each model was assumed to be 10 minutes which was divided into 100 time points.

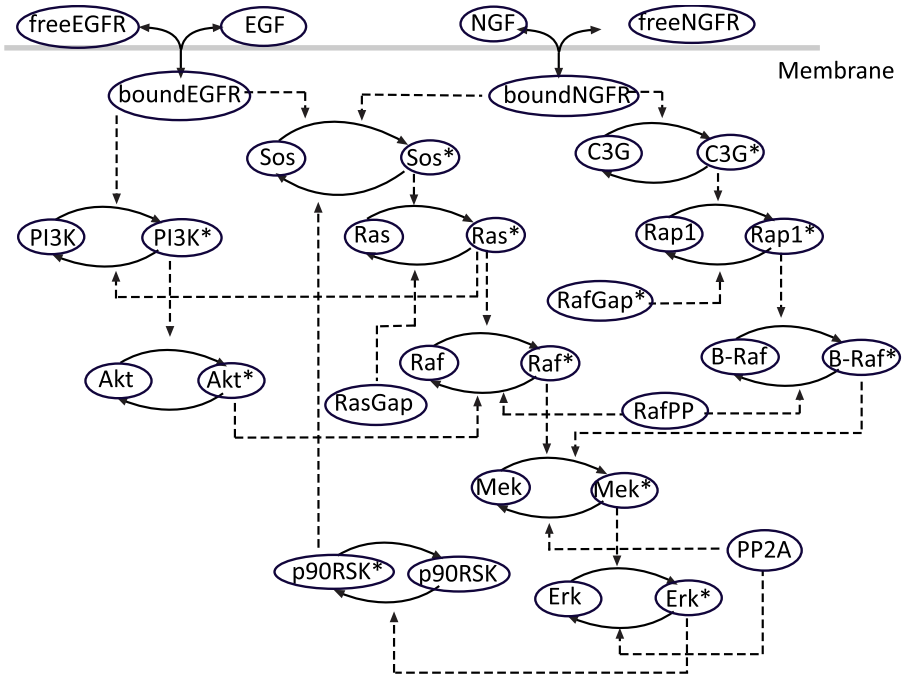


Figure 2.5: EGF-NGF pathway

2.8 Results

Our algorithm is implemented in Python. All experiments were performed on an Intel i7-4980HQ processor (2,8 GHz quad core Haswell with SMT)

with 16 GB of memory. For each of the pathway case study discussed in the previous section, we consider the probability distributions at an arbitrarily chosen time point. As one cannot compute the exact joint probability for large systems, we evaluate them considering the mutual information between any pair of variables (computed from 10.000 simulations of the system), to understand where correlations are lost. We use Prop. 2 to compute MI values for the Tree Cluster representation.

We explain these numbers in more detail in the following subsections.

2.8.1 Enzyme Catalysis

The system is very simple with only 4 variables. The tree obtained using the Chow-Liu algorithm is the same over all time points, with $\{\{E, S\}, \{E, P\}, \{E, ES\}\}$ as set of non-disjoint clusters. To compare with a disjoint cluster representation, we chose the set of disjoint clusters with highest mutual information, that is $\{\{E, S\}, \{ES, P\}\}$. On this example, in addition to computing the largest difference in MI, we provide the maximum difference of the probability of joints and the Kullback-Leibler divergence as the system is small enough to compute them. Fig. 2.9 shows the approximated correlations obtained using the different approximations at an arbitrarily chosen time point (corresponding to 2 minutes).

Fig.2.7 shows the measures at that time of the system. It can be seen that our Tree Cluster representation manages to preserve most of the mutual information (of the original distribution, 0.277 of 0.278) between variables, which translates to minimal error on computed probabilities (< 0.005). It is important to note that the case of Disjoint Clusters while better than FF, is still short of capturing all the dependencies faithfully in the distributions: it considers independent a pair (out of only 16) variables with $MI = 0.11$ (the maximum correlations between two different variables have $MI = 0.27$). This results in a probability error 10 times higher (0.05) than using Tree Cluster, which is small but significant for such a trivial example.

2.8.2 Abstracted Apoptosis Pathway

We display on Fig. 2.9 the approximated correlations obtained using the different approximations for an arbitrary time point (corresponding to 30 minutes). Fig.2.8 shows the statistics for this time point. Our Tree Cluster approach captures most of the mutual information between variables (0.1 out

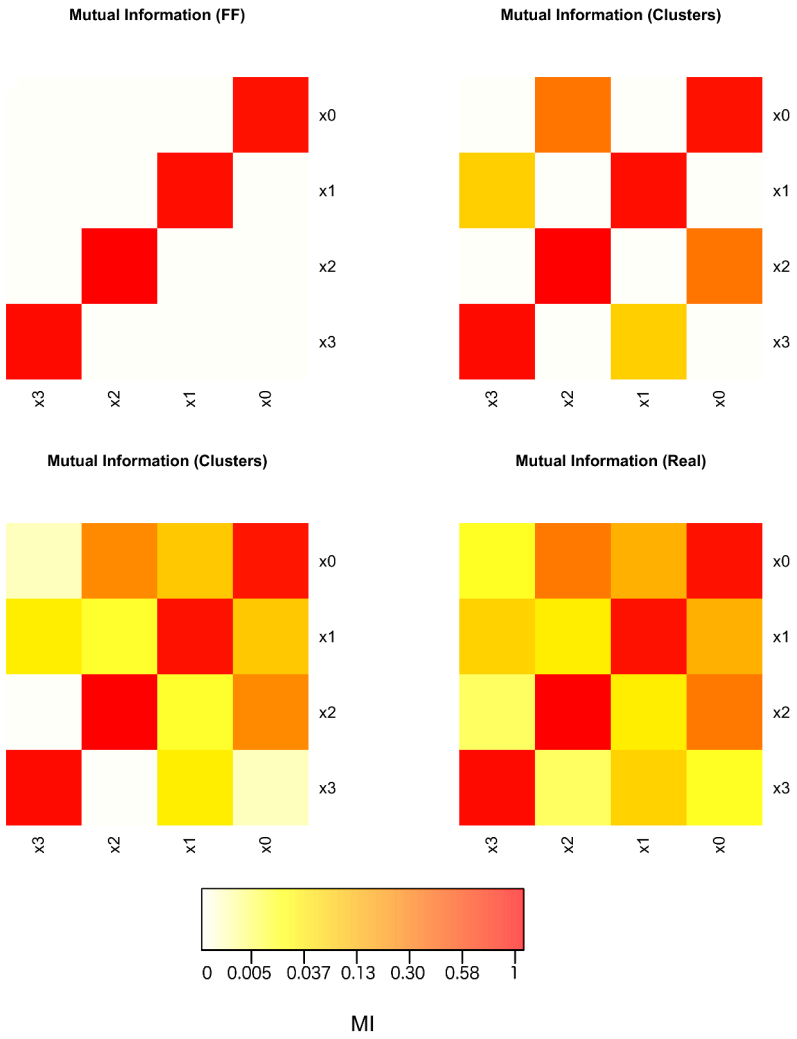


Figure 2.6: Mutual information shared by the 4 variables using different approximations. In order : Fully factored, Disjoint clusters, Tree cluster. The last representation is the real mutual information shared.

Representation	Mean MI	max MI Error	Max P error	KL diverg.
FF	0.22	0.27	0.22	0.31
Disjoint Cluster	0.26	0.11	0.05	0.12
Tree Cluster	0.277	0.04	0.005	0.001
Exact	0.278	0	0	0

Figure 2.7: Table representing the errors of approximations w.r.t. the real distribution for the enzyme catalytic reaction's probability distribution at 2 minutes.

of 0.12), and the maximum error on the MI is the least (0.12) compared to FF or Disjoint Clusters $(0.2, 0.32)$. Also, the size of the representation does not increase too much (225 values vs 125 or 50).

Fig. 4.4 shows the two trees computed by algorithm 1 [CL68] at the arbitrarily chosen time of 20 and 90 minutes. Most links of the tree follow direct correlations, except for the link Bid-cPARP at 90 minutes. Our interpretation is that, at 90 minutes, Bid does not play much of a role anymore, and its correlation is not meaningful. Further, Bax, Bcl2c and Mcl1, which are highly correlated, and which transduce or inhibit the signal are connected towards the downstream only through R^*. The reason is that the correlation with R^* is higher than the direct correlations, and as we produce a tree, the direct correlations are removed by the algorithm. Notice that this interaction graph can change in time (compare at time 20 and 90 when Bid swaps from one side of the tree to the other side).

Representation	mean MI	max MI Error	Size of representation
FF	0.06	0.32	50
Disjoint Cluster	0.08	0.2	125
Tree Cluster	0.1	0.12	225
Exact	0.12	0	10^7

Figure 2.8: Table representing the errors of approximations w.r.t. the real distribution for the Apoptosis pathway's probability distribution at 30 minutes.

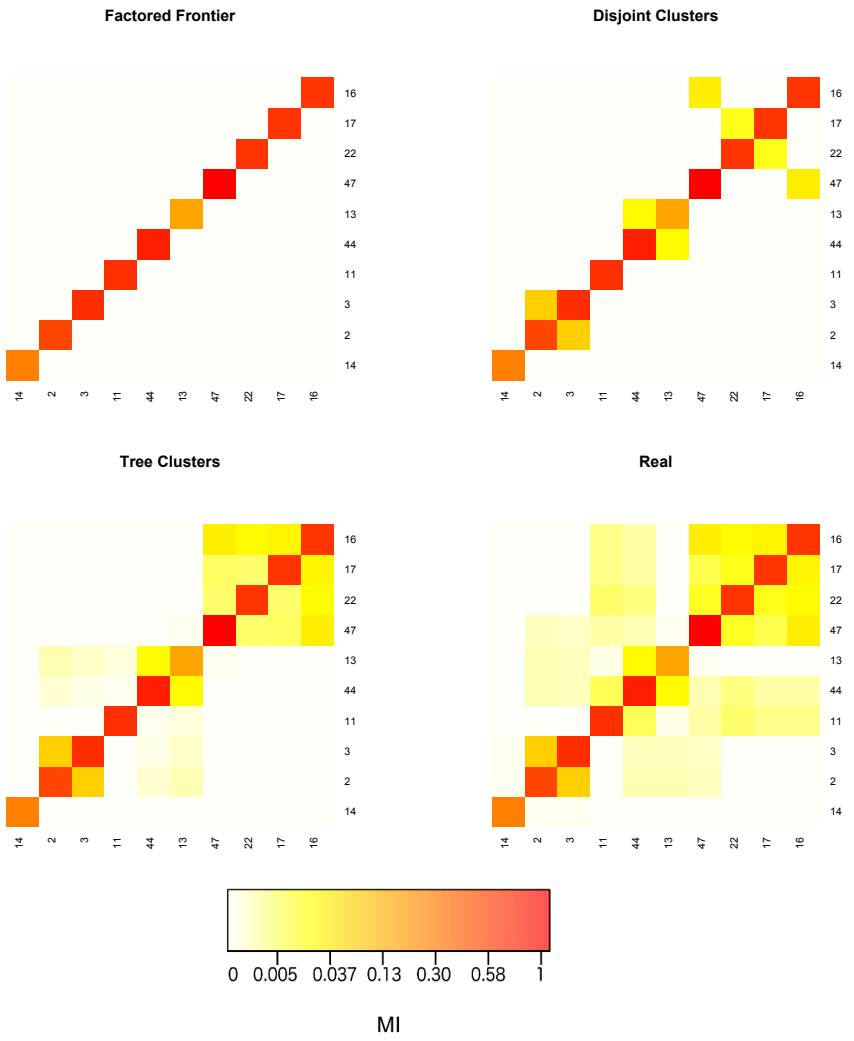


Figure 2.9: Mutual information shared by the variables using different approximations. In order : Fully factored, Disjoint clusters, Tree clusters. Then is shown the real MI.

2.8.3 EGF-NGF Pathway

In case of the EGF-NGF pathway, we consider a biologically reasonable set of disjoint clusters, grouping a species with its activated form, as their concentrations are very correlated. We display on Fig. 2.11 the approximated correlations obtained using the different approximations at time 5 minute of the EGF-NGF pathway. Tree Cluster manages to keep some correlations among almost every pair of variables, which is not the case for FF or Disjoint clusters, assuming independence of almost every pair of variables. The loss of information is also minimal, as confirmed by Fig.2.10 (MI error ≤ 0.07).

Representation	mean MI	max MI Error	Size of representation
FF	0.016	0.6	160
Disjoint Cluster	0.019	0.2	400
Tree Cluster	0.023	0.07	775
Exact	0.026	0	10^{22}

Figure 2.10: Table representing the errors of approximations w.r.t. the real distribution for the EGF-NGF pathway's probability distribution at 5 minutes

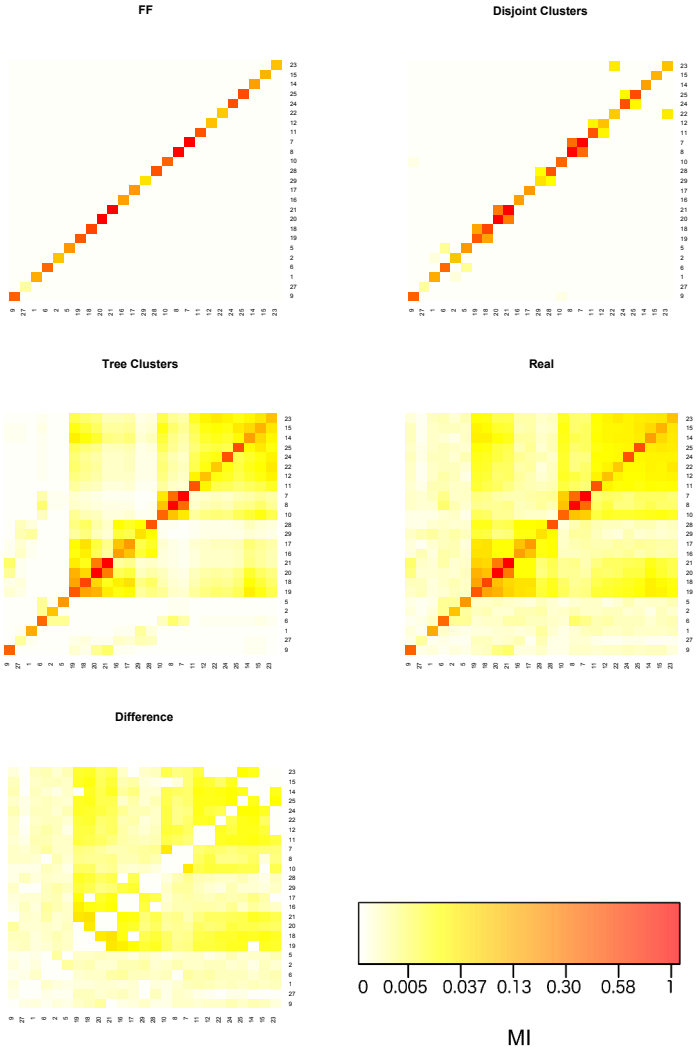


Figure 2.11: Mutual information shared by the variables using different approximations. In order : Fully factored, Disjoint clusters, Tree clusters. Then are showed the real MI as well as the difference between Tree Cluster and the real MI for clarity.

2.9 Conclusion

The tree cluster representation we presented in this chapter is a good candidate to represent the distribution of large biological systems where components are deeply connected. The use of this representation can be extended to other problems where the relations between a high number of components must be represented. For the problem at hand: The modelization of the evolution of acquired resistance to TRAIL, this representation allow to keep track of the important correlations between the key proteins involved in the resistance. We will see in the next two chapters how we can model the evolution of the system over time.

Chapter 3

Cellular level: Modeling the evolution of multivariate distributions

3.1 Introduction

Beyond the modeling of the distribution, our goal is to be able to model the evolution of a population of cells with respect to a specific pathway. To do this, several solutions are possible:

The system can be modeled as ODEs (see next section) which is a fine-grain model: Results are precise but the computation is slow, particularly for large systems. To use ODEs with our multivariate distribution we need to generate randomly thousands of simulations from the initial distribution. It is then possible to compute the probabilities to be in each cluster configuration at each time point as we will see in the next chapter.

Another possibility that we will explore is to use DBNs, a coarse-grained abstraction, instead. We will present how to obtain a DBN abstraction from an ODE model. In this chapter we will show that simulating DBNs is faster that simulating the fine-grained model it abstracts. The results of this chapter have been published in [PBP+17, PPB+16b, PBP+15] on which I collaborated.

3.2 Ordinary Differential Equations

In order to model the complex behavior and interactions of signaling networks, researchers have proposed multiples computational models [KL06].

A set of Ordinary Differential Equations (ODEs) is one of the main modelling frameworks used to describe the evolution of a biological system when it can be reduced to a set of competitive reactions. It has the following features : It is a deterministic model that describes precisely the evolutions of multiple variables. It is a fine-grained model meaning that it describes the evolution of the system after an infinitesimal time period. Concretely, to numerically integrate ODEs generally require time steps of the order of the second to represent accurately the results of multiple competitive chemical reactions. In the case of biological systems, the evolution of each possible biochemical component can be modeled as a differential equation making ODE models easy to generate even for large systems and is precise as long as every componant stays in large enough quantity.

For instance let us consider a simple biological reaction of Fig. 3.1.

Simulating ODEs requires short time steps to account for the influences of each microvariation on the system and stay accurate. On the other hand, we want to be able to study the behavior of the system over weeks. Using ODEs is possible but would be very time consuming. Also it is important to note that to obtain the new distribution we would require multiple simulations (in the order of thousands) for the new distribution to be significative.

It is easy to see that for those reason, the use of ODEs would require billions of steps and be intractable for a system studied during multiples weeks.

We propose instead a coarse grain model, that is a model describing the

$$S + E \underset{k_2}{\overset{k_1}{\rightleftharpoons}} ES \xrightarrow{K_3} E + P$$

The underlying ODEs are :
$\frac{dS}{dT} = -k1 * S * E + k2 * ES$
$\frac{dE}{dT} = -k1 * S * E + k2 * ES + k3 * ES$
$\frac{dES}{dT} = k1 * S * E - k2 * ES - k3 * ES$
$\frac{dP}{dT} = k3 * ES$

Figure 3.1: Simple enzymatic reaction modeled as an ODE.

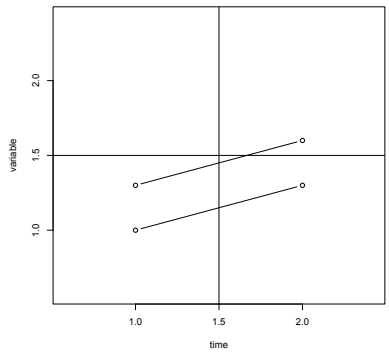


Figure 3.2: Two simulations of ODEs of a variable that evolves linearly over time. After discretization for the DBN (represented by the horizontal and vertical separators), the lower one will give stay in the interval 1, while the second goes from the interval 1 to 2. The CPT of the corresponding DBN will give at time 1 the variable 50% of chance to keep the value 1 and 50% to take the value 2.

evolution of the system with time steps of the order of minutes. This model uses dynamic bayesian networks (DBNs).

3.3 The Dynamic Bayesian Network model

Using a discrete coarse-grain model has two advantages over ODEs :

- Because these allow to use time steps several orders larger than the reaction times, we can use a smaller number of time points and it is faster to generate a simulation.
- Because they use directly discrete values, the same as the ones that represent the distribution of clusters. Hence one simulation of a discrete model correspond to many simulations of ODEs.

When the range of variables is quantized in a discrete set of intervals, the dynamics can be stochastic, even when it represents a deterministic dynamic,e.g. given by ODEs. Consider the following case represented in fig 3.2:

- Two ODE configurations $c1$ and $c2$ at time point t.

- They result in configurations $d1$ and $d2$ at time $t+1$.
- $c1$ and $c2$ lie in the same quantized configuration
- $d1$ and $d2$ lie in different quantized configurations

The result of that is that the same configuration can result in multiple different configurations. Stochasticity allow to deal with this.

3.3.1 Definition of DBNs

DBNs are a type of compact Markov chain. In order to obtain a compact representation, the values taken by a variable at a given time depend only of the values of a small subset of variables at the previous time point.

Throughout this chapter, we will use the following notations :

Let $\{X_1, \ldots, X_n\}$ be a set of ordered random variables. $\{0, 1, \ldots, T\}$ a set of time points. Let i and j two number such as $1 \leq i \leq n$ and $1 \leq j \leq n$. We denote X_i^t the variable X_i at time t.

A DBN is a graph model representing a set of random variables. Every variable X_i at a given time point t depends on a fixed set of variable at time $t-1$ that will be called parents Pa of X_i.

Formally we represent a DBN as the following structure $\mathcal{D} = (\mathcal{X}, T, Pa, (CPT_i^t)_{1\leq i\leq n}^{t\leq T})$ where :

- T is a positive integer with t ranging from a set of time points $\{0, 1, \ldots, T\}$.
- $\mathcal{X} = \{X_i^t \mid 1 \leq i \leq n, 0 \leq t \leq T\}$ is the set of our variables. For instance, X_i^t can be the variable representing the concentration of a protein at time t.
- Each variable X_i at each time t is associated with a set of parents $Pa(X_i^t)$. Parents must satisfy the following properties :

  - $Pa(X_i^0 = \emptyset)$

  This mean that at time 0, variables don't have parents.

  - If $X_j^{t'} \in Pa(X_i^t)$ then $t' = t-1$.

  For any other time point t the variables possess a group of parent variables at $t-1$.

- If $X_j^{t-1} \in Pa(X_i^t)$ for some t then $X_j^{t'-1} \in Pa(X_i^{t'})$ for every $t' \in \{1, 2, \ldots, T\}$.

  A given variable X_i will keep the same group of parents for all $t > 0$. For instance in figure 3.3, the variable X_1 has X_0 and X_1 as parent except for $t = 0$ where the variable has no parents.

- For all $t \leq T$, $1 \leq i \leq n$, CPT_i^t is the conditional probability table (CPT) for variables i at time point t as described below.

The links between parents and children are associated with the edges of the graph. Nodes at the $t-1$ time point are connected to nodes at the t time point, and this remains invariant as t ranges over $\{1, 2, \ldots, n\}$. The regular structure of a DBN induces the function PA over variables given by: $X_j \in PA(X_i)$ iff $X_j^{t-1} \in Pa(X_i^t)$ for some t. We will denote $\hat{\imath}$ the variables parent of i that is we define $\hat{\imath} = \{j \mid X_j \in PA(X_i)\}$.

Conditional Probability Tables

We call CPTs the table, associated with each variable for each time point, that specifies the probability that a variable takes a given value knowing the values of all its parents. The table is noted CPT_i^t for the variable X_i^t. For $t \geqslant 1$, this CPT specifies the probabilities $P(X_i^t \mid X_{\hat{\imath}}^{t-1}))$.

$$CPT_i^t(x_i \mid \vec{y}_{\hat{\imath}}) = P(X_i^t = x_i \mid X_{\hat{\imath}}^{t-1} = \vec{y}_{\hat{\imath}})$$

Thus $CPT_i^t(x_i \mid \vec{y}_{\hat{\imath}}) = p$ specifies that p is the probability of $X_i = x_i$ at time t given that at time $t-1$, $X_{j_1} = y_{j_1}, X_{j_2} = y_{j_2}, \ldots, X_{j_m} = y_{j_m}$ with $\hat{\imath} = \{j_1, j_2, \ldots, j_m\}$.

The semantics of a DBN is the following: from a configuration $\vec{y}$ at time $t-1$, the probability to move to configuration $\vec{x}$ at time t is:

$$CPT^t(\vec{X} = \vec{x} \mid \vec{Y} = \vec{y}) = \prod_{i=1}^{n} CPT_i^t(x_i \mid \vec{y}_{\hat{\imath}})$$

That is it represents the heterogenous Markov chain $(S, \delta)^t$ with $t \leq T$, a set of configuration $S = V^x$ and a stochastic function of transition $\delta^t(\vec{y}, \vec{x}) = CPT^t(\vec{X} = \vec{x} \mid \vec{Y} = \vec{y})$.

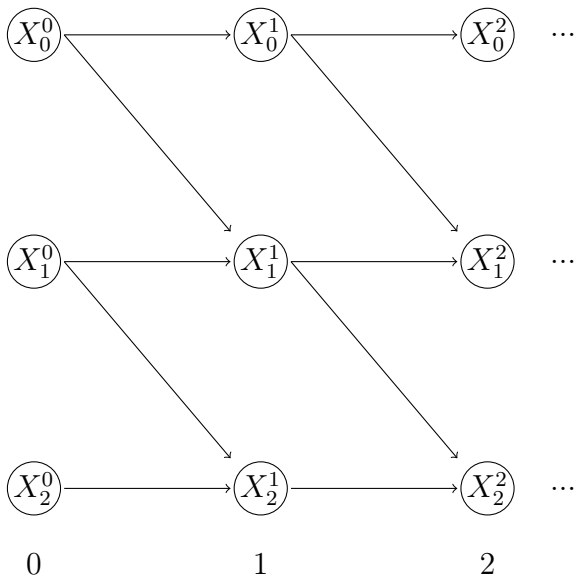


Figure 3.3: Representation of a simple DBN. Here for instance, the variable X_1 at time 1 depends on the variables X_0 and X_1 at time 0

$$
\begin{aligned}
&CPT(X_1^1 = 0 | X_1^0 = 0, X_0^0 = 0) = 0.7 \\
&CPT(X_1^1 = 1 | X_1^0 = 0, X_0^0 = 0) = 0.3 \\
&CPT(X_1^1 = 0 | X_1^0 = 0, X_0^0 = 1) = 0.6 \\
&CPT(X_1^1 = 1 | X_1^0 = 0, X_0^0 = 1) = 0.4 \\
&CPT(X_1^1 = 0 | X_1^0 = 1, X_0^0 = 0) = 0.3 \\
&CPT(X_1^1 = 1 | X_1^0 = 1, X_0^0 = 0) = 0.7 \\
&CPT(X_1^1 = 0 | X_1^0 = 1, X_0^0 = 1) = 0.1 \\
&CPT(X_1^1 = 1 | X_1^0 = 1, X_0^0 = 1) = 0.9
\end{aligned}
$$

Figure 3.4: Representation of the CPT of a two state variable X_1 at time $t = 1$ depending on two variables, itself and X_0 at time $t = 0$.

3.4 From ODEs to DBNs

The work in this section mostly come from [LHT11]. To obtain DBNs from ODEs we need to define :

1. The time steps
2. The set of variables to consider
3. The discretization of variables
4. The initial conditions
5. The parents relations
6. The conditional probability tables

Liu et al. proposed a method to generate DBNs from ODEs [LHT11]. The required values are obtained as follow :

1. Time steps:

   Time steps are chosen by the user, limited to time points of importance. These are either points for which we have experimental data or that we want to consider. The fewer time points the faster the DBN simulation.

2. Set of variables to consider:

   The variables considered in the DBN are the concentrations of all the biological species in the system. As a side note, the original paper also consider unknown constants as variables of the DBN to be able to infer their value.

3. Discretization of variables:

   The range for values each variable can take is quantized in a set of intervals $I = \{I_0, ..., I_{K-1}\}$ with K as the number of intervals for the variable. To determine those ranges, minimum and maximum values are statistically determined for all variables. Intervals are then determined using uniform discretization of section 2.6. For complexity reasons, k is typically = 5: {very low, low, medium, high, very high} concentrations.

4. Initial conditions:

   The initial conditions for the DBNs are the discretized probability distribution of the initial conditions of the variables taken *independently*.

5. Parents relations:

   Parents relations are derived directly from the ODE equations : Each variable's parents are the set of variables appearing in the differential equation describing its evolution including the variable itself.

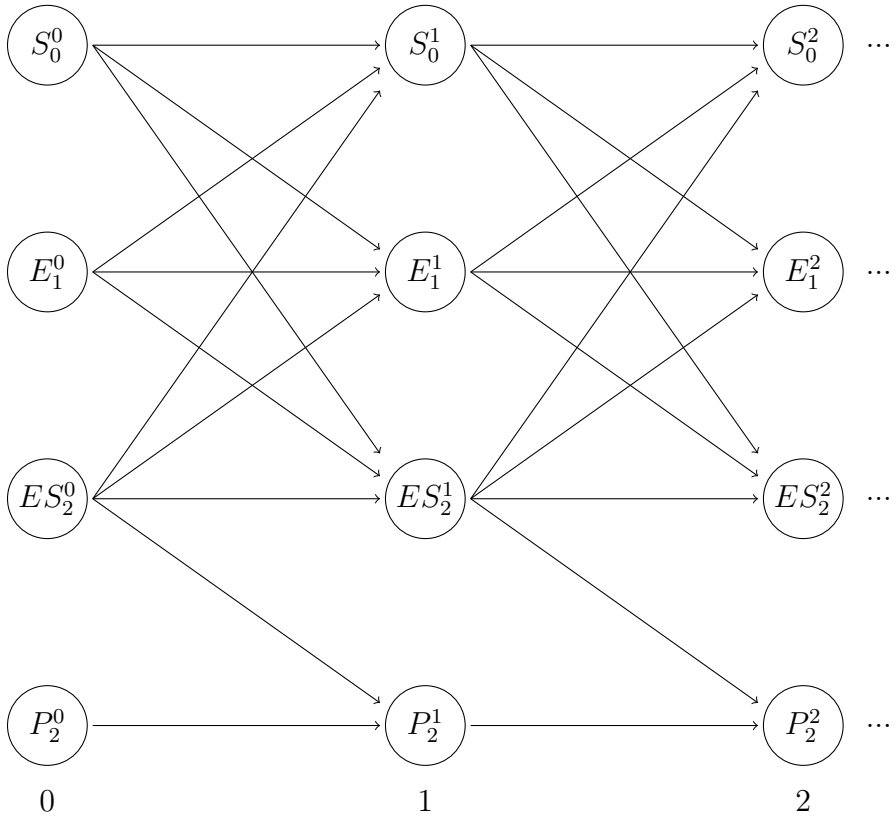
6. Conditional probability tables:

   Millions of ODE simulations are drawn. We denote $nb(Cond)$ the number of simulations satifying the conditions $Cond$. For a given variable the CPTs are approximated by the following formula applied to the results obtained from ODE simulations :

$$CPT_i^t(x \mid \vec{y}) = \frac{nb(X_i^t = x, X_{\hat{i}}^{t-1} = \vec{y})}{nb(X_{\hat{i}}^{t-1} = \vec{y})}$$

   Notice that having $nb(X_{\hat{i}}^{t-1} = \vec{y}) = 0$ creates singularities that we will discuss in section 4.1. We will consider in a first step that $nb(X_{\hat{i}}^{t-1} = \vec{y}) > 0$ for all $i \in \mathcal{X}$ AND $t < T$.

If we use the example from section 3.2 we have the following set of ODEs:
$\frac{dS}{dT} = -k1 * S * E + k2 * ES$
$\frac{dE}{dT} = -k1 * S * E + k2 * ES + k3 * ES$
$\frac{dES}{dT} = k1 * S * E - k2 * ES - k3 * ES$
$\frac{dP}{dT} = k3 * ES$
[LHT11] transform them into the following DBN:



The parents for a given variable in the DBN are all the species that have a direct impact in its evolution in the ODE plus itself. For instance here P's production depend only on the concentration of ES, his parents are for all time points > 0 ES and P. This can be written as : $\widehat{ES} = \{ES, P\}$

There are some notable issues with using directly [LHT11], in particular for the apoptosis pathway:

- *Uniform discretization:* The issue with using the uniform discretization is that for some variables (e.g. polymerized molecules with non linear dynamics), the range of values is very limited (e.g. with extremely small values), except for rare outliers with extreme values. In such cases, the uniform discretization would bundle almost all the values together in a single discretized interval, and be almost useless. The more different cases fall in the same interval, the more simulations are used to generate the same CPTs, with a wide range of values. If the lower values of an interval have notable different effects compared to

higher values, the CPTs will allow a wide range of results reducing the precision of the simulation, see fig. 3.5.

- *Choice of parents:* The choice of parents proposed by [LHT11] can be limiting. First, variables can have many parents (In particular, the C8 protein in the apoptosis pathway has 9 parents using the usual method). The efficiency of the DBN approach (both in terms of space complexity and simulation time) scales down exponentially with the number pa of parents of a variable. Formally, the size of a CPT is $O(k * |V|^{pa})$, where $|V|$ is the number of values for each variable, and k the number of variables. Second, the time elapsed between time points can be several order of magnitude larger than the time needed for reactions to occur. Thus choosing parents based on other correlations than direct reactions can give better results.

3.5 DBNizer method

In this section we will decribe the methods that were used in the tool **DBNizer** [PBP+17]. We will describe our strategy to obtain a discrete probabilistic structure that represents a system dynamics. In practice, our algorithms will take as inputs a large set of trajectories sampled at discrete time points. We will assume that this set represents all the relevant dynamics of the original system.

Compared to the original method by Liu et al., we will use the following method to build the DBN:

1. The time steps:
   The time steps follow the same rules as [LHT11].

2. Set of variables:
   In order to improve efficiency, we consider explicitely only a subset of of variables, selected using mutual information (MI) as we will see in section 3.5.2

3. The uniform discretization of variables is replaced by a Lloyd-Max discretization when needed, as we will discuss in subsection 3.5.1

4. The initial conditions follow the same method as [LHT11].

5. The parents relations are not based on reactions but evaluated using medium term impact as we will discuss in subsection 3.5.3

6. The conditional probability tables are built using the method in [LHT11].

The key steps of **DBNizer** include the identification of the most important variables and the inference of the most informative local dependencies between them. To do so, we will rely on well-established information theory notions. **DBNizer** automatically constructs the full DBN structure. First it generates a large number of trajectories of the underlying biochemical model by numerical integration of the original model. Second, it selects the suitable subset of variables to be used for DBN construction and infers their edge relationships. Third, it calculates probabilities for each entry in the CPTs through simple counting (as in [LHT11]). It also automatically considers model refinement through iterative improvements.

As a first step, given that we are interested in a discrete abstraction of the underlying system, we will describe how to discretize the values of model variables using entropy. After this, we will explain how using mutual information, our algorithm chooses a small subset of the most relevant variables of the original system and then infers a directed graph of the most important influences between them.

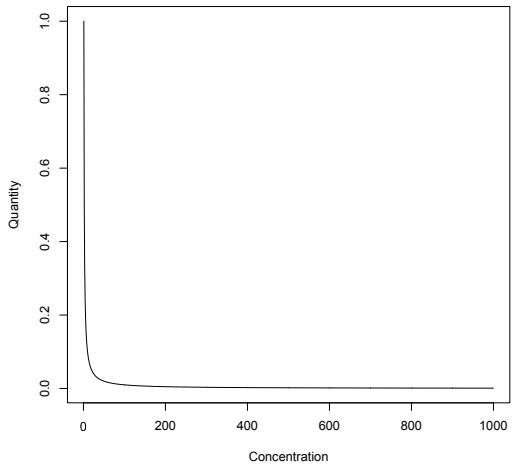
3.5.1 Discretization in DBNizer

As seen in Section 2.6, a choice can be made between uniform discretization and entropy based discretization.

We use entropy to check the effectiveness of the uniform discretization. Only for variables X with an acceptable level of entropy (we chose $H(X) \geq 0.4$ for the apoptosis pathway), we stick with the uniform discretization.

For variables where uniform discretization have a low entropy (< 0.4), we resort to an alternate quantization algorithm which automatically discretizes these variables with the goal of maximizing the entropy. For this, we first sample enough simulations of the HSD model to obtain a histogram of values for each variable, over all time points. Based on these histograms, we partition the values to have an equal number of samples in each interval (or as close as possible). This ensure that the entropy is 1 (or close to 1). On top of that, we use the Lloyd-Max [Llo82, Max60] discretization algorithm which minimizes the distortion (quadratic distance of the samples and their discrete

a.



b.

Uniform		Equal size	
Range	Weight	Range	Weight
1-200	0.785	1-2	0.2
201-400	0.093	3-11	0.203
401-600	0.054	12-510	0.198
601-600	0.038	51-224	0.199
801-1000	0.030	225-1000	0.2

c. d.

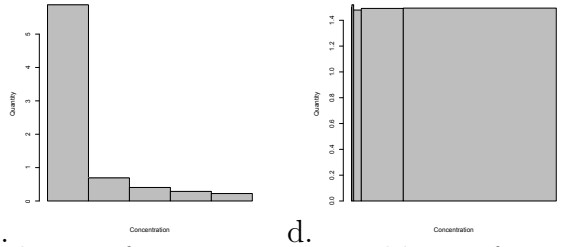


Figure 3.5: a. Evolution of a continuous variable y in function of x. b. Shows the range obtained for a uniform and an equal size discretization as well as the weight of those ranges. Figures c. and d. are a visual representation of uniform and equal size discretization respectively.

values). The downside of this method is that the biological interpretation can be lost. Additionally, this increases the size of the internal representation of transition probabilities in the abstract model. These are the reasons we use it only for variables where uniform discretization results in a low entropy.

3.5.2 Inferring key random variables

In high-dimensional dynamical systems, not all variables convey the same quantity of information on the dynamics. In order to define a set of important variables, we propose to use mutual information between variables and the signal of interest. In the apoptosis pathway, we are interested in the cell fate (dynamics of death). The latter is modeled as a binary variable D: it represents if the cell was alive or dead 8 hours after exposing the cell to 250 ng/ml TRAIL (death is defined by a cPARP concentration threshold of 100000 units in the HSD model [BSDB14]).

We first compute for each variable the mutual information between its initial configuration and the cell fate. A simple way of choosing relevant variables is to select the k variables having the highest mutual information w.r.t the signal of interest (here D). However, doing so is not necessarily a good choice. For instance, assume that variables X and Y have very similar dynamics. Hence $MI(D;X)$ and $MI(D;Y)$ are very similar (say high). However, selecting both variables $\{X,Y\}$ is not efficient as the value for (X,Y) does not bring much more information than say X alone. This can be automatically detected by considering $MI(D;Y|X)$, which would be much lower than $MI(D;Y)$. In the extreme case where $X = Y$, we have $MI(D;Y|X) = 0$.

Consequently, we adopted the following scheme. First we select the variable, say X_1, having the highest mutual information with D, $MI(D;X_1)$, and hence being the most important variable for its effect on death. Then we select the second variable, say X_2, as the one that, conditioned on X_1, has the highest mutual information with D, $MI(D;X_2|X_1)$. We continue this for k steps until the mutual information, given by $MI(D;X_k|X_1 \cdots X_{k-1})$, is sufficiently low. At this stage we stop our computation.

While this identifies variables (which have an initial value) that have a considerable impact on death, it may miss some key intermediate variables important for conveying the signal of death. We explained at the begining of this section how to find such variable.

3.5.3 Inferring local dependencies between key variables

A crucial step towards constructing an accurate abstraction is to find the set of important local dependencies between the identified variables. Our strategy to find those is again to rely on mutual information. Let $V = \{v_1....v_k\}$

be the k identified variables, and $T = \{0, 1....t-1\}$ be the discretized set of t time points.

Direction of local dependencies

We first build the directed graph based on the reaction network, $G_{RN} = (V_{RN}, E_{RN})$. The set of vertices of G_{RN} is the set of variables of the original system. Edges of G_{RN} are defined with respect to the reaction network of the biological system: $(X, Y) \in E_{RN}$ if the concentration of X influences the concentration of Y in some reaction, that is, if X appears in the differential equation of Y. For instance, if we have a one-way reaction that produces Z from X and Y, then X, Y and Z will be vertices, and (X, Z), (Y, Z) and (X, Y) and (Y, X) will be edges of G_{RN}. Once we have constructed the reaction network graph, we build the graph G_V^+ over the set of vertices V made of the selected variables $\{v_1....v_k\}$. In G_V^+, we have an edge (v_i, v_j) iff v_j is reachable from v_i in G_{RN}. Stated differently, G_V^+ is the vertex-induced subgraph of the transitive closure of G_{RN}.

Selecting important parents

We chose to limit the number of parents to 4 per variable. Increasing the number of parents does not improve the accuracy much while slowing down the simulations, while decreasing it reduces the accuracy [PBP+17].

Graph G_V^+ reflects potential dependencies between selected variables. However, not all are useful (either because they are negligeable or redundant). To obtain a smaller, but still sufficiently informative set of local dependencies between variables, we define $G_{MI} = (V, E_{MI})$ by refining G_V^+ using mutual information. Denoting E_v the predecessors of $v \in V$ in G_V^+, and $Z_1^m, \ldots, Z_k^m, Y^m, V^{m+1}$ the random variables of $z_1, \ldots, z_k, y$ at time m and v at time $m+1$, we define:

$$M(Y; V|Z_1, \ldots, Z_k) = \max_{m \in T} MI(Y^m; V^{m+1}|Z_1^m, \ldots, Z_k^m)$$

For each key variable v, we select the variables influencing v iteratively, as in Section 3.3. First, we select the variable, $z_1 \in E_v$, having the highest value for $M(Z; V)$. Then we select $z_2 \in E_v$ as the variable having the highest $M(Z; V|Z_1)$, and iterate until we have the desired number of parents.

3.5.4 Choice of additionnal variables

There may be additional variables, not identified in Section 3.5.2, that are important for the transduction of signal. Adding these variables in the DBN can further improve its accuracy. For this, we iteratively find additional variables v and build an associated DBN to assess how important they are. We rank a DBN according to the weighted mean difference between the simulation outputs of the original biochemical model and the DBN. More precisely, to select an additional variable on top of a set V of variables, we rank the DBN $MIDBN_{V\cup\{v\}}$ automatically built with set of variables $V\cup\{v\}$, for every $v \notin V$. We select v optimizing the rank of $MIDBN_{V\cup\{v\}}$. We iterate from $V' = V \cup \{v\}$, and stop when no additional variable really improves the accuracy. The exact subset of variables chosen by this iterative discovery is sensitive to parameters of our tool. This is because several variables carry similar information. Indeed, different choices do not impact the results much

3.6 Computational results

In this section we will outline our key experimental results to compare the different models according to different metrics (time per simulation, accuracy, etc.). Unless stated otherwise, we consider treatments with 250 ng/mL of TRAIL and simulate cell behaviors for 8 hours after treatment. The concentration of *cPARP* was used as an indicator of cell death. "Observations" (i.e. time points) were available every 2 minutes for the first 30 minutes, and every 15 minutes for the subsequent 7.5 hours. We used 100000 simulations of the original HSD model to populate the CPT entries of the DBNs. Using more simulations does not improve the accuracy of the DBNs (Table 3.1). All experiments were carried on a quad core 2.8Ghz Intel Xeon E5-1603 CPU with 8 GB RAM.

We will consider DBNs obtained by our tool DBNizer, as described in the next subsection. For comparison, we consider DBN *RNDBN*, defined using the technique advocated in [LHT11], where the local dependency between nodes is chosen from the underlying reaction network G_{RN}. For analysis purpose, we also consider DBN $MIDBN_{58}$ with the same 58 variables as *RNDBN*, but which differs by the parent relation, defined using our mutual information based procedure. DBN simulations are performed using look-ahead simulations [PPB+16b].

3.6.1 Abstractions produced by DBNizer

Using the approach described in Section 3.5.2, we select variables having maximal effect on cell fate. Setting a cut-off of 0.005 for conditional mutual information, we obtained 6 variables, namely by order of importance *Bcl2c*, *XIAP*, *Flip*, *Bax*, *Mcl1* and *Bid* (Table 3.2). The species that are not considered, have an initial concentration with almost no impact on the cell fate. In addition to these 6 variables, we also added *cPARP* as it is the marker for cell death. This set of variables is used to define and compute a mutual information based DBN, $MIDBN_7$.

To test the robustness of our variable selection scheme, we reiterated the computations for a significantly different amount of TRAIL, namely 10 ng/mL. The key native variables did not vary much: only the least informative variable, *Bid*, is replaced by *Smacm* (Table 3.2).

Following the procedure described in Section 3.5, the complexes *tBid-Bax*, *C8-Bid* and activated-*C3* were iteratively added, resulting in DBNs of increasing size, $MIDBN_8$, $MIDBN_9$, and $MIDBN_{10}$. The procedure stopped at 10 variables since adding any other variable to $MIDBN_{10}$ did not improve significantly the discerning power of the DBN.

The variables considered by $MIDBN_{10}$ are depicted in white in Figure 3.6. The associated network of causalities (parent relation) computed automatically for $MIDBN_{10}$ is represented on Figure 3.6. This network has several interesting features. First, one could be surprised by the fact that the activated initiator Caspase8 ($C8^*$) does not appear. The DBN does not need the concentration of $C8^*$ explicitly as it can be fairly-well evaluated using *Flip* and R^*. The same goes for Bax^*, generally considered a critical player for apoptosis decision, which can be fairly well evaluated using *tBid-Bax*, *Bax*

Samples used to build $MIDBN_{58}$	Discerning power **(HSD: 100%)**
500000	94.14%
100000	94.12%
50000	93.5%

Table 3.1: Effect of different sample sizes on the discerning power of the most demanding DBN 58 for 250ng/ml of TRAIL. All the 3 DBNs gave comparable results. We settled on 100000 samples used to build every MIDBN in the paper because it offers a good compromise between accuracy and DBN construction time. This is easily tunable as an exposed parameter in DBNizer.

TRAIL = species	250 ng/mL MI	TRAIL = species	10 ng/mL MI
Bcl2c	0.33	Bcl2c	0.456
XIAP	0.023	XIAP	0.014
Flip	0.023	Bax	0.008
Bax	0.021	Mcl1	0.01
Mcl1	0.025	Smacm	0.011
Bid	0.020	Flip	0.009

Table 3.2: Conditional mutual information of species with respect to death decision computed in two different conditions. The identified set of most important variables does not vary much in the two conditions.

and *Mcl1*. The level of free *Mcl1* is therefore more informative on the cell fate than the level of Bax^*. One can hypothesize that this comes from the

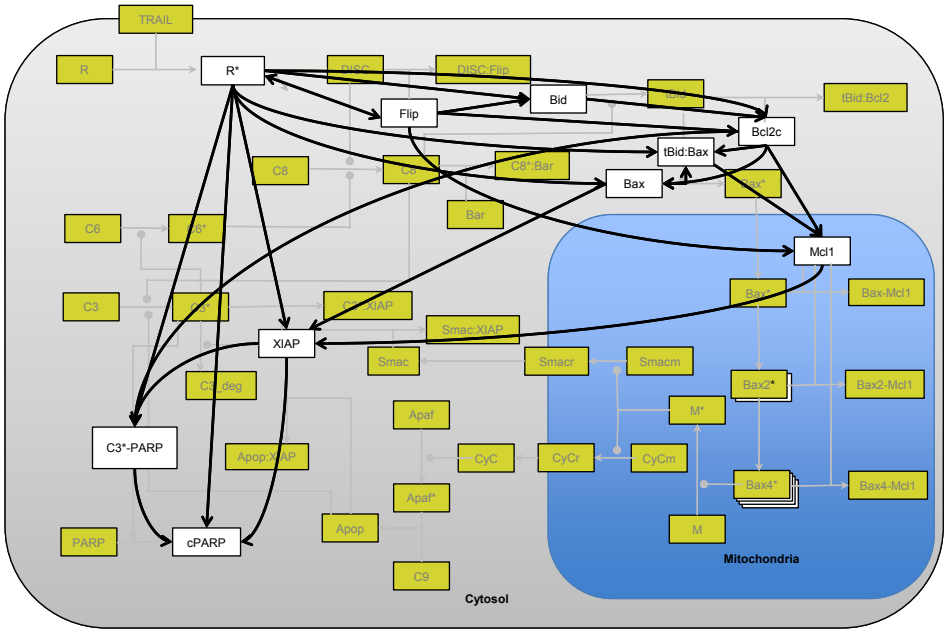


Figure 3.6: Inferred connectivity network for $MIDBN_{10}$ (self connections not represented). The nodes of the DBN are *Flip*, *Bid*, *Bcl2c*, *tBid-Bax*, *Bax*, *Mcl1*, *C3-PARP*, *XIAP*, *cPARP* and activated form of *R*. Selected variables correspond either to the upper part of the pathway or to its lower part, with only one representative of mitochondrial processes (*Mcl1*).

fact that free *Mcl1* is able to efficiently sequester recently-produced active *Bax*, making its level a measure of the cell's resistance to apoptosis induction. More generally, the set of reactions taking place into the mitochondrion is barely represented. The direct inhibitor *XIAP* of the executioner caspase $C3^*$ is in our abstraction directly influenced by *tBid-Bax* and *Mcl1* (in addition to $R*$; Figure 3.6). This strongly suggests that the mitochondrion acts as a black box with a fast (given the timescale of DBNs) and relatively simple input/output function.

3.6.2 Quality and efficiency of the abstractions

In this section, we evaluate the different abstractions produced. Ideally, behaviors predicted using the original or an abstract model should match. However, because the original model is stochastic (and the abstract one too), such a direct comparison is not possible. A first, global measure of quality is given by the comparison of the predicted percentage of cell death. The original HSD model predicts that nearly 70% of cells die. Abstract models should predict similar values (see Table 3.3, second column). Moreover, the timing of death, that is, the distribution of death times, should be similar (see Figure 3.7).

A more refined measure of abstraction quality is provided by the discerning power. The probability that a cell dies depends on its initial state,

Model	cell death (**HSD: 69.9%**)	discerning power (**HSD: 100%**)	Time / 1000 simulations (**HSD: 56s**)
$MIDBN_7$	70.43%	96.14%	**2.13**s (**26.3**X)
$MIDBN_8$	**69.57**%	96.31%	2.64s (21.21X)
$MIDBN_9$	69.33%	96.37%	2.98s (18.8X)
$MIDBN_{10}$	69.03%	**96.84**%	3.30s (17X)
$MIDBN_{58}$	66.85%	94.12%	73.05s
RNDBN	92.29%	85.53%	299s

Table 3.3: Quality and efficiency of the abstractions. All mutual information based DBNs show good results for quality as measured by percentage of cell death and discerning power (see text for their definition), with $MIDBN_{10}$ providing the best results. All compact DBNs show good performance as measured by simulation time with at least a ten times speedup with respect to the reference HSD model.

that is, the initial concentrations of the proteins involved in signal transduction. Indeed, it has been observed that applying TRAIL to two sister cells just after division results in highly correlated fates (dead or alive) of the two cells [SGA+09]. We say that a model has a good discerning power if for many different initial conditions, it is able to predict the death probability obtained with the original model for the same initial conditions (see Fig. 3.8 for an illustration). Note that it is a more stringent criterion than the overall death percentage. In practice, we ran 200000 simulations of the HSD model, and record in each case the fate of the cell (dead or alive) together with its initial configuration defined as the discrete value of the initial concentrations of the 6 selected key proteins whose initial configurations have an impact on the cell

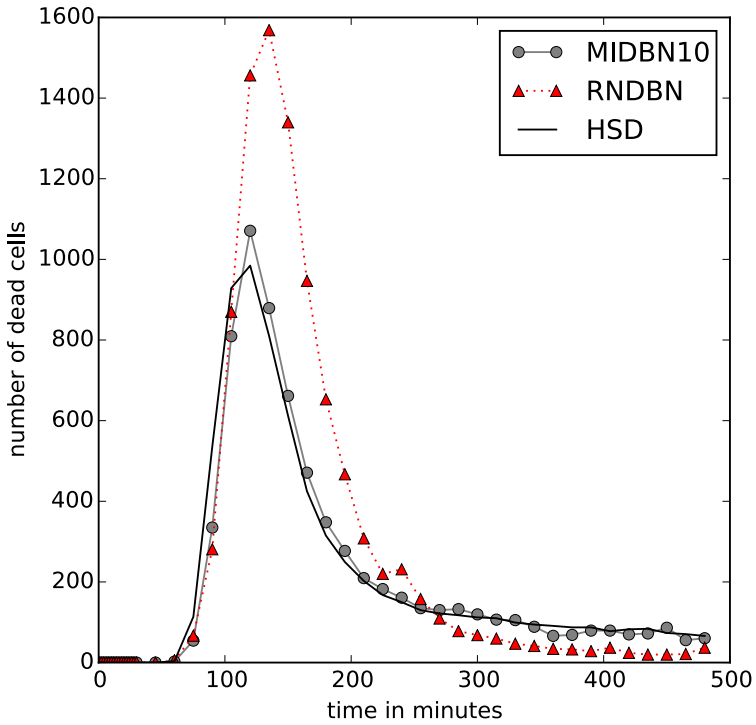


Figure 3.7: Distribution of the time of death during a TRAIL treatment as predicted by the reference HSD model and two abstractions, $RNDBN$ and $MIDBN_{10}$. The death distribution of $MIDBN_{10}$ very closely follows that of the HSD model with only a marginal error at the peak value. $RNDBN$ on the other hand significantly overestimates the number of cell death during the period 100-200 minutes and slightly underestimates it later on.

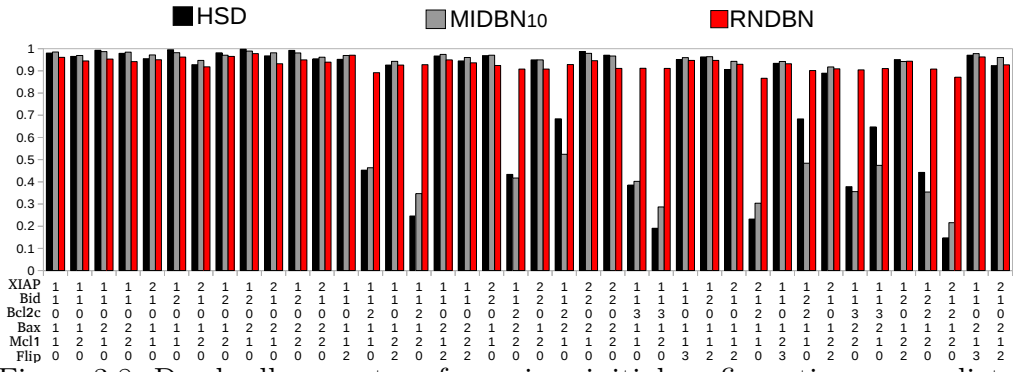


Figure 3.8: Dead cell percentage for various initial configurations as predicted by HSD, $MIDBN_{10}$ and $RNDBN$. As expected, death probability is maximal when pro-apoptotic proteins *Bid* and *Bax* are high and anti-apoptotic proteins, *XIAP*, *Bcl2c*, *Mcl1*, and *Flip*, are low (*e.g.* 100% in initial configuration 120210). The converse holds as well (*e.g.* 15% in initial configuration 212120) $MIDBN_{10}$ accurately follows the dynamics of the original model, in accordance with its good discerning power. In contrast the cell death percentage for *RNDBN* does not vary much over all initial configurations.

fate. For instance, the configuration $XIAP : low$, $Bid : high$, $Bcl2c : verylow$, $Bax : high$, $Mcl1 : low$, $Flip : verylow$ (configuration `120210`) was highly represented (2628 samples) and highly associated to cell death (99% probability), whereas the configuration $XIAP : high$, $Bid : low$, $Bcl2c : veryhigh$, $Bax : low$, $Mcl1 : high$, $Flip : verylow$ (configuration `213120`; 809 samples) leads to cell death in only 9% of the simulations. For each initial configuration, we compute the difference between the predictions by the HSD model and the DBN abstraction weighted by the percentage of occurrence of this profile in the HSD simulations. For statistical reasons, we focused on the 60 most frequent configurations that together represent 50% of the 200000 simulations. Any single such configuration is represented in at least 700 simulations. The discerning power is then defined as 100% minus this weighted error (see Table 3.3, second column).

Regarding the efficiency of the simulation, we assess the time needed to run 1000 simulations of the original HSD model and of its various abstractions, $MIDBN_7$, $MIDBN_8$, $MIDBN_9$, $MIDBN_{10}$, and for reference $MIDBN_{58}$ and $RNDBN$. All these information are provided in Table 3.3 (last column).

As represented in Fig. 3.7, and summarized in Table 3.3 (first column), all DBNs using mutual information provide good to very good descriptions

of the dynamics of cell death. This is in sharp contrast to the reaction network based DBN, *RNDBN*. The comparison of *RNDBN* and $MIDBN_{58}$, having both 58 nodes, clearly shows that the critical feature is to have a proper parent relation. As expected, the comparison of DBNs with 7 to 10 variables (Table 3.3) shows that adding more variables improves accuracy. However, the performance of $MIDBN_{58}$ is slightly worse than that of $MIDBN_7$, indicating that in probabilistic representations there might be a trade off between the capacity to store information (favoring large DBNs) and to reuse it (favoring small ones).

Similar results are found for the discerning power (Table 3.3 (second column)). All MI based DBNs have a $> 94\%$ discerning power, in sharp contrast to *RNDBN* ($< 86\%$). To better analyze the low performance of *RNDBN*, we represent the predicted percentage of cell death in different initial configurations (Figure 3.8). We observe that irrespective of the initial configuration, *RNDBN* predicts a constant high death rate ($> 80\%$): Influences between variables are not well captured by *RNDBN* for this challenging dynamical system (high dimension, strong non-linearities).

The analysis of simulation times (Table 3.3, last column) shows that the performance of abstraction depends strongly on their size. Large DBNs, *RNDBN* and $MIDBN_{58}$, are actually slower to simulate than (an optimized implementation of) the original HSD model. All compact DBNs however show good performance, being at least 17 times faster than the HSD model (for $MIDBN_{10}$) and up to 26 times faster (for $MIDBN_7$). Experiments run for treatment with 10ng/mL TRAIL display very similar results.

In summary, using MI based DBNs is essential to obtain abstractions of good quality; and using low-dimensional DBNs is essential to obtain efficient abstractions. $MIDBN_7$ to $MIDBN_{10}$ present both advantages, with slightly different trade-offs.

3.6.3 Importance of MI based abstractions

In the previous section, we found that the approach used to define local dependencies between variables has a critical impact on the abstraction quality. To better understand why this choice is so important, we focus in this section on how the parent relation is represented in *RNDBN* and in $MIDBN_{58}$ for one particular complex, namely $M^* - Smacm$.

An excerpt of the network is represented in more detail in Figure 3.9. Because of the reaction forming $M^* - Smacm$, the parents of $M^* - Smacm$

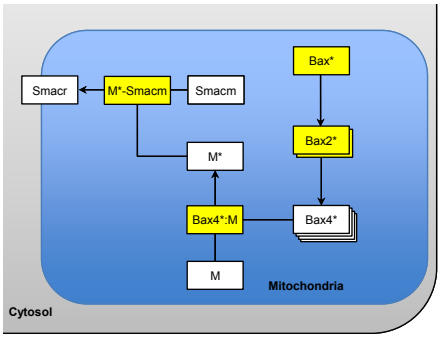


Figure 3.9: Excerpt of the apoptosis reaction network focusing on the complex $M^* - Smacm$ and mitochondrial reactions. For the structure-based $RNDBN$, parents of $M^*{-}Smacm$ are $M^*{-}Smacm$, $Smacm$ and M^*. For the MI-based $MIDBN_{58}$, parents are $M^*{-}Smacm$, Bax^*, $Bax2^*$, and $Bax4^*{-}M$, depicted in yellow.

in $RNDBN$ are M^*, $Smacm$, and $M^*{-}Smacm$. In $MIDBN_{58}$, Bax^*, $Bax2^*$, and $Bax4^*{-}M$ have been chosen as parents by our MI-based approach, in addition to $M^*{-}Smacm$ itself (depicted in white in Figure 3.9).

We conjecture that the speed of reactions from $Bax4^*$ to $M^*{-}Smacm$ is faster than the timestep of the DBN, while Bax^* and $Bax2^*$ have a more gradual evolution. This seems confirmed considering a trajectory of the system, as depicted in Fig 3.10. Because of causality, one would expect that a significant increase of reactants in a reaction would cause a significant increase of the products. After discretization, this should typically lead to two successive threshold crossing events, reactants being followed by products. However, if reactions are fast it may often be the case that the two threshold crossings happen during the same time interval and therefore appear simultaneously after time discretization, thereby loosing causality and mutual dependence between the current values of parents and the future value of the variable to predict. This can be observed in Figure 3.10 where both M^* and $M^*{-}Smacm$ cross their threshold during the interval 150-165 minutes. Defining parent relations based on the reaction network might therefore not be appropriate for systems showing fast and slow dynamics (referred to as "snap-action" in [ABS+08]) as is the case for apoptosis.

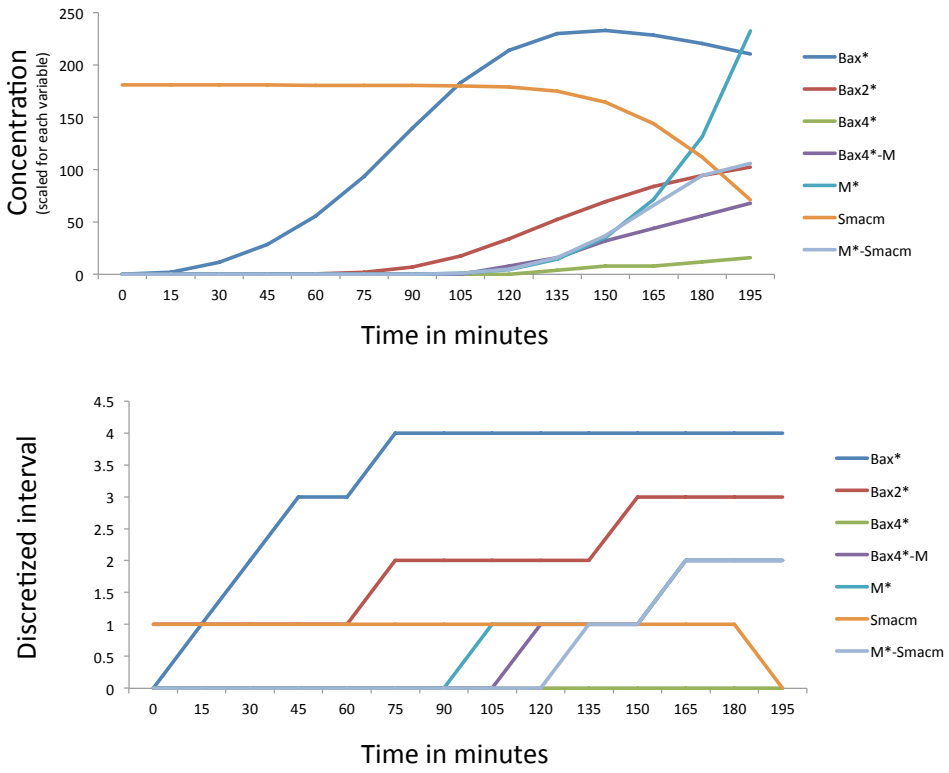


Figure 3.10: Temporal evolution of the concentration (top) and its discretized value (bottom) for selected variables of one simulation of the apoptosis pathway. During the time interval 150-165, $Bax4^* - M$, M^* and $Smacm - M^*$ go simultaneously from level 1 to level 2. Using these fast evolving variables is not as informative as using variables $Bax^*, Bax2^*$ evolving more slowly.

3.7 Conclusion

In this chapter we proposed a strategy to automatically infer from ODEs the structure of small DBNs and demonstrate their accuracy and efficiency. This model allows to accurately represent cell death resulting from the apoptosis pathway compared to the original model DBNs were generated from, the HSD model (the mismatch is of the order of 1%). The model is an order of magnitude faster than the original simulations and is also general and can be used for different set of data given that sufficient simulations are given to train the DBN.

While DBNs describe accurately the values of variables in regard to their parents, they can fail to keep track of correlations that exist between variables

for different reasons than parents values (e.g. mutually exclusive proteins). As a future work, we propose an alternative: a class of models more accurate than DBNs, that would encode probabilities of tuples of variables in CPTs rather than over single variables (e.g. $CPT^t_{i,j}(x, y \mid \vec{U})$).

Chapter 4

Cellular level: Analysing the evolution of the pathway

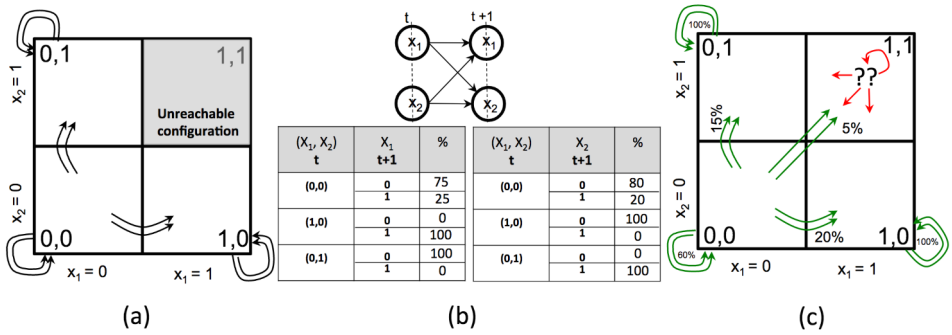
Modeling the evolution of a pathway containing dozens of species over a long period of time can lead to several computation hurdles. The main one is keeping the complexity of the calculations under a reasonable level without losing too much precision. DBNs as shown previously represent a good way to reduce this complexity. Still, analysing the DBN is a complex task. Two ways to do so will be described in this chapter : The first one, by drawing many runs using a simulation algorithm. This is not an easy task because of singularities [PPB+16b], which are configurations of the system that are reached even if they never were present in the original data (and in the CPT) and thus from which the DBN can't determine a following step. The second one, by computing the probability distribution using an (approximated) inference algorithm. Results from this chapter come from [PPFG18, PPFG17] in which I am first author. I also implemented the main algorithm, which is freely available at https://perso.crans.org/~genest/D22.zip.

4.1 Singularities

A problem that can arise with DBNs that wasn't explored in the previous chapter is the risk of singularities.

We will show through a simple example how those singularities appear:

Consider a system with 2 variables X_1 and X_2 each of which takes values from the discrete set $\{0, 1\}$. Figure 4.1 (a) shows the possible configurations



(X_1, X_2) t	X_1 t+1	%
(0,0)	0	75
	1	25
(1,0)	0	0
	1	100
(0,1)	0	100
	1	0

(X_1, X_2) t	X_2 t+1	%
(0,0)	0	80
	1	20
(1,0)	0	100
	1	0
(0,1)	0	0
	1	100

Figure 4.1: Example of singularity. (a) shows the possible configurations of the system. (b) shows the dependencies (top) and the CPTs generated from this system (bottom). (c) shows the new possible configurations derived from the CPTs.

of the system for all value assignments of the variables. For instance, the rectangle corresponding to $(0, 0)$ means that the system is at configuration defined by $X_1 = 0$ and $X_2 = 0$ and so on. Figure 4.1 (a) also shows that the original dynamics of the system forbids it from reaching the configuration $(1, 1)$ i.e., the system cannot reach the configuration with both $X_1 = 1$ and $X_2 = 1$. More precisely, if the system is in configuration $(0, 0)$ at time t, at time $t+1$, it can stay in the same configuration $(0, 0)$, or go to configuration $(0, 1)$ or $(1, 0)$. However, it cannot go to configuration $(1, 1)$.

Fig. 4.1 (b) shows the DBN construction from the system in Fig. 4.1 (a), as outlined in the previous subsection. Variables X_1^{t+1}, X_2^{t+1} are dependent on both X_1^t and X_2^t, as shown on the top part of the figure. The bottom part of Fig. 4.1 (b) shows the conditional probability table for X_1^{t+1} and X_2^{t+1}. For instance, in the table for X_1^{t+1}, the first column describes the parent configuration tuples, i.e, the values for X_1^t and X_2^t. The second column describes the value X_1^{t+1} can take. The third column represents the corresponding probabilities. The first row describes that if $(X_1^t, X_2^t) = (0, 0)$, then X_1^{t+1} taking value 0 (resp. 1) has probability 75% (resp. 25%) of happening. This CPT is stochastic: $\sum_{v\in\{0,1\}} CPT_i^t(v \mid (0, 0)) = \sum_{v\in\{0,1\}} CPT_i^t(v \mid (0, 1)) = \sum_{v\in\{0,1\}} CPT_i^t(v \mid (1, 0)) = 1$. Notice that owing to the system dynamics, there is no value in the CPT corresponding to (X_1^t, X_2^t) taking value $(1, 1)$, which is a source of singularity in the CPT. Two problems arise from singu-

larities, as shown in Fig. 4.1 (c). First, from configuration $(X_1^t, X_2^t) = (0, 0)$, there is a (small but non null) probability to sample 1 for variable X_1^{t+1} and 1 for variable X_2^{t+1}, and thus to end up in configuration $(1, 1)$. This is a problem because configuration $(1, 1)$ does not correspond to a behavior of the original system. An additional problem is that from this singular configuration $(1, 1)$, there is no defined outgoing probability distribution.

This is manageable using different methods:

For simulations by avoiding those singularities through look-ahead algorithms that actively look if generated simulations will create singularities and retry if they do so [PPB+16b]. We will show in the next section one such algorithm.

For inference (see section 4.3.1), avoiding those singularities is not possible anymore as all possible paths are explored. This leads to substochastic distributions, that is for which entrie sums up to less than 1. This must be corrected or else the total probability will shrink over time. A simple solution is to renormalize at every step. The outcome is then similar as the look-ahead algorithm as missing CPTs are ignored with both frameworks.

4.2 Look-ahead simulations

Typically drawing a simulation from the DBN involves starting from an initial value assignment over all variables of the DBN (drawn from the initial distribution). Next, for each variable, independently, we pick its value assignment at the next time point according to the distribution dictated by the CPT entries corresponding to its current value assignment. We continue this procedure for all time points of the DBN. In this simulation procedure, singularities appear if a configuration that was never explored in the simulations that served to generate the CPTs is reached.

We use instead a procedure called *look-ahead simulation* that actively avoid singularities. Its main idea is to sample tuples of values $\vec{v}$ representing the configuration at time t, according to the configuration $\vec{y}$ at time $t-1$ and the corresponding CPTs, but conditioned to the fact that $\vec{v}$ has been seen at time t at least once during the original simulations of the system. Computing exactly this conditional probability is however not possible in general as that would require to recompute on the fly the probability for each of the many possible tuples $\vec{v}$.

Instead, for efficiency reasons, we over-approximate this set of configura-

Algorithm 2: Look-ahead simulation of DBNs

$W[1,...,n] = \{-1,...,-1\}$ ▷ $W[i] = -1$ means that the value of variable i is not yet set **for** $i = 1,...,n$ **do**
  $Pre_i^t(W) = \{\vec{l} \in V^{\hat{\imath}} \mid \sum_{k \in K} CPT_i^t(k \mid \vec{l}) = 1 \wedge \forall m$ with $W[m] \neq -1, \vec{l}[m] = W[m]\}$
  $s \leftarrow 0$ **forall the** $\vec{l} \in Pre_i^t(W)$ **do**
    ▷ Compute the probabilities for value assignments in $Pre_i^t(W)$
    $p \leftarrow 1$ **forall the** $j \in \hat{\imath}$ **do**
      ▷ Compute probabilities of $\vec{l}$
      $p \leftarrow p \cdot CPT_i^t(\vec{l}[j] \mid v_{\hat{\jmath}})$
    $Prob[\vec{l}] \leftarrow p$
    $s \leftarrow s + p$ ▷ Sum of probabilities in $Pre_i^t(W)$
  if $s = 0$ **then**
    Abort ▷ No configurations are consistent with W
  $x \leftarrow rand([0, s])$
  $z \leftarrow 0$ **forall the** $\vec{l} \in Pre_i^t(W)$ **do**
    ▷ Pick a $\vec{l}$ according to $Prob$
    $z \leftarrow z + Prob[\vec{l}]$ **if** $x \geq z$ **then**
      ▷ The current $\vec{l}$ has been chosen for variables in $\hat{\imath}$ **forall the** $j \in \hat{\imath}$ **do**
        $W[j] \leftarrow \vec{l}[j]$
      break
  for $i = 1,...,n$ **do**
    $v_i \leftarrow W[i]$

tions. We fill up iteratively a partial value assignment W remembering the values of variables which have been already set ($W[i] = -1$ if i has not yet been set). The i-th iteration, focused on the variable i, assigns values for parents of i, that is for variables in $\hat{\imath}$. For this, we consider the set $Pre_i^t(W)$ which only retains value assignments $\vec{v}_{\hat{\imath}}$ for $\hat{\imath}$ at time t which have been seen in the simulations and which are consistent with W.

Notice that by (inductive) construction of $v_{\hat{\jmath}}$, $CPT_i^t(\vec{l}[j] \mid v_{\hat{\jmath}})$ is well defined at line 8.

Theorem 1 *In the case where there is no singular configurations, Algorithm 2 generates the same distribution over configurations W than a naive algorithm.*

4.3 Bayesian Inference

Instead of generating simulations, we can use Bayesian Inference to update the probability distribution for each time point knowing the computational probability tables of the DBN.

Because the probability distribution P^t is often too complex to be used directly, we often use other representations, possibly approximated and called belief states B^t. Inference is an iterative process that computes a belief state of the system at time t, given the belief state at $t-1$ and the conditionnal probability tables.

The formula of the belief state in general terms is:

$$B^{t+1}(\vec{x}) = f(B^t, CPT^t)$$

where f is a function which depends upon the exact representation of the belief state. We will see some examples below.

4.3.1 Exact inference

Consider the case where the belief state is the exact distribution: $B^t(\vec{x}) = P(\vec{X}^t = \vec{x})$. In this case, function f is given by the following :

$$B^{t+1}(\vec{x}) = \sum_{\vec{u}} B^t(\vec{u}) \prod_i CPT_i^t(x_i|\vec{u}_i)$$

For each time point, we require the joint probability distribution of the system $B^t(\vec{u}) = P(\vec{X}^t = \vec{u})$ at the previous time point, meaning we need to know each possible joint probability distribution $P(X_1^t = u_1, ..., X_n^t = u_n)$, the system can take. Each variable X_n^t can take v discrete values in $\{1, ..., v\}$. Because there are n variables that can each take v values, there are v^n states for the system at time t. With n large, this quickly becomes intractable.

Algorithm 3: Factored Frontier Algorithm

Input : Initial conditions P^0_{FF}, probability tables $P^t_i(x|u_{\hat{\imath}})$
Input : Parents $\hat{\imath}$ for each i for each time point t
Initialization:: $B^0(y_i, z_j) = P^0_{FF}(y_i, z_j)$
for $t \in [0..T-1]$ **do**
  for $i \in N, x_i \in V$ **do**
    $B^{t+1}(x_i) = \sum_{\vec{u}_{\hat{\imath}} \in V^{\hat{\imath}}} B^t(\vec{u}_{\hat{\imath}}) \times CPT^t_i(x_i|\vec{u}_{\hat{\imath}}))$

4.3.2 Factored frontier

There exist multiples methods to infer a dynamic bayesian network in a tractable manner. The Boyen Koller algorithm [BK98] does an exact update of the system, then marginalize on selected subsets. This approximation reduce the complexity of inference with DBN. A variation of this algorithm called the Factored frontier algorithm [MW13] forgo the exact update for an approximated one. It calculates the new distribution of a variable using the distributions of all the parents without accounting for correlations between those parents. In that case, the belief state is given as $B^t_{FF}(i, x_i)$ which is an approximation of $P(X^t_i = x_i)$. It represents the distribution $B_{FF}(\vec{x}) = \prod_i B^t_{FF}(i, x_i)$. This approximation removes the correlations that exist between the variables. The formula of the belief state in that case is for all $i \in I$:

$$B^{t+1}_{FF}(i, x_i) = \sum_{\vec{u}_{\hat{\imath}} \in V^{\hat{\imath}}} B^t_{FF}(i, \vec{u}_{\hat{\imath}}) \times CPT^t_i(x_i|\vec{u}_{\hat{\imath}}) \quad \text{[MW13]}$$

Algorithm 3 is a rough representation of FF.

4.4 Tree Clustered Inference

We now generalize the *Factored Frontier* algorithm to take into account correlations. We denote by $app(P)$ an approximation of a probability distribution P under a more manageable form. For FF we have $app(P) = P_{FF}$ that uses the fully factored approximation from chapter 2. We now use a more precise approximation, namely the non disjoint clusters approximation, $app(P) = P_{\text{NDC}}$.

4.4.1 A Generic Inference Algorithm

The belief state B^t is computed inductively as follows: First, let $B^0 = app(P^0)$. Then, inductively, we let Q^{t+1} be the exact probability distribution computed by CPT^t from B^t: $Q^{t+1}(\vec{x}_X) = \sum_{\vec{u} \in V^X} B^t(\vec{u}_X) \prod_{i=1}^n CPT_i^t(x_i \mid \vec{u}_{\hat{i}})$. We last define $B^{t+1} = app(Q^{t+1})$.

In general, Q^{t+1} cannot be computed explicitly in a tractable way. We can however show that for $app(B) = B_K$ for a set K of non disjoint clusters associated with a tree T, one can compute B^t. For that, we will use prop. 2 to compute $B^t(\vec{u}_{\hat{i} \cup \hat{j}})$ and prop. 4 to compute $B^{t+1}(\vec{x})$.

Notice that the set of clusters K can vary with time. In practice, we will compute at each time point t, the best approximation using clusters using the Chow-Liu method (see Chap. 2) to select the optimal K^t.

Proposition 4 *Let X_i, X_j be two variables and $\hat{\imath}, \hat{\jmath}$ their parents. Then* $B^{t+1}(X_i = x_i, X_j = x_j) =$

$$\sum_{\vec{u}_{\hat{i} \cup \hat{j}} \in V^{\hat{i} \cup \hat{j}}} B^t(\vec{u}_{\hat{i} \cup \hat{j}}) \times CPT_i^t(x_i | \vec{u}_{\hat{i}}) \times CPT_j^t(x_j | \vec{u}_{\hat{j}})$$

Algorithm 4: Clustered Factored Frontier (CFF)

Input: Trees $G^t = (I, E^t)$, for each time point $t \leq T$
Input: Parents $X_{\hat{\imath}}$ for each variable X_i, $i \in I$
Input: Local transition probabilities $CPT^t(X_i | X_{\hat{\imath}})_{i,t}$
Input: Initial distributions $P^0(X_i, X_j)$ for $\{i, j\} \in E^0$
Init : $B^0(X_i, X_j) = P^0(X_i, X_j)$ for $\{i, j\} \in E^0$
for $t \in [1..T]$ **do**
  for $\{i, j\} \in E^t$ **do**
    compute $B^{t-1}(X_{\hat{\imath} \cup \hat{\jmath}})$ by the message passing algorithm on G^{t-1}
    $B^t(X_i, X_j) = \sum_{x_{\hat{\imath} \cup \hat{\jmath}}} B^{t-1}(x_{\hat{\imath} \cup \hat{\jmath}}) \cdot CPT^t(X_i | x_{\hat{\imath}}) \cdot CPT^t(X_j | x_{\hat{\jmath}})$

Proof For $t \geq 0$, we have:

$$\begin{aligned} B^{t+1}(x_i, x_j) =& Q^{t+1}(x_i, x_j) = \sum_{\vec{x} | \vec{x}_i = x_i, \vec{x}_j = x_j} Q^{t+1}(\vec{x}) \\ =& \sum_{\vec{x} | \vec{x}_i = x_i, \vec{x}_j = x_j} \sum_{\vec{u}} \prod_k B^t(\vec{u}) (CPT_k^t(x_k \mid \vec{u}_{\hat{k}})) \\ =& \sum_{\vec{u}} B^t(\vec{u}) \sum_{\vec{x} | \vec{x}_i = x_i, \vec{x}_j = x_j} \prod_k (CPT_k^t(x_k \mid \vec{u}_{\hat{k}})) \\ =& \sum_{\vec{u}} B^t(\vec{u}) \cdot CPT_i^t(x_i \mid \vec{u}_{\hat{\imath}}) \cdot CPT_j^t(x_j \mid \vec{u}_{\hat{\jmath}}) \end{aligned}$$

The last of the above equalities follows since each of the summands within the expression adds up to 1. We separate $\vec{u}$ into $(\vec{v}, \vec{v}')$ with $\vec{v}$ with variables in $\hat{\imath} \cup \hat{\jmath}$. By applying the definition of marginalization, we obtain:

$$\begin{aligned} B^{t+1}(x_i, x_j) =& \sum_{\vec{v}} CPT_i^t(x_i \mid \vec{v}_{\hat{\imath}}) \cdot CPT_j^t(x_j \mid \vec{v}_{\hat{\jmath}}) \sum_{\vec{v}'} B^t(\vec{v}, \vec{v}') \\ =& \sum_{\vec{v}} CPT_i^t(x_i \mid \vec{v}_{\hat{\imath}}) \cdot CPT_j^t(x_j \mid \vec{v}_{\hat{\jmath}}) B^t(\vec{v}) \end{aligned}$$

□

Denoting p the maximal number of parents $\hat{\imath} \cup \hat{\jmath}$ of a cluster $\{i, j\}$, we get:

Theorem 2 *For app the approximation on the tree of clusters, Algo. 4 inductively computes B^T from B^0 in time $O(T \cdot |I| \cdot |V|^p \cdot (|I| + |V|^2))$.*

Proof The correctness of the proof comes directly from Prop. 4. The complexity follows from the following:

- The t comes from the induction on the time point.
- There are at most $|X|$ clusters as each node as a unique parent on the tree, which gives the last $|X|$.
- Now, for each cluster $\{i, j\}$, one computes at most $|V|^{pa}$ values corresponding to $B^t(X_k = \vec{u}_k)_{k \in \hat{\imath} \cup \hat{\jmath}}$. This takes time $|V|^{pa+1} \cdot |X|$, using Prop. 2.

- Further, Algo. 4 computes for each $x_i \in V$ and $x_j \in V$ a value by summing over $|V|^{pa}$ values, which gives a complexity of $|V|^{pa+2} = |V|^{pa+1} \cdot |V|$.

□

Practical considerations

In practice, the transition probabilities $CPT^t(X_i|\vec{X}_{\hat{\imath}})$ are matrices of dimension $|V| \times |V|^{|\hat{\imath}|}$. They derive from data produced by fine grain models, and thus may exhibit singularities, such as zero rows in the case where the conditioning value $X_{\hat{\imath}} = x_{\hat{\imath}}$ was never observed in the data (see [PPB+16b] for a discussion). In addition, an expression like

$$\tilde{B}^t(X_i) \quad \propto \quad \sum_{x_{\hat{\imath}}} CPT^t(X_i|\vec{x}_{\hat{\imath}}) B^{t-1}(\vec{x}_{\hat{\imath}}) \tag{4.1}$$

involves a large number of small values, and may thus suffer from rounding errors. We thus introduce a renormalization in (4.1) to ensure that it yields a proper probability distribution on variable X_i.

In the same way, the central relation of Algo. 4

$$B^t(X_i, X_j) = \sum_{x_{\hat{\imath}\cup\hat{\jmath}}} B^{t-1}(\vec{x}_{\hat{\imath}\cup\hat{\jmath}}) \cdot CPT^t(X_i|\vec{x}_{\hat{\imath}}) \cdot CPT^t(X_j|\vec{x}_{\hat{\jmath}}) \tag{4.2}$$

may not sum to one and furthermore, when marginalizing out X_j in $B^t(X_i, X_j)$ and X_k in $B^t(X_i, X_k)$, one may not get the same marginal $B^t(X_i)$. We therefore rely on (4.1) (renormalized) to compute the expected marginals $\tilde{B}^t(X_i)$ and $\tilde{B}^t(X_j)$ at time t, and then impose the term $B^t(X_i, X_j)$ derived from (4.2) to satisfy both these marginals. This is performed by the standard Iterative Proportional Fitting Procedure (IPFP). In our experiments, convergence (up to numerical noise) took place in 5 to 6 iterations of IPFP.

4.4.2 An algorithm with reduced complexity

We now refine Algo. 4, that performs approximated inference, in order to lower the exponential factor of $|V|$ from $|\hat{\imath} \cup \hat{\jmath}| + 2$ to $\max(|\hat{\imath}|, |\hat{\jmath}|) + 2$. The improvement can thus be significant.

Indeed the most space and time consuming task is generating the probability table of $B^{t-1}(\vec{u}_{\hat{\imath}\cup\hat{\jmath}})$.It is possible though to use the fact that dependencies follow a tree to factorize $B^{t-1}(\vec{u}_{\hat{\imath}\cup\hat{\jmath}}) \cdot CPT(x_i|\vec{u}_{\hat{\imath}}) \cdot CPT(x_j|\vec{u}_{\hat{\jmath}})$ into $B^{t-1}(\vec{u}_K) \cdot CPT(x_i|K) \cdot CPT(x_j|K)$ for some set K with $|K| \leq min(|\hat{\imath}|, |\hat{\jmath}|)$ and thus reduce the necessary probability table size. The idea can be shown in a simple example: Let's consider three variables A, B and C. A and B form a cluster as well as B and C at time t and $A, B \in \hat{\imath}$ and $B, C \in \hat{\jmath}$ as shown in figure 4.2. We want to compute $B^{t-1}(\vec{u}_{\hat{\imath}\cup\hat{\jmath}}) \cdot CPT(x_i|\vec{u}_{\hat{\imath}}) \cdot CPT(x_j|\vec{u}_{\hat{\jmath}})$.

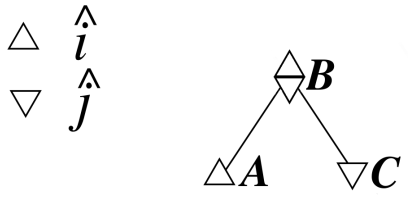


Figure 4.2: A small example showing how CPTs can be reduced. Here with proper marginalization it is possible to reduce them so they depend only on B.

We set $K = \{B\}$ as B "separates" A from C. Let's look at variable A. It is in $\hat{\imath}$ and not in $\hat{\jmath}$, and this variable is in clusters with variable B. It is possible to remove the variable A from $CPT(x_i|\vec{u}_{\hat{\imath}})$ making it dependant on B only. This requires only the distribution on $\{A, B\}$ because $CPT(x_i|\vec{u}_K) = \sum_{u_A} \cdot CPT(x_i|\vec{u}_{\hat{\imath}}) \cdot B(\vec{u}_{\hat{\imath}}|\vec{u}_K)$. We can then repeat this process for $\hat{\jmath}$ and remove C from $CPT(x_j|\vec{u}_{\hat{\jmath}})$ knowing the distribution on $\{B, C\}$. After the marginalization, $B^{t-1}(u_{\hat{\imath}\cup\hat{\jmath}})$ is of size $|V|$ instead of $|V|^3$. And the tables needed for the marginalization were of size $|V|^2$

Notice that we couldn't do the same for B without requiring the whole distribution.

Formally, let us consider a *separating set* $K \subseteq I$ of nodes that separates nodes of $\hat{\imath}$ from those of $\hat{\jmath}$ on tree $G = (I, E)$, *i.e.* any path from $\hat{\imath}$ to $\hat{\jmath}$ on G crosses K (see Fig.4.3). Notice that one has $\hat{\imath} \cap \hat{\jmath} \subseteq K$, and that taking $K = \hat{\imath}$ or $K = \hat{\jmath}$ satisfies this separation property. In the following, we build a separating set K with good algorithmic properties:

First, for $k \in K$ not a leaf of G, we define the *k-section of* G as follows: it is the minimal subtree of G rooted at k, containing at least 2 nodes, and such

that its leaves are either in K or are leaves of G. For instance, on Fig. 4.3, we have $r \in K$, and the r-section has 6 nodes, including r and s. It has three leaves, two being the children of s, plus another leaf in K. For $k, k' \in K$ not leaves of G, the intersection between the k-section and the k'-section have either no node in common, or only one, which is either k or k'. For $k \in K$ not a leaf of G, we define the *strict k-section* of G as the k-section minus the leaves of the k-section which are in K.

The set K is built bottom-up, that is recursively starting from leaves and progressing up towards the root of G. Each node will be tagged by a number in $\{0, 1, 2, 3\}$. Nodes tagged 3 will be nodes of K. We first tag a leaf by:

- 0 if it is not in $\hat{\imath} \cup \hat{\jmath}$,
- 1 if it is in $\hat{\imath} \setminus \hat{\jmath}$,
- 2 if it is in $\hat{\jmath} \setminus \hat{\imath}$,
- 3 if it is in $\hat{\imath} \cap \hat{\jmath}$.

We then inductively tag a node by:

- 3 if it has a child tagged by 1 (or himself is in $\hat{\imath}$) and if it has a child tagged by 2 (or himself is in $\hat{\jmath}$). Else:
- 1 if it is in $\hat{\imath} \setminus \hat{\jmath}$ or as at least one child tagged as 1,

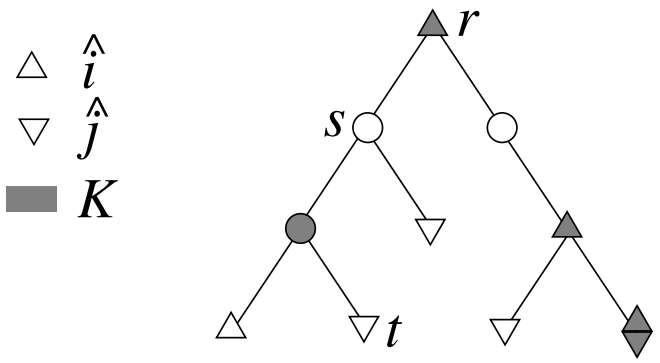


Figure 4.3: A more complex exemple. Here the separating set K between $\hat{\imath}$ and $\hat{\jmath}$ is shown in grey. CPTs can be reduced to depend only on this separating set.

- 2 if it is in $\hat{\jmath} \setminus \hat{\imath}$, or has one child tagged by 2
- by 0 otherwise.

At the end of the procedure, we set K as the set of nodes tagged by 3. That is, a node k where at least one branch of type i and one of type j meet is declared to belong to K. The separating set K in Fig. 4.3 has been built using this algorithm. Node s is tagged 2 as it has one child in $\hat{\jmath}$. Then r is tagged 3 as it is himself in $\hat{\imath}$ and it has has a child (node s) tagged by 2. It is easy to see that K is a separating set, and that for all $k \in K$ not a leaf of G, the strict k-section contains at least one node of $\hat{\imath}$ and one node of $\hat{\jmath}$ (possibly, this is the same node if it is in $\hat{\imath} \cap \hat{\jmath}$, and possibly, this is k itself). In particular, $|K| \leq min(|\hat{\imath}|, |\hat{\jmath}|)$.

We now examine an efficient algorithm that computes $CPT(y_i|\vec{x}_K)$ from $CPT(y_i|\vec{x}_{\hat{\imath}})$. It inductively eliminates nodes in $\hat{\imath} \setminus K$, section by section, in a bottom-up fashion, removing nodes in $\hat{\imath} \setminus K$ and adding nodes from K. We consider the properties of this procedure with respect to $\hat{\imath}$, but the same holds with respect to $\hat{\jmath}$.

Let us define a set of nodes that will evolve in our procedure :

- $A \subseteq K \cup \hat{\imath}$ progressively introduces nodes of K in replacement of nodes of $\hat{\imath}$, initialized to $A = \hat{\imath}$. At the end of the procedure, $A = K$.

Let us start from deepest k-sections in the tree, for $k \in K$. First, if k a leaf, there is nothing to do as $k \in K \cap \hat{\imath}$. The set A stays the same. Otherwise, for all k' below k in the tree, k' has already been considered by induction, that is $k' \in A$. Let C be the subset of nodes in the strict-k-section that are in $\hat{\imath}$. We will explain how to perform an operation amounting to:

$$A \;:=\; (A \setminus C) \cup \{k\}$$

At every step, $|A|$ does not increase, as $|C| \geq 1$ (in the case where $C = \{k\}$, nothing happens). At the begining of the procedure, we have $P(y_i|\vec{x}_{\hat{\imath}})$, that is $P(y_i|\vec{x}_A)$ as A is initialized to $\hat{\imath}$. It is easy to inductively compute $P(y_i|\vec{x}_{(A\setminus C)\cup\{k\}})$ from $P(y_i|\vec{x}_A)$. We have:

$$CPT(y_i|\vec{x}_{(A\setminus C)\cup\{k\}}) \;=\; \sum_{x_{C\setminus\{k\}}} CPT(y_i|\vec{x}_A)\, B(\vec{x}_A|\vec{x}_{(A\setminus C)\cup\{k\}})$$

Algorithm 5: Improved CFF (ICFF)

Input: Trees $G^t = (I, E^t)$, for each time point $t \leq T$
Input: Parents $X_{\hat{\imath}}$ for each variable X_i, $i \in I$
Input: Local transition probabilities $CPT^t(X_i|X_{\hat{\imath}})_{i,t}$
Input: Initial distributions $P^0(X_i, X_j)$ for $\{i, j\} \in E^0$
Init : $B^0(X_i, X_j) = P^0(X_i, X_j)$ for $\{i, j\} \in E^0$
for $t \in [1..T]$ **do**
  for $\{i, j\} \in E^t$ **do**
    for *node k of tree in K in a bottom up fashion* **do**
      for *each child v of k with v labeled 1* **do**
        Inductively search the tree top down from v till nodes of K or leaves are found. Let C be the set of nodes in $\hat{\imath} \setminus \hat{\jmath}$ found, and D be the set of nodes of K found plus k.
        Define $CPT_i^t(y_i \mid \vec{x}_{A \setminus C \cup \{K\}}) =$

$$\sum_{\vec{x}_K} \frac{B^{t-1}(C \cup D)}{B^{t-1}(D)} CPT_i^t(y_i \mid \vec{x}_A)$$

        let $A := A \setminus C \cup \{K\}$
      `/* symetrically for` v `labeled 2. */`
    compute $B^{t-1}(\vec{x}_K)$ by the message passing algorithm on G^{t-1}
    $B^t(y_i, y_j) = \sum_{\vec{x}_K} B^{t-1}(\vec{x}_K) \cdot CPT_i^t(y_i|\vec{x}_K) \cdot CPT_j^t(y_j|\vec{x}_K)$

Let D be the set of nodes in K that are in the k-section, and let $E = A \setminus (C \cup D)$. We have $A \cup \{k\} = C \cup D \cup E$ and $(A \setminus C) \cup \{k\} = D \cup E$. Using (2.3), as E is separated from C, D by D, we obtain:

$$B(\vec{x}_A|\vec{x}_{(A \setminus C) \cup \{k\}}) = \frac{B(\vec{x}_{C \cup D \cup E})}{B(\vec{x}_{D \cup E})} = \frac{B(\vec{x}_{C \cup D})}{B(\vec{x}_D)}$$

It thus suffices to set:

$$CPT(y_i|\vec{x}_{(A \setminus C) \cup \{k\}}) := \sum_{x_{C \setminus \{k\}}} CPT(y_i|\vec{x}_A) \cdot \frac{B(\vec{x}_{C \cup D})}{B(\vec{x}_D)} \tag{4.3}$$

We prove that using this method improves greatly the complexity.

Theorem 3 *Given $B^0 = P^0$, one can compute $B^1, \ldots, B^T$ in time $O(T \cdot |I| \cdot (|I| + |V|) \cdot p \cdot |V|^{p+1})$, where $p = max_{t,\{i,j\} \in E^t} max(|\hat{\imath}|, |\hat{\jmath}|)$.*

Proof We have :

$$B(y_i, y_j) \;=\; \sum_{x_{\hat{\imath}\cup\hat{\jmath}}} B(\vec{x}_{\hat{\imath}\cup\hat{\jmath}})\, CPT(y_i|\vec{x}_{\hat{\imath}})\, CPT(y_j|\vec{x}_{\hat{\jmath}}) \tag{4.4}$$

where distribution $B(\vec{X})$ is a tree distribution on $G = (I, E)$, which is thus defined by the tuple $[\, B(X_u, X_v)\,]_{\{u,v\}\in E}$.

The fact that K separates $\hat{\imath}$ from $\hat{\jmath}$ entails that $\vec{X}_{\hat{\imath}}$ and $\vec{X}_{\hat{\jmath}}$ are conditionally independent given $\vec{X}_K$ for distribution B. Therefore

$$B(\vec{x}_{\hat{\imath}\cup\hat{\jmath}}) \;=\; \sum_{x_K} B(\vec{x}_{\hat{\imath}}|\vec{x}_K) B(\vec{x}_{\hat{\jmath}}|\vec{x}_K) B(\vec{x}_K) \tag{4.5}$$

By plugging this expression into (4.4) one gets

$$B(y_i, y_j) \;=\; \sum_{x_K} CPT(y_i|\vec{x}_K)\, CPT(y_j|\vec{x}_K)\, B(\vec{x}_K) \tag{4.6}$$

where

$$CPT(y_i|\vec{x}_K) \;=\; \sum_{x_{\hat{\imath}}} CPT(y_i|\vec{x}_{\hat{\imath}})\, B(\vec{x}_{\hat{\imath}}|\vec{x}_K) \tag{4.7}$$

$$CPT(y_j|\vec{x}_K) \;=\; \sum_{x_{\hat{\jmath}}} CPT(y_j|\vec{x}_{\hat{\jmath}})\, B(\vec{x}_{\hat{\jmath}}|\vec{x}_K) \tag{4.8}$$

Now, notice that if we inductively run (4.3) from $A_0 = \hat{\imath}$ till $A_{N+1} = K$ we obtain $\prod_{\ell=0}^{N} B(\vec{x}_{A_{\ell+1}}|\vec{x}_{A_\ell}) = B(\vec{x}_{A_N}|\vec{x}_{A_0})$. We now show that $\prod_{\ell=0}^{N} B(\vec{x}_{A_{\ell+1}}|\vec{x}_{A_\ell}) = B(\vec{x}_{A_N}|\vec{x}_{A_0}) = B(\vec{x}_{\hat{\imath}}|\vec{x}_K)$. For that we proceed by induction showing that

Proposition 5

$$B(\vec{x}_{A_{\ell+2}}|\vec{x}_{A_\ell}) = B(\vec{x}_{A_{\ell+2}}|\vec{x}_{A_{\ell+1}}) \cdot B(\vec{x}_{A_{\ell+1}}|\vec{x}_{A_\ell})$$

Proof Consider $A_{\ell+1} = A_\ell \setminus C \cup \{k\}$ and $D = (C \cup \{k\}) \cap K$. Thanks to formula 2.3 on NDC, because D separates C and $A_{\ell+1}$ we have:

$$B(\vec{x}_{A_{\ell+2}\cup A_\ell}) = B(\vec{x}_{A_{\ell+2}\cup A_{\ell+1}}) \cdot \frac{B(\vec{x}_{C\cup D})}{B(\vec{x}_C)}$$

$$B(\vec{x}_{A_{\ell+1}\cup A_\ell}) = B(\vec{x}_{A_{\ell+1}}) \cdot \frac{B(\vec{x}_{C\cup D})}{B(\vec{x}_C)}$$

Thus :

$$B(\vec{x}_{A_{\ell+2}\cup A_\ell}) = B(\vec{x}_{A_{\ell+2}\cup A_{\ell+1}}) \cdot \frac{B(\vec{x}_{A_{\ell+1}\cup A_\ell})}{B(\vec{x}_{A_{\ell+1}})} \tag{4.9}$$

We can compare both side of the statement:

$$B(\vec{x}_{A_{\ell+2}}|\vec{x}_{A_\ell}) = \frac{B(\vec{x}_{A_{\ell+2}\cup A_\ell})}{B(\vec{x}_{A_\ell})}$$

$$B(\vec{x}_{A_{\ell+2}}|\vec{x}_{A_{\ell+1}}) \cdot B(\vec{x}_{A_{\ell+1}}|x_{A_\ell}) = \frac{B(\vec{x}_{A_{\ell+2}\cup A_{\ell+1}})}{B(\vec{x}_{A_{\ell+1}})} \cdot \frac{B(\vec{x}_{A_{\ell+1}\cup A_\ell})}{B(\vec{x}_{A_\ell})}$$

Thus using (4.9):

$$B(\vec{x}_{A_{\ell+2}}|\vec{x}_{A_\ell}) = B(\vec{x}_{A_{\ell+2}}|\vec{x}_{A_{\ell+1}}) \cdot B(\vec{x}_{A_{\ell+1}}|\vec{x}_{A_\ell})$$

□

We can see using that proof that at the end of the two inner for loops of Algo 5, we have indeed computed $CPT(x_i|\vec{x}_K)$ and $CPT(x_j|\vec{x}_K)$.

Let us now analyse the complexity of the algorithm. The recursion step requires first to compute $B(x_{C\cup D})$ and $B(x_D)$. According to Prop. 2, this can be done in time $O(|I| \cdot V^{|C\cup D|})$. We now show that $|C \cup D| \leq |\hat{\imath}| + 1$. Let us partition $\hat{\imath}$ into $J_1 \uplus J_2 \uplus J_3$, with J_1 the set of nodes of $\hat{\imath}$ which are *not* below k, $J_2 = C$ and J_3 the rest, that is nodes below the strict k-section. By construction, each strict k-section contains at least a node of $\hat{\imath}$. It means that $|D \setminus \{k\}| \leq J_3$. Thus $|C \cup D| \leq 1 + |J_2| + |J_3| \leq 1 + |\hat{\imath}|$. Then we need to perform the summation $\sum_{x_{C\setminus\{k\}}} P(y_i|x_B) \cdot \frac{B(x_{C\cup D})}{B(x_D)}$. For that, we need to consider a table with $1 + |A \cup C \cup D| = 1 + |A \cup \{k\}|$ variables. As $|A| \leq |\hat{\imath}|$, it gives a tables with at most $|V|^{|\hat{\imath}|+2}$ entries. Computing the sum is linear in the number of entries. We then need to repeat this process for all $k \in K$, which is less than $|\hat{\imath}|$ times. Hence one can compute $P(y_i \mid x_K)$ for all y_i, x_K in time $O(|\hat{\imath}|(|I| + |V|)|V|^{1+|\hat{\imath}|})$.

To conclude the proof, let us gather all elements. We have T time points and $|I|$ clusters (as they form a tree). For each pair of values (y_i, y_j), one must derive $P(y_i|x_K), P(y_j|x_K)$ and then perform (4.6). The latter has complexity $O(|V|^K)$ with $|K| \leq min(|\hat{\imath}|, |\hat{\jmath}|)$, which is clearly dominated by the

computation of $P(y_i|x_K), P(y_j|x_K)$. This results in a total complexity of $O(T \cdot |I| \cdot (|I| + |V|) \cdot p \cdot |V|^{p+1})$. □

Notice that it is a crude upper bound on the worst case complexity, and the actual complexity will be almost always better than that by a factor $|V|$ to $|V|^2$. If further there is a node in $|\hat{\imath} \cap \hat{\jmath}|$ for all $(i, j) \in E^t$, it suffices to set it as the root of G to obtain an immediate improvement of factor $|V|$. Also, when removing nodes from $\hat{\imath}$, one can remove branches of k-sections which does not contain nodes in $\hat{\imath}$. On the other hand, the complexity in Theorem 2 will always be the actual complexity in all cases. So even if $|\hat{\imath} \cup \hat{\jmath}|$ is close to $\max(\hat{\imath}, \hat{\jmath})$, using this improved algorithm is faster than using the non improved version: we obtained improvement of an order of magnitude using it, and no slowdown.

4.4.3 Error Analysis

We can analyze the error $\Delta^t = |P^t - B^t|$ obtained at time t, w.r.t. the one step error $\epsilon_0 = \max_t |Q^t - B^t|$, when using Algo. 4 (or equivalently Algo. 5 which computes the same quantities but with a better complexity).

Following [BK98, PAL+12], this scheme ensures that, denoting by $\beta \leq 1$ the contraction factor associated with the DBN:

Proposition 6 *$\Delta^t \leq \epsilon_0 \sum_{j=0}^{t} \beta^j$. Further, if $\beta < 1$, we have $\Delta^t \leq \frac{\epsilon_0}{1-\beta}$.*

Proof By definition, we have that after applying the CPT to two distributions P, P', the results $\tilde{P}, \tilde{P}'$ will be at distance at most $|\tilde{P} - \tilde{P}'| \leq \beta |P - P'|$. In particular, we have that $|P^t - Q^{t-1}| \leq \beta |P^{t-1} - B^{t-1}|$.

Now, we shall show that Δ^t can be bounded by $\epsilon_0(\sum_{j=0}^{t} \beta^j)$. By definitions and triangular inequality, we have:

$$\begin{aligned} \Delta^t &= |B^t - P^t| \\ &\leq |B^t - Q^t| + |Q^t - P^t| \\ &\leq \epsilon_0 + \beta \Delta^{t-1} \end{aligned}$$

Then by recursively computing the second factor, we obtain,

$$\begin{aligned} \Delta^t &\leq \epsilon_0 + \beta_t \epsilon_0 + \beta\beta\epsilon_0 + \ldots + (\beta\beta\cdots\beta)\epsilon_0 \\ &\leq \epsilon_0 (\sum_{j=0}^{t} \beta^j) \end{aligned}$$

Further if $\beta < 1$, we have:

$$\Delta^t \leq \epsilon_0(\sum_{j=0}^{t} \beta^j) \leq \epsilon_0(\sum_{j=0}^{\infty} \beta^j) = \frac{\epsilon_0}{1-\beta}$$

□

Bounding the one step error

We can further analyze on the fly the one step error ϵ_0 made at each step. For that, it suffices to consider the result of [CL68] for the Chow Liu approximation: we have that the one step error at step k is $\varepsilon^k = |B^t - Q^t| = \sum_i H^t(x_i) - H^t(\vec{X}) - \sum_{(i,j)\text{clusters}} H^t(x_i, x_j)$, where H^t stands for the entropy (at time t), defined as follows:

$$H^t(\vec{X}) = -\sum_{\vec{x}_X \in V^X} Q^t(\vec{x}_X) \log Q^t(\vec{x}_X)$$

Now, $H(x_i, x_j)$ and $H(x_i)$ are already computed by our algorithm for all i and all clusters (i, j). Computing $H^t(\vec{X})$ exactly is however more complex, as Q^t is a multivariate distribution over tens of variables. Nevertheless, it suffices to under-approximate it in order to over-approximate the one step error ε^k.

Under-approximating the entropy: One easy under-approximation is $H(X) \geq 0$. To improve it, one can compute better values by computing a subset S of tuples for which $Q(x)$ is large, and under-approximate $Q(x)$ for these tuples. This can be done in a way very similar to the computation of spikes in [PAL+12]:

It suffices to use $B^t(x_i, x_j)$ and $B^{t+1}(y_i, y_j)$ for clusters in order to select thousands of tuples $\vec{x}$ at time t and $\vec{y}$ at time $t+1$ with potentially large $B(\vec{x})$ and $Q(\vec{y})$ (ones which have the largest projection on clusters). Let S^t and S^{t+1} be these two sets of tuples. For $\vec{x} \in S^t$, the probability $B^t(\vec{x})$ is computed exactly from values of $B^t(x_i, x_j)$ for the clusters. For $\vec{y} \in S^{t+1}$, we under-approximate $Q^{t+1}(\vec{y}) \geq \sum_{\vec{x} \in S^t} B^t(\vec{x}) \prod_i P_i^t(y_i \mid \vec{x}_i)$.

We then use the following to under-approximate $H^{t+1}(\vec{X})$:

$$H^{t+1}(\vec{X}) \geq -\sum_{\vec{y} \in S^{t+1}} Q^{t+1}(\vec{y}) \log(Q^{t+1}(\vec{y}))$$

4.5 Results

In this section we will detail the result obtained with INFERNO, a program generated in python 2.7 to simulate the evolution of biological pathways using the cluster approximation on DBNs shown in the previous section. It is freely available at https://perso.crans.org/~genest/D22.zip. All experiments were performed on an Intel i7-4980HQ processor (2,8 GHz quad core Haswell with SMT) with 16 GB of memory.

The program is tested using the 3 different sets of data that were shown in chapter 2.

- An enzyme catalysis system: The time scale of the system is 10 minutes which was divided into 100 time points. The parents relations for the DBN are obtained using [LHT11, LZT+11]. Conditional probability tables were populated by drawing 10^5 simulations from the underlying ODE model.

- The extrinsic apoptosis pathway (TRAIL): The time horizon of the model is the first 90 minutes period after injection of TRAIL, which was divided into 22 time points. Again, as before, 10^5 trajectories were generated by simulating the HSD model to fill up the conditional probability tables.

- The EGF/NGF pathway: The time horizon of each model was assumed to be 10 minutes which was divided into 100 time points. The parents relations for the DBN are obtained using [LHT11, LZT+11]. To fill up the conditional probability tables, 10^5 trajectories were generated by simulating the ODE model.

For each of the pathway case study, we consider the the approximated inference algorithm, compared with statistical simulations of the DBNs using the algorithm from [PPB+16b]. Results can be found in Fig. 4.5. We report mean error over marginals normalized to FF (with $FF = 100\%$), as the raw numbers are not meaningful - most marginals being irrelevant and thus diluting the raw error tremendously).

We explain these numbers in more detail in the following subsections.

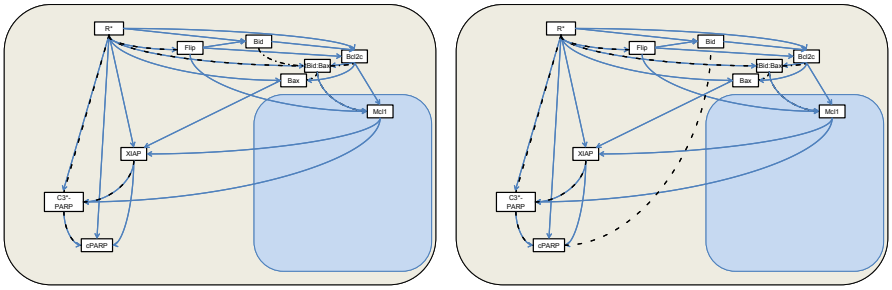


Figure 4.4: Trees built from the abstracted Apoptosis pathway. At 20 minutes on the left, and 90 minutes on the right. The dependencies for CPTs are represented by blue arrows, the edges of the trees are represented by dotted black arrows.

4.5.1 Enzyme Catalysis

As evident in Fig.4.5(a), our method is the most accurate for inference: 20 times less errors overall than Disjoint clusters, and 30 times less maximal error, while being only 20% slower.

4.5.2 Abstracted Apoptosis Pathway

Fig.4.5(b) shows that our algorithm based on Tree Cluster makes minimal error (≤ 0.06). In terms of trade off, it makes half the errors compared with Disjoint Clusters and takes only 1.5 times longer to compute. Compared with FF, it improves accuracy by 7-8 times (FF is very inaccurate on some variable), while being 6.3 times slower. Fig. 4.6 (left) shows the dynamic of the marginal for RAct over time as computed by the different algorithms: Tree Cluster is extremely close to the simulative curve, while Disjoint Cluster is sizably off and FF makes larger errors.

4.5.3 EGF-NGF Pathway

This pathway allows us to compare the inference algorithms with another approximated algorithm, called *HFF (Hybrid FF)* [PAL+12]. In short, HFF keeps a small number of joint probabilities of high value (called spikes), plus an FF representation of the remaining of the distribution. The more

(a) Enzyme catalytic reaction:

Method	Max. Error	Mean Error (normalized)	Nb. Error > 0.1	Comput. Time
FF	0.17	100%	49	0.2s
Disj. Cluster	0.12	65%	16	0.5s
Tree Cluster	0.004	3%	0	0.6s

(b) Apoptosis pathway:

Method	Max. Error	Mean Error (normalized)	Nb. Error > 0.1	Comput. Time
FF	0.44	100%	124	2.2s
Disj. Cluster	0.12	24%	2	9.8s
Tree Cluster	0.06	14%	0	13.8s

(c) EGF-NGF pathway (normalized wrt FF for comparison with HFF):

Method	Max. Error	Mean Error	Nb. Error > 0.1	Comput. Time
FF	100%	100%	100%	1x
HFF (3k)	62%	60%	50%	10x
HFF (32k)	49%	38%	35%	1100x
Disjoint Cluster	84%	79%	84%	1.9x
Tree Cluster	32%	14%	16%	4.2x

Figure 4.5: Table representing the errors of the different inference algorithms.

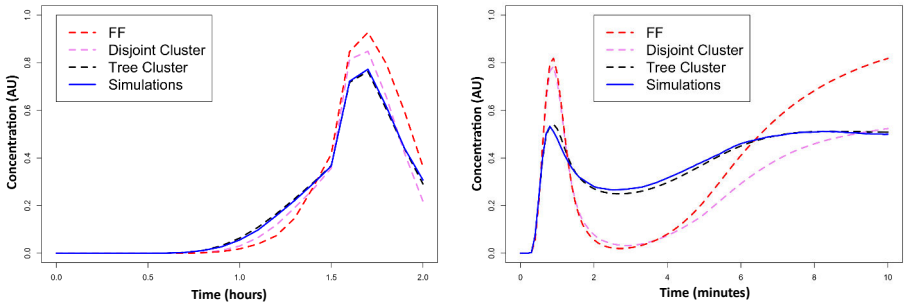


Figure 4.6: Evolution of $P(cPARP = 2)$ in the apoptosis pathway (left) and of $P(ErkAct = 2)$ in the EGF-NGF pathway(right) as computed by the inference based either on FF or Tree cluster approximations (broken lines), compared with the real value (solid line).

spikes, the more accurate and the slower the algorithm. As HFF has been implemented in another language (C++) on a different data structure, we report the error with FF as the baseline in order to draw a fair comparison,

in terms of errors (FF=100%) and time (FF=1x). The superiority of our approach for inference is even more evident in this case. Fig.4.5(b) shows that our method produces 3 times less errors overall than the most accurate method considered before. The maximal errors and number of errors greater than 0.1 are also substantially reduced. FF and disjoint clusters can be 2 to 4 times faster, but with very large errors (see Fig. 4.6 right for an example of large error), while HFF proposes worse results both in terms of time and of accuracy.

4.5.4 Discussion on Inference Algorithms

We compare the evolution of concentrations of each molecule using the different inference algorithms (Fig. 4.5). Overall, performing inference based on the tree clustered representation is fast (less than 15 seconds), while being the most accurate of all the inference algorithms we tested, included HFF with a lot of spikes (32k). To visualize the errors incurred by different approximations, we draw in Fig.4.6 the probability that Erk^* takes a medium concentration in the EGF-NGF pathway and the probability that R^* takes a medium concentration in the apoptosis pathways. The tree cluster approximation follows very closely the simulative curve (in this examples as well as in every examples we considered), while other algorithms are further away, sometimes being far from what is computed by simulations.

4.6 Conclusion

In this chapter, we reviewed several inference algorithms. With different case studies, we show that the algorithm based on non-disjoint clusters of size two forming a tree structure offers a very good trade-off between accuracy and tractability.

In the context of a tissue made of tens of thousands of cells, capturing the inherent variability of the population of cells is crucial. In order to study multi-scale systems in a tractable way, we thus advocate a two-step approach: Firstly, abstract the low level model of the pathway of a single cell into a stochastic discrete abstraction, e.g. using chapter 3.

Secondly, use a model of the tissue, which does not explicitly represent every cell but qualitatively explains how the *population* evolves. In this way, one needs not explicitly represent the concentration of each of the tens of

thousands of cells, but rather only keep one probability distribution. We show how to build such a population model in the next chapter. We will finally use approximate inference, as discussed in this chapter, to keep track of the evolution of the probability distribution.

Chapter 5

Tumour level: A layered subpopulation abstraction

The last part of the thesis tackles the creation of a model that can represent the tumour growth and its response to treatment accounting for the transient resistance that was discussed in the introduction. This model must allow to account for a large amount of cells (up to a million for an unvascularized tumour) and be compatible with the cellular model we have shown in the previous chapter. A poster [PPB+16a] describes the content of this chapter and a paper is a work in progress.

5.1 Different representations

5.1.1 Usual models

There exist multiple mathematical models for tumour growth [BLB+14]. Those models can give great insight on how to use treatment optimally, like for example [SZH+]. Surprisingly, even if growth of tumour is determined by complex factors, it obeys simple laws that can easily be modeled: Tumours usually follow a short exponential growth phase followed by a linerar growth phase [SMC+04]. Many of those models consider only the number of tumoral cells but not their position. There are some, though, that focus on the 3d growth of a spheroid of tumour cells. These models allow to take into account some specific consequences to the positions of the cells like the necrosis of the cells in the middle of the tumour due to the reduced access to nutrients

and oxygen. This also allows to take into account the diffusion of treatment inside the tumour [CRB14]. Tumours of big enough size can become resistant to treatment due to the treatment being unable to reach the center of the tumours [HYM07].

5.1.2 Population model

In our case, we need a way to take into account variability among cells in the tumour as it has implications on their fate (see chapter 2). As said before, considering every single cell cause a combinatorial explosion. A way to represent the cell's fates is to group them in sub populations with the same constraints and external stimuli. This way the stochastic abstraction shown in the previous chapter can be used on the whole subpopulation at once.

The main variables that can influence a cell's fate (its growth and its death) are the following :

- the access to nutrients and growth possibility. Cells in the middle of an unvascularized tumour receive few nutrients and their growth rate is reduced while their chances to necrotize rise.

- the quantity of treatment received. This depends on the permeability of the tumour to treatment and the cells position.

Because both of those are directly linked to the distance to the surface, a layered spheroid representation makes sense. The outer layer keeps the same conditions while inner layer have different conditions depending on their proximity to the outer layer.

5.2 Representation of the system and its evolution

To represent the state of the system, we can consider each layer as an entity containing one single variable : its discretized cell density.

The system evolution is computed using a DBN-like framework. The concentration of a layer at time t depends on the concentration of a subset of layers at time t-1. Natural choices for this subset are the layer of interest

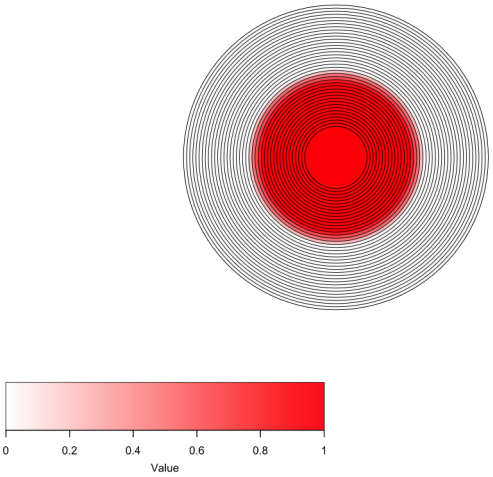


Figure 5.1: Layered representation of a tumour. Layers colored in red are fully filled with cells while white ones are empty.

and the layers directly above and below (the inner layer only considers the layer above).

To obtain data to evaluate the initial conditions and fill the CPTs, we used a modified version of a spacial tumour growth model by Waclaw et al [WBP+15].

5.2.1 Choice of layer sizes and hypothesis

For the DBNs we consider that the new concentration of a layer is dependant of itself and the layers directly above and below at time t-1, except for the first layer that depends on the layer above and itself.

According to the DBN model represented in chapter 3, to model the growth of each layer, we would need CPTs for each layer at each time point. Such a model can only replay what was observed. However we want to generate a predictive model, which is able to predict behaviors that were not observed directly.

A solution for that is to use the same CPTs for all time points and all layers (but the inner layer). This gives the model some predictive power derived from the knowledge of a limited period of time.

If we want to use CPTs that way, we need to be sure to choose time steps

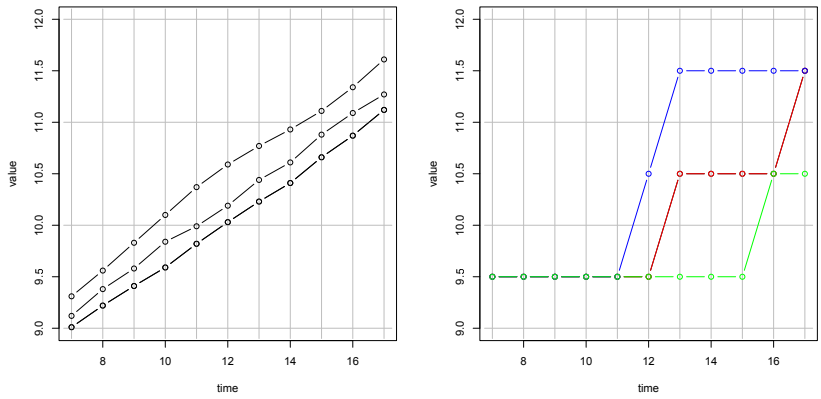


Figure 5.2: Left: A discretization of the data. Right: 3 DBN runs using those discretized values for CPTs

and layer sizes that make CPTs identicals.

A study of the growth model we use shows that the growth is cubic with the number of cells. To obtain layers that are filled at the same rate, we thus need layers to have the same radius.

5.2.2 Time-independent CPTs

Something important to note with DBN-like frameworks is that the time dependence is a way to deal with the information lost with the discretization. Indeed, using the same CPTs for all time points can lead to wide inaccuracies as shown in figure 5.2.

This example represents a possible discretization for a given set of data. Let's consider simple time-independent CPTs generated by this set, that depend only of the specie itself. The CPTs generated tell that if the discretized value is [10-11] at a given time it will have 21% chances to be [11-12] at the next time step and 79% chances to stay the same. We can clearly see that the data suggest that it should take around 5 or 6 time steps for the value to go in the range [11-12] after entering the range [10-11]. However with those CPTs, there is a 18% chance to stay 7 units of time or more in a row in the [10-11] range. With those CPTs, there is also a 21% chance that it takes only two steps to go from the range [9-10] to [11-12]. Those problems stem from the fact that the DBN keeps no track on the position of a value inside

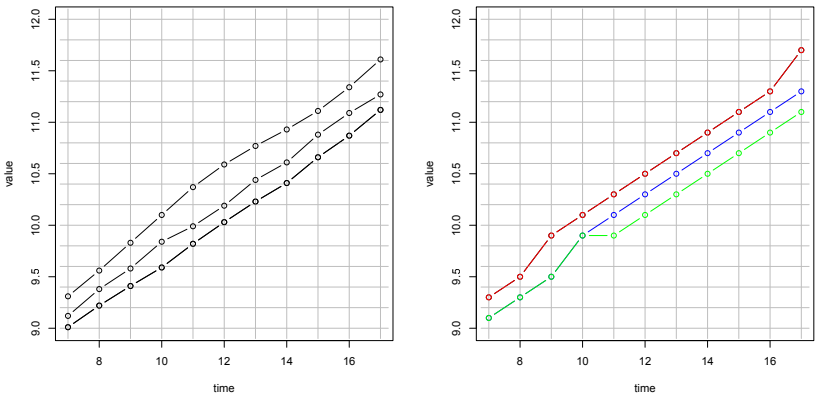


Figure 5.3: Left: A more refined discretization of the data. Right: 3 DBN runs using those discretized values for CPTs

a range. The larger the range, the more inaccurate it will be. Usually for complexity reasons, the number of discrete values can't be too high (often around 5).

In order to handle a unique CPT for multiple time points, we require a precise discretization: More discrete states make all tangible variations in the simulation generate CPTs that possess those variations. The more discrete states, the more precisely those variations are collected. Figure 5.3 shows the results of multiplying by 5 the number of discretized states on the same data as the previous example. With this second example it's impossible to go from the value 9.5 to 11.5 in 2 steps. It's possible (but unlikely) to be stuck in the same state multiple times in a row. And the more discrete states, the smaller a variation can be and still induce a garanteed change of state.

On the other hand this bigger CPT matrix also becomes more impractical to store and to use and more discrete states require more simulations to fill the matrix of the CPTs.

5.2.3 CPT choice for the model

To balance the necessity of fine tuned discrete states and a reasonable size for CPTs, we use different precisions in the CPTs. The number of discretized states is fixed to 101 but for CPT we ignore the units for the layers aboves and below and round them to the nearest multiple of ten (reducing the number of

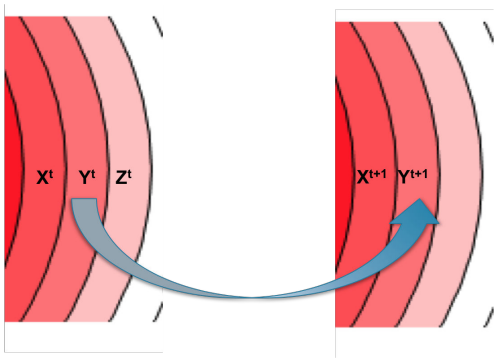


Figure 5.4: The CPT for Y^{t+1} are of the form: $CPT(y^{t+1}|x_1^t, y_1^t, y_2^t, z_1^t, x_1^{t+1})$ with x, y, z possibles values of X, Y and Z respectively with $x_1 = x \div 10$ and $x_2 = x\%10$ and same for y and z.

possible values to 11). This reduce the size of the CPT matrix from roughly 10^8 to 10^6. The first layer depend only on the layer above and itself, its matrix size is thus around 10^5. In those 10^5 cases, a lot are impossible. An extreme example would be a full layer while the layers below are empty. Instead of the explicit representation, we use a sparse representation to only represent the possible cases and further reduce the size of the CPTs.

Our first expermentation with this model shows that adding an extra dependency to the layer below at the same time step helps keeping the layers consistant (less singularities, see Chap. 4). Like the layer below at the previous time point, the precision is kept to the tens. No modification is done to the calculation of the first layer. This new model forces to compute the CPTs from the inner layer to the outer one due to those extra dependencies. The CPTs are shown in figure 5.4.

5.2.4 Data generation for the model

We use a modified version of the TumourSim algorithm [WBP+15]. In this modified version, because the genetic influence part of the algorithm is not our concern here, we set the genetic evolution of the population to 0. Instead, to create data for a spheroid representation, we generate the barycenter of the tumour at various times of execution, then count the cells inside each assigned layer by looking at their distance of the barycenter. The inner layer is a sphere of large radius due to the fact that data are only collected when

the spheroid is large enough so the barycenter stay stable while the tumour grows. All the other layers are concentric spherical shells of the same radius. Their inner sphere radius being the outer layer of the previous layer. A cell is considered inside a layer if its center falls inside this layer. The number of cells inside each layer is transformed in a density by dividing by the volume of the layer. Those densities are then discretized into a range of values between 0 and 100. The discretized values are chosen using the maximum entropy discretization shown in chapter 2.

By generating 10000 runs of the desired evolution (like growth phase alone or growth phase followed by a treatment phase) we can generate the CPTs. We consider only 2 different variables : the inner layer and the other layers for CPT generations. Considering all layers identical allow to fill the CPTs with a reduced quantity of data.

5.2.5 DBN-like model

We present now a model that simulates the evolution of a tumour using the DBN generated using the method previously presented. This model is an unpublished work in progress. It currently allows to simulate DBNs using different CPTs for sets of time points. This allows to simulate changes in the conditions of tumour like alterning growth phases where the tumour isn't under stress and treatment phases where the tumour is actively attacked by a treatment.

5.2.6 Singularities

During the simulations, because of the high number of discretized states, singularities may appear (see Chap. 4). When such a singularity occurs, instead of discarding the run, the program simply removes the last two time points done and restart with the values of the last remaining time point. This operation can be done at most 25 times for a single run, after that the run is discarded. This avoids run beeing stuck on dead ends. While this method is not foolproof, it greatly reduces the number of runs lost due to singularities.

5.3 Model and results

We test the algorithm on multiple sets of data generated from the modified version of TumourSim which is the algorithm presented in [WBP+15]. This agent based algorithm model the growth of a 3D tumour from a single cell and can include a treatment phase. The main modifications done to the algorithm is adding a calculation of the barycenter of the tumour as well as calculate all the positions of the cells relative to the barycenter. A few other minor modifications are made to allow successions of growth and treatment phases.

The main data sets used are the following :

- A growth phase model.
- A death phase model on surface (time to treat = 250, end treatment = 350).
- A diffuse death phase model (time to treat = 200, end treatment = 300).

Values for the TumourSim parameters are as follow for all 3:

- Growth rate: 1
- Death rate: 0.8
- Treatment growth rate: 0.2
- Treatment death rate: 0.95
- Normal model with no migration and no genetic influence.

Time points are selected as following : $T_n = \text{time } 100+2*n$ in the original simulation. For all 3 models, the 200 first time points are used.

5.3.1 Growth model

The first model is a simple growth model. The DBN is trained on a limited number of time points (in our experiment: 50) and the inital conditions are the distribution of the initial conditions of our training data.

Our goal is twofold :

- Have DBN simulations that follow the original simulations with good precision.

- Be able to extend those simulations indefinitively

To test the extrapolation of the DBN, we generate data for 100 time points and use the first 50 time points for training while using the 50 others for comparision. The results of the experiments are shown in figure 5.6.

5.3.2 Death models

Taking death due to treatment into account in our model requires to switch between two different CPTs for the DBNs : One for the growth phase and one for the death phase.

Our training data includes a growth phase followed by a treatment phase where cells die massively. Two models are tested : A surface death, where treatment can't penetrate the inner layers of the tumours and a treatment on a permeable tumour, where all cells are affected. Both versions are integrated in the TumourSim model.

For our first experiments, we trigger the switch to "death mode" at the same time point as the original model. Intuitively, the "initial conditions" after the switch will match approximately the conditions in the original model then the death phase start. Thus the CPTs of the death phase for those conditions should exist, avoiding singularities.

To be sure our model can handle different times of treatments with the same data, we make two sets of data: A learning set with a given treatment time. And a comparision set with an other treatment time. Our CPTs are generated using the first set, but we make the switch at the time of the second set and compare it to it.

5.3.3 Simulation of cycles

A major concern when simulating multiple cycles of growth and death is that the transition between two different CPT structures can easily lead to singularities in the DBN. For the permeable tumour treatment for instance, death affect all layers equally, making a porous structure where for example all layers can be half-filled. The growth phase on the other side normally have all layers filled, one after an other. Never in a normal growth phase, three consecutive layers are half filled at the same time. This means that no data will exist in the CPTs to restart the growth from there.

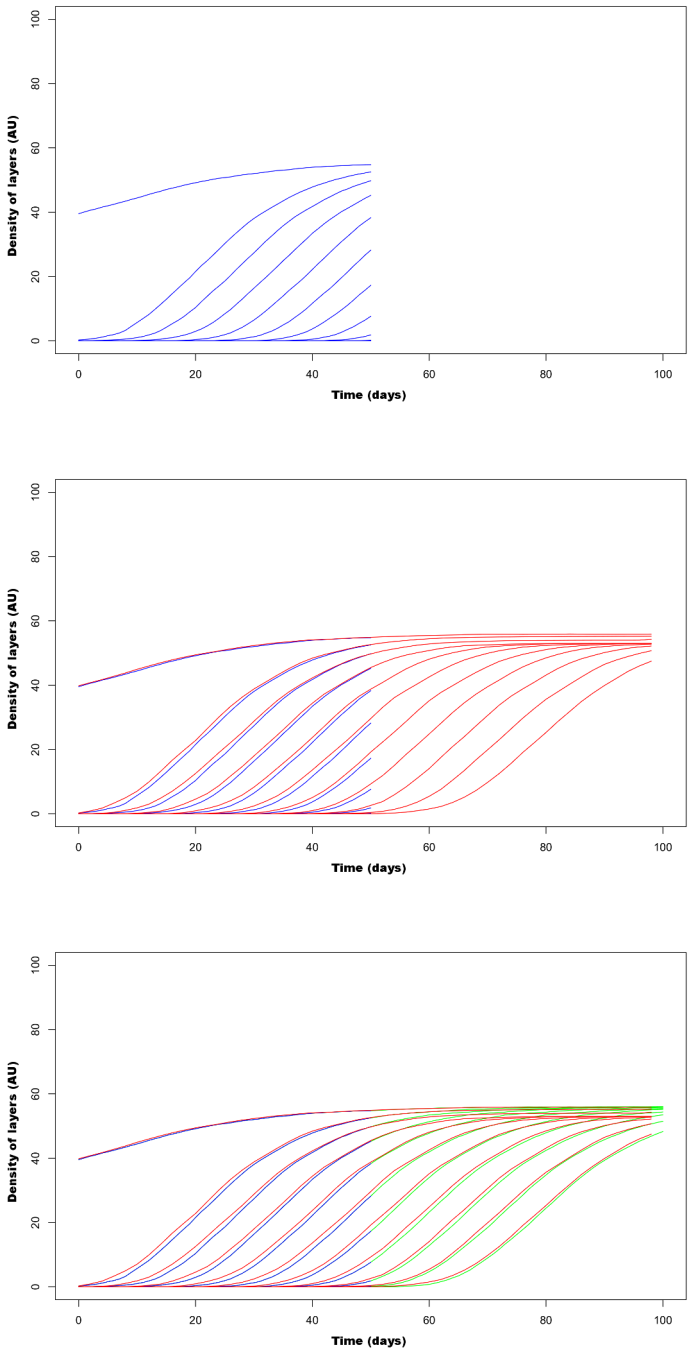


Figure 5.5: Mean densities of multiple layers. In blue: Training data from the original model. In red: Result from the DBN. In green : Extended data from the original model

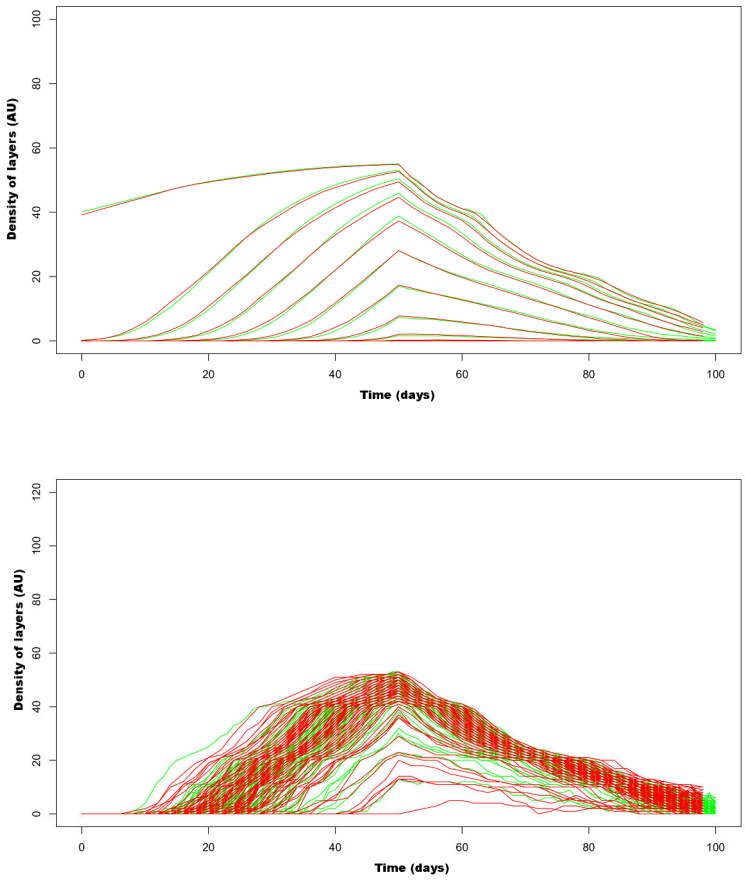


Figure 5.6: Death on surface model. The first figure represent the mean of 1000 experiments on all layers. The second is a representation of 100 experiments for the third layer

For death on surface, the state of the system at the end of the death phase is closer to possible states after a growth phase, allowing in most cases the growth to restart but it's still possible to reach states in the death phase that never obtained in the growth phase.

A solution to that problem is to use the fact that our CPTs are time independent and layer independent and force the generation of growth data after death phases.

5.4 Conclusion

This model describing a tumour growth is still a work in progress and several hurdles have yet to be solved. Mainly, the incorrect transitions between successive growth and death phases due to lack of corresponding CPTs requires an efficient and fast CPT completion method we didn't develop yet.

Chapter 6

Outlook

6.1 Summary and conclusion

Throughout chapter 2 to 4 we have described a way to model the evolution of a poulation of cells under a pathway as a distribution. The resulting model is a promising candidate for a multi-level model of a tumour reaction to treatment. This model has many advantages : It can keep track accurately of multiple dependant proteins concentrations using a reasonable amount of computationnal ressourses (both space and time). The representation as a distribution allows to draw conclusion for entire groups of cells sharing the same conditions. The inference model allows to infer the new state for the whole group at once instead of requiring multiple simulations. Also time is quite flexible in the model. Generating CPTs with different time steps can allow to precisely model critical moments with short time steps (like immediately after TRAIL injection) and use longer time steps for periods during which the system evolves slowly, reducing the computationnal strain on the model.

In chapter 5 we started a proposition for a model that allows to represent the evolution of a tumour as a spheroid with reduced computationnal strain compared to usual agent based models.

6.2 Perspectives

The next logical step in this work is to combine the two abstractions to describe the growth of a tumour and its reaction to successive treatments

acknowledging transient resistances. This would be done by running the layer model for growth, while running in parallel the cellular model for each subpopulation to account for both death due to treatment and acquired resistance. The proposed idea is to consider three different steps :

- Step 1: TRAIL injection. (step skipped when no injection) Look at model of tumour to compute the TRAIL level received by each layer of the tumour depending on its permeability and the quantity injected.
- Step 2: Update the distribution of each layer w.r.t TRAIL level. Run the evolution of the apoptosis pathway during a few time points, update to the new distribution and extract the proportion of cells killed during that time.
- Step 3: Compute the growth of the tumour using the number of deaths from step 2 as death rates for the respective layers (during one time step equivalent in duration to the sum of time steps in step 2).

The steps are repeated multiple times to obtain the full simulation for the desired period of time.

This multi-level model would pose multiple challenges: The growth time scale is weeks while the apoptosis pathway's time scale is hours to day. Synchronysing both can thus be challenging. Layers aren't immuable objects, new ones can appear as the tumour grows and cells can move from a layer to an other. Taking that into account may require exchange of information between the layers, which isn't done by the current model. Also, while the pathway abstraction is a good indicator of different proteins concentrations in most cases, it currently fails to correctly model the evolution of proteins when partial death is involved. Causes are unknown but it seems to be result in an overepresentation of states that lead to death. Tackling all these challenges will be the aim of future work.

Author's publications

[PBP+15] Palaniappan S, Bertaux F, Pichené M, Fabre E, Batt G, Genest B. (2015). Approximating the dynamics of the Hybrid Stochastic-Deterministic Apoptosis pathway. Poster at CMSB.

[PBP+17] Palaniappan S, Bertaux F, Pichené M, Fabre E, Batt G, Genest B. (2017). Discrete Stochastic Abstraction of Biological Pathway Dynamics: A case study of the Apoptosis Pathway. *Bioinformatics* 33 (13): 1980-1986, Oxford University Press.

[PPB+16a] Pichené M, Palaniappan S, Batt G, Fabre E, Genest B. (2016). Efficient Analysis of Multi-level Biological Systems Using Abstraction. Poster at HSB.

[PPB+16b] Palaniappan S, Pichené M, Batt G, Fabre E, Genest B. A (2016). Look-Ahead Simulation Algorithm for DBN Models of Biochemical Pathways *Hybrid Systems Biology, 5th International Workshop, HSB, Oct 2016, Grenoble, France* 9957, pp.3-15, 2016, Lecture Notes in Computer Science.

[PPFG17] Pichené M, Palaniappan S, Fabre E, Genest B. (2017). Non-Disjoint Clustered Representation for Distributions over a Population of Cells. Poster at CMSB, p 324-326, LNBI 10545.

[PPFG18] Pichené M, Palaniappan S, Fabre E, Genest B. (2018). Modeling Variability in Populations of Cells using Approximated Multivariate Distributions. Submitted. Available at https://perso.crans.org/~genest/PPFG18.pdf

Bibliography

[ABS+08] Albeck JG, Burke JM, Spencer SL, Lauffenburger DA and Sorger PK. (2008) Modeling a snap-action, variable-delay switch controlling extrinsic cell death, *PLoS Biology*, vol. 6, 12, pp. 2831–2852,

[Ber16] Bertaux F. (2016). Cell-based multi-scale modeling for systems and synthetic biology : from stochastic gene expression in single cells to spatially organized cell populations. Modeling and Simulation. Université Pierre et Marie Curie - Paris VI. English. NNT : 2016PA066101. tel-01405430v2

[BHC+04] Brown KS, Hill CC, Calero GA, Lee KH, Sethna JP and Cerione RA. (2004). The statistical mechanics of complex signaling networks: nerve growth factor signaling. *Physical Biology 1*, pages 184–195.

[BK98] Boyen X and Koller D. (1998). Tractable inference for complex stochastic processes. In *UAI-98*, pages 33-42.

[BLB+14] Benzekry S, Lamont C, Beheshti A, et al. (2014). Classical Mathematical Models for Description and Prediction of Experimental Tumour Growth. Mac Gabhann F, ed. *PLoS Computational Biology.*;10(8):e1003800. doi:10.1371/journal.pcbi.1003800.

[BSDB14] Bertaux F, Stoma S, Drasdo D, and Batt G. (2014). Modeling Dynamics of Cell-to-Cell Variability in TRAI-Induced Apoptosis Explains Fractional Killing and Predicts Reversible Resistance. *PLoS Comput Biol* ,10(10), 14

[CAB] Chaitanya GV, Alexander JS, Babu PP. (2010). PARP-1 cleavage fragments: signatures of cell-death proteases in neurodegeneration. Cell Communication and Signaling: CCS. 2010;8:31. doi:10.1186/1478-811X-8-31.

[CEJ+97] Chaudhary PM, Eby M, Jasmin A, et al. (1997). Death receptor 5, a new member of the TNFR family, and DR4 induce FADD-dependent apoptosis and activate the NF-kappaB pathway. *Immunity.* ;7(6):821-830.

[CL68] Chow CK, Liu CN. (1968). Approximating discrete probability distributions with dependence tree In *IEEE ToIT* 14 (3) : 462-467.

[CRB14] Celli JP, Rizvi I, Blanden AR, Massodi I, Glidden MD, Pogue BW and Hasan T (2014). An imaging-based platform for high-content, quantitative evaluation of therapeutic response in 3D tumour models. *Scientific reports* 4, 3751, doi: 10.1038/srep03751.

[Csi75] I. Csiszar. (1975). I-divergence geometry of probability distributions and minimization problems. In *The Annals of Probability,* Vol. 3(1): 146–158.

[CT06] Cover TM and Thomas JA. (2006), *Elements of information theory*, 2nd Edition. Wiley-Interscience. ISBN 0-471-24195-4

[DNM+97] Donzelli M, Negri C, Mandarino A, Rossi L, Prosperi E, Frouin I, Bernardi R, Bürkle A, and Scovassi AI. (1997). Poly(ADP-ribose) synthesis: A useful parameter to identify apoptotic cells. *Histochem.* J. 29, 831-837.

[FH73] Folkman J, Hochberg M. (1973). Self-regulation of growth in three dimensions. *The Journal of experimental medicine* 138, 745-753.

[FRSS13] Flusberg DA, Roux J, Spencer SL, Sorger PK. (2013). Cells surviving fractional killing by TRAIL exhibit transient but sustainable resistance and inflammatory phenotypes. *Mol Biol Cell.*;24:2186-2200. doi: 10.1091/mbc.E12-10-0737.

[HW00] Douglas Hanahan and Robert A. Weinberg. (2000). The hallmarks of cancer , *Cell*, Volume 100, Issue 1, 7 January 2000, Pages 57-70

[HYM07] Kyle AH, Huxham LA, Yeoman DM and Minchinton AI. (2007). Limited tissue penetration of taxanes: a mechanism for resistance in solid tumours. *Clinical cancer research: an official journal of the American Association for Cancer Research* 13, 2804-2810, doi: 10.1158/1078-0432.CCR-06-1941.

[KL06] Klipp E, Liebermeister W. (2006). Mathematical modeling of intracellular signaling pathways. *BMC Neuroscience.* ;7(Suppl 1):S10. doi:10.1186/1471-2202-7-S1-S10.

[LC56] Levi-Montalcini R and Cohen S. (1956) In vitro and in vivo effects of a nerve growth-stimulating agent isolated from snake venom. *Proc. Natl. Acad. Sci. U.S.A.* 42 695-699

[LHT11] Liu B, Hsu D, and Thiagarajan PS. (2011) Probabilistic approximations of ODEs based bio-pathway dynamics. *Theoretical Computer Science* , 412:2188-2206 may 2011.

[Llo82] Lloyd SP. (1982). Least squares quantization in PCM. *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 129-137, Mar.

[LZT+11] Liu B, Zhang J, Tan PY, Hsu D ,Blom AM, Leong B, Sethi S, Ho B, Ding JL, and Thiagarajan P. (2011). A computational and experimental study of the regulatory mechanisms of the complement system. *PLoS Comput Biol*, 7(1), e1001059.

[Mal91] Malvestuto F. (1991). Approximating Discrete Probability Distributions with Decomposable Models. In *Trans. on Systems, Man and Cybernetics,* Vol. 21(5): 1287–1294.

[MF91] Marks DI, and Fox RM. (1991). DNA damage, poly(ADP- ribosyl)ation and apoptotic cell death as a potential common pathway of cytotoxic drug action. *Biochem. Pharmacol.* 42, 1859-1867.

[Max60] Max J. (1960). Quantizing for minimum distortion. *IEEE T.Inf.Theory*, vol. 6, 1 pp. 7–12.

[MLM+07] Mérino D, Lalaoui N, Morizot A, Solary E, Micheau O. (2007). TRAIL in cancer therapy: present and future challenges. *Expert Opinion on Therapeutic Targets.* ;11(10):1299-1314. doi:10.1517/14728222.11.10.1299.

[MW13] Murphy K, and Weiss Y. (2013). arXiv:1301.2296

[NDB+97] Negri C, Donzelli M, Bernardi R, Rossi L, Bürkle A, and Scovassi AI. (1997). Multiparametric staining to identify apoptotic human cells. *Exp. Cell Res.* 234, 174-177.

[PAL+12] Palaniappan SK, Akshay S, Liu B, Genest B, Thiagarajan PS. (2012). A Hybrid Factored Frontier Algorithm for Dynamic Bayesian Networks with a Biopathways Application. In *TCBB 9(5)*:1352-1365, IEEE/ACM.

[PBP+15] Palaniappan S, Bertaux F, Pichené M, Fabre E, Batt G, Genest B. (2015). Approximating the dynamics of the Hybrid Stochastic-Deterministic Apoptosis pathway. Poster at CMSB.

[PBP+17] Palaniappan S, Bertaux F, Pichené M, Fabre E, Batt G, Genest B. (2017). Discrete Stochastic Abstraction of Biological Pathway Dynamics: A case study of the Apoptosis Pathway. *Bioinformatics* 33 (13): 1980-1986, Oxford University Press.

[POC+97] Pan G, O'Rourke K, Chinnaiyan AM, et al. (1997). The receptor for the cytotoxic ligand TRAIL. *Science.* ;276(5309):111-113.

[PPB+16a] Pichené M, Palaniappan S, Batt G, Fabre E, Genest B. (2016). Efficient Analysis of Multi-level Biological Systems Using Abstraction. Poster at HSB.

[PPB+16b] Palaniappan S, Pichené M, Batt G, Fabre E, Genest B. A (2016). Look-Ahead Simulation Algorithm for DBN Models of Biochemical Pathways *Hybrid Systems Biology, 5th International Workshop, HSB, Oct 2016, Grenoble, France* 9957, pp.3-15, 2016, Lecture Notes in Computer Science.

[PPFG17] Pichené M, Palaniappan S, Fabre E, Genest B. (2017). Non-Disjoint Clustered Representation for Distributions over a Population of Cells. Poster at CMSB, p 324-326, LNBI 10545.

[PPFG18] Pichené M, Palaniappan S, Fabre E, Genest B. (2018). Modeling Variability in Populations of Cells using Approximated Multivariate Distributions. Submitted. Available at https://perso.crans.org/~genest/PPFG18.pdf

[RHH+09] Rehm M, Huber HJ, Hellwig CT, Anguissola S, Dussmann H, et al. (2009). Dynamics of outer mitochondrial membrane permeabilization during apoptosis. *Cell Death and Differentiation* 16: 613-623.

[SGA+09] Spencer SL, Gaudet S, Albeck JG, Burke JM, Sorger PK (2009). Non-genetic origins of cell-to-cell variability in TRAIL-induced apoptosis. *Nature* 459: 428-432.

[SMC+04] Monica Simeoni, Paolo Magni, Cristiano Cammia, Giuseppe De Nicolao, Valter Croci, Enrico Pesenti, Massimiliano Germani, Italo Poggesi and Maurizio Rocchetti. (2004). Predictive Pharmacokinetic-Pharmacodynamic Modeling of Tumor Growth Kinetics in Xenograft Models after Administration of Anticancer Agents. DOI: 10.1158/0008-5472.CAN-03-2524 Published February 2004

[Sri01] Srivastava RK. (2001). TRAIL/Apo-2L: Mechanisms and Clinical Applications in Cancer *Neoplasia*, Vol. 3, No. 6, pp. 535 - 546.

[SW49] Shannon CE, Weaver W. (1949). The Mathematical Theory of Communication, *Univ of Illinois Press*. ISBN 0-252-72548-4

[SZH+] Stein S, Zhao R, Haeno H, Vivanco I, Michor F (2018) Mathematical modeling identifies optimum lapatinib dosing schedules for the treatment of glioblastoma patients. *PLoS Computational Biology* 14, e1005924.

[TMC72] Taylor JM, Mitchell WM, and Cohen S. (1972) Epidermal growth factor, physical and chemical properties. *J. Biol. Chem.* 247 5928-5934

[WBP+15] Waclaw B, Bozic I, Pittman ME, Hruban RH, Vogelstein B, Nowak MA. (2015). A spatial model predicts that dispersal and cell turnover limit intratumour heterogeneity, *Nature 525* , no. 7568 ,September 10, 2015,: 261-64. doi:10.1038

[Won11] Wong RS. Apoptosis in cancer: from pathogenesis to treatment. (2011). *Journal of Experimental & Clinical Cancer Research*: CR. 2011;30(1):87. doi:10.1186/1756-9966-30-87.