CYNTIA EICO HAYAMA NISHIDA

Controlling Biological Systems through Batch Reinforcement Learning

Doctoral thesis presented to the Escola Politécnica da Universidade de São Paulo to obtain the degree of Doctor of Science.

São Paulo

2020

CYNTIA EICO HAYAMA NISHIDA

Controlling Biological Systems through Batch Reinforcement Learning

Corrected Version

Doctoral thesis presented to the Escola Politécnica da Universidade de São Paulo to obtain the degree of Doctor of Science.

Concentration field:

Computer Engineering

Advisor:

Profa. Dra. Anna Helena Reali Costa

São Paulo
2020

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 19 de agosto de 2020

Assinatura do autor: Cyntia Eico Hayama Nishida

Assinatura do orientador: _____

Catalogação-na-publicação

Nishida, Cyntia Eico Hayama
Controlling Biological Systems through Batch Reinforcement Learning / C. E. H. Nishida -- versão corr. -- São Paulo, 2020.
126 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Inteligência artificial 2.Aprendizado de máquina 3.Aprendizado por reforço 4.Sistemas biológicos 5.Redes de regulação gênica I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

"A complex system that works is invariably found to have evolved from a simple system that works." - John Gaule

"We should be taught not to wait for inspiration to start a thing. Action always generates inspiration. Inspiration seldom generates action." - Frank Tibolt

ACKNOWLEDGEMENT

This doctoral thesis would not have been possible without the financial support of the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES – Finance Code 001). Besides a scholarship, CAPES also provided financial support for two conferences through Programa de Excelência Acadêmica (Proex).

Foremost, I would like to express the deepest appreciation to my advisor Prof. Anna for the continuous support of my doctoral study and research, for her patience, motivation, guidance and expert advice. It was a great honor to work with her for the last four years.

Special thanks to my qualification committee: Prof. Bianchi and Prof. Leliane for their constructive feedback, insightful comments and hard questions.

I thank all my fellow labmates for the support, interesting conversations and friendship. I also want to thank Prof. Anarosa and Prof. Jaime for their enthusiasm and help over the last years. Among the friends who have encouraged me during this thesis, I must single out Lucas and Guilherme for their support and immense belief in me.

Last but not the least, I am deeply grateful to my family for all the invaluable support, love and encouragement. Specially my mother Alice that deeply supported, comforted and believed me throughout my life. Although my father Shigenori was not able to see this work, his memory was with me all the way to the end.

ABSTRACT

Over the last decades, there has been an increase in interest in the study of biological networks (BT) to understand how the entities interact with each other during the execution of biological tasks and how to intervene in these networks for the treatment of diseases. A major challenge in the intervention of BTs is to discover when and what interventions should be applied in order to shift them to healthy phenotypes. A set of entity activity profiles, called basin of attraction (BOA), takes a BT to a specific phenotype; therefore, a healthy BOA drives the BT to a healthy phenotype. However, most proposals disregard BOAs and derive an intervention strategy avoiding certain entity activity profiles, which may disrupt the execution of biological tasks and cause other diseases. Additionally, without the complete observability of all entities, it is difficult to infer an accurate model and to identify whether the current BOA is healthy. Instead of generating an intervention strategy from an inaccurate model, we propose a novel framework that learns a strategy directly from a batch of experiences provided in advance. Our proposed framework, named Basin of Attraction Control Framework through Experiences (BOAConFE), combines BOA and machine learning techniques to derive intervention strategies that are capable of quickly shifting partially observable BTs to healthy BOAs, while reducing the number of interventions. BOAConFE computes the probability of observations being in healthy BOAs in order to use it to deal with partial observability and incorporate BOA knowledge into our framework. Building on BOAConFE, we propose multiple Steps Basin of Attraction Fitted Q-Iteration (mS-BOAFQI) that uses these probabilities together with a trajectory composed of multiple steps to define intervention strategies from experiences. We empirically demonstrate that BOAConFE can quickly shift a partially observable BT to healthy BOAs, while reducing the number of interventions. Our results highlight the benefits of using multiple steps to deal with partial observability and BOA probability to explore BOA information.

Keywords: Artificial Intelligence, Machine Learning. Reinforcement Learning. Biological Networks. Gene Regulatory Networks.

RESUMO

Nas últimas décadas, o estudo de redes biológicas (BT - do inglês textitBiological neTworks) cresceu em importância por possibilitar o entendimento de como entidades biológicas interagem entre si para realizar tarefas biológicas e como intervir nessas redes para tratar doenças pela aplicação de terapias. Um dos principais desafios da intervenção em BTs é descobrir quando e quais intervenções aplicar para torná-las mais saudáveis. Um conjunto de perfis de atividade de entidades, denominado bacia de atração (BOA), leva a BT em direção a um determinado fenótipo; desta forma, para uma BT ter um fenótipo saudável é necessário que ela esteja em uma BOA saudável. Entretanto, grande parte das propostas desconsidera a existência de BOAs e obtém estratégias de intervenção que evitam certos perfis de atividade de entidades, o que pode causar doenças. Além disso, sem a completa observação de todas as entidades envolvidas em uma tarefa biológica, é difícil inferir um modelo preciso e identificar se a BOA atual é saudável. Em vez de gerar uma estratégia de intervenção a partir de um modelo impreciso, nós propomos um novo arcabouço que aprende as estratégias diretamente de um conjunto de experiências coletado previamente. O arcabouço proposto, denominado *Basin of Attraction Control Framework through Experiences* (BOAConFE), integra o conceito de BOA e técnicas de aprendizado de máquina para calcular estratégias de controle capazes de levar BTs rapidamente para BOAs saudáveis, enquanto reduz a quantidade de intervenções. BOAConFE calcula a probabilidade de observações estarem em BOAs saudáveis para lidar com a observabilidade parcial e incorporar o conhecimento sobre BOAs no arcabouço. BOAConFE utiliza o método proposto *multiple Steps Basin of Attraction Fitted Q-Iteration* (mSBOAFQI) que usa essas probabilidades junto com uma trajetória composta por múltiplos passos para definir estratégias de intervenção a partir das experiências. Nós mostramos empiricamente que BOAConFE consegue levar rapidamente uma BT parcialmente observável para BOAs saudáveis, enquanto reduz o número de intervenções. Os nossos resultados destacam os benefícios de usar múltiplos passos para lidar com observabilidade parcial e a probabilidade de observações estarem em BOAs saudáveis para explorar o conhecimento sobre BOAs.

Palavras-chave: Inteligência Artificial, Aprendizado de Máquina, Aprendizado por Reforço, Sistemas Biológicos, Redes de Regulação Gênica.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

BN	Boolean Network
BNp	Boolean Network with Perturbation
BOA	Basin of Attraction
BOAConFE	Basin of Attraction Control Framework through Experiences
BOAFQI	Basin of Attraction Fitted Q-Iteration
BRL	Batch Reinforcement Learning
BT	Biological Network
DNA	Deoxyribonucleic Acid
FQI	Fitted Q-Iteration
FQI-Sarsa	Fitted Q-Iteration State-Action-Reward-State-Action
LSFTDI	Least-Squares Fitted TD(λ) Iteration
MDP	Markov Decision Process
mSBOAFQI	Multiple Steps Basin of Attraction Fitted Q-Iteration
RL	Reinforcement Learning
RNA	Ribonucleic Acid
Sarsa	State-Action-Reward-State-Action

LIST OF SYMBOLS

A	Action space for Markov Decision Processes
a	Action
$a^{'}$	Follow-up action
a_i^t	Action for experience i at time t
a^{ia}	Intervention action
a^{na}	Non-intervention action
B	A set of BNp
b	Number of bins in discretization
C	Cost function for Markov Decision Processes
c	Cost signal
D	Data set
$\mathcal{D}$	Desirable basin of attractions
F	List of prediction functions
$\mathscr{F}$	Experience set, given by $\{(o, a, r, c, o', a')\}$ or $\{(z, u, r, c, z', u')\}$
$\mathscr{F}_{\mathcal{D}}$	Desirable experience set
$\mathscr{F}_{\mathcal{U}}$	Undesirable experience set
$f^{(i)}$	Prediction function for the ith entity
g	Number of entities in a Boolean network
H	Max number iterations
h	Iteration number
i, j, k, l	Variable index
m	Number of steps in mSBOAFQI
n	Number of observable entities

N	Number of samples in a data set
n_r	Number of cycles
n_c	Number of control entities
O	Observation space
o	Observation
o'	Follow-up observation
o_i^t	Observation for experience i at time t
p	Entity perturbation probability.
Q	State-action value function
$Q*$	Optimal State-action value function
$\tilde{Q}$	Approximated state-action value function
$Q_{\mathcal{D}}$	State-action value function for desirable basins of attraction
$Q_{\mathcal{U}}$	State-action value function for undesirable basins of attraction
$\tilde{q}$	State-action value for a single experience
$\mathbb{R}$	Space of real numbers
R	Reward function for Markov Decision Processes
r	Reward signal
$r_{\mathcal{D}\mathcal{D}}$	Reward for being and staying in a desirable basin of attraction.
$r_{\mathcal{D}\mathcal{U}}$	Reward for being in a desirable and going to an undesirable basin of attraction.
$r_{\mathcal{U}\mathcal{U}}$	Reward for being in an undesirable and going to an desirable basin of attraction.
$r_{\mathcal{U}\mathcal{U}}$	Reward for being and staying in an undesirable basin of attraction.
S	State space
s	State

s'	Follow-up state
s_i^t	State for experience i at time t
T	Transition function for Markov Decision Processes
t	Time step
$\mathcal{TS}$	Experience set, given by $\{(o, a, \tilde{q}(o, a))\}$ or $\{(z, u, \tilde{q}(z, u))\}$
$\mathcal{TS}_{\mathcal{D}}$	Experience set for desirable experiences
$\mathcal{TS}_{\mathcal{U}}$	Experience set for undesirable experiences
$\mathcal{U}$	Undesirable basin of attractions
U	mSBOAFQI action space
V	State value function
X	A set of entities
$\mathbf{X}$	An ordered array of all entities
$\mathbf{x}$	A possible Boolean network state
$\mathbf{x}^t$	State of a Boolean network at time step t
x_i^t	Value of the ith entity in state at time step t
x_i	ith entity of a Boolean network
x_i'	ith entity of a Boolean network at the next time step
Z	mSBOAFQI state space
α	Learning rate
γ	Discount factor
κ	Reward for visiting a desirable state
Π	Space of all policies
π	Policy for agent behaviour
π^*	Optimal policy for agent behaviour
ψ	Penalization for applying an intervention action

σ	Attractor
Ω	Set of attractors
$\Omega_{\mathcal{D}}$	Set of desirable attractors
τ	Temperature parameter
$\boldsymbol{\zeta}$	Perturbation array
$\boldsymbol{\zeta}_i$	Perturbation variable

CONTENTS

1 INTRODUCTION

Treating, curing and preventing diseases are important topics since the beginning of human kind. Some diseases are genetic and are inherited from ancestors, others may be caused by external factors such as environmental, invader organisms or bad habits. Still, these factors affect the functioning of genes and, consequently, may improve or worsen the functioning of cells and organs. For example, the administration of too much oxygen caused an increase of blindness in premature infants worldwide by 1953, while too little oxygen resulted in a higher rate of brain damage and mortality (SILVERMAN, 2004). Until today, the optimal oxygen rate is still unclear and as Owen and Hartnett (2014) pointed out, data indicate that this blindness disease is multifactorial, including a genetic factor.

The genetic code is one of the main factors for defining an organism phenotype, that is, observable characteristics such as hair color. Within an organism, the cells of the skin, heart, eyes and any other cell differentiation types have the same DNA and what specializes cells in different types is which parts of the same DNA are active. Active sections of DNA work harmoniously to code proteins, which participate in almost all biological functions, such as hormone signaling, communication, defense and structural functions. In other words, an organism may have different systems, each one with different organs and tissues, which have different cell types. Different cell types produce different proteins, whose concentrations determine cell function. From entities scaling from DNA to populations or ecosystems, a biological system models how they work together to carry out a particular task, such as the digestion of food by digestive organs together with intestinal microbiome (CAMAZINE et al., 2001).

A biological system forms a complex network composed of several entities that interact among themselves in a self-organized manner (CAMAZINE et al., 2001). This network can be modeled as a biological network (BT), a graph with entities as nodes and edges as an influence from a source node to a target node. A positive link indicates that the source entity promotes and activates the target entity, while a negative link indicates suppression and inhibition. In this work, we consider a binary BT, therefore an entity can have only two values: 1 for active and 0 for inhibited. In this case, each

entity activity profile, also known as state, is an ordered array composed of the binary activity level of each entity at a given time. Given the current state and the interaction rules among entities, it is possible to predict the activity level of a particular entity at the next time step. By predicting all entities, the result is a prediction of the next network state.

Binary models are concerned with qualitatively describing interaction rules among entities and allow users to have a high-level understanding of the biological function (KARLEBACH; SHAMIR, 2008). Although continuous models are more precise and better represent biological systems, discretization has many advantages (GALLO et al., 2015): model inference is usually more efficient and effective, since it requires a smaller set of entity activity data to infer a good model; in addition, discrete data makes the decision process faster and less sensitive to noise (GARCIA et al., 2013).

When a biological task, such as cell division, is being performed, reactions happen in a chain of reactions called biological pathway, where products of reactions cause the next one (NELSON; COX; LEHNINGER, 2008). In the context of BTs, a BT goes to the starting state of this task and then follows a chain of states that leads to its execution. Therefore, a BT performs a biological task by following a particular chain of states that represents the steps necessary to execute it. After finishing this task, the BT dynamics converges to a sequence of states it cycles for a long time. This sequence, known as attractor, is linked to cell type and phenotype because it represents a stable point of equilibrium in entity activity levels (KAUFFMAN, 1992). A BT keeps cycling on an attractor until the biological task is restarted or until a disturbance occurs that changes the current state to a different one that represents a starting state of a path towards an attractor. The set of all starting states that go towards a particular attractor is part of the basin of attraction (BOA) of the attractor it goes to.

Mutations, perturbations or environmental factors move a BT to a different starting state or facilitate the transition from a healthy to a diseased phenotype (GROSS, 2011). When these factors happen and a network stops following the correct steps of a biological pathway, diseases may appear as a BT is not properly executing biological tasks. Certain types of treatments aim at intervening in the network to fix these problems by changing how entities affect the network. For example, some cancers evade the immune system by activating a defense cell checkpoint that suppress the immune

response. In immunotherapy treatments, the drug Nivolumab is used to inhibit this checkpoint (BRAHMER et al., 2012), which in turn boosts the immune system against cancer cells that use this checkpoint to evade the defense system. However, this type of drug affects the whole organism and leaves the immune response unchecked, what may cause autoimmune-like side effects as the authors pointed out.

Interventions affect the whole organism, unhealthy and healthy cells alike. Hence, it is important to design effective treatments that are capable of treating diseases with less side effects. Control techniques can support this endeavour, since they can determine a policy defining what actions (interventions) to perform in each state, to reach a certain goal (DATTA et al., 2003). In this intervention problem, the goal is to make an organism healthier by leading a BT to healthier BOAs and applying fewer interventions to achieve the goal while avoiding side effects.

A BT model helps to understand the long-term consequences of each action, simulating and/or predicting how it reacts when an action is applied and then using this simulation to plan future actuation. Although planning allows to find an optimal policy (BRYCE; VERDICCHIO; KIM, 2010; ERDOGDU; POLAT; ALHAJJ, 2017), the quality of its solution is highly dependent on the quality of the BT model. When the accuracy of the model cannot be guaranteed, the derived policy can perform poorly in a real-world biological system. Another alternative is to learn a policy directly from intervention experiences (FARYABI; DATTA; DOUGHERTY, 2007; ENGELHARDT, 2019), that are composed of transition samples together with a performance index. An example of an experience-based technique is batch reinforcement learning (BRL) that can learn a policy by realizing which actions return the best long-term results through experience samples (LANGE; GABEL; RIEDMILLER, 2012; SUTTON; BARTO, 2018).

Most BT intervention methods seek to avoid a set of states considered unhealthy and undesirable. Using the example of the defense system checkpoint, these methods consider all states that have the checkpoint active as undesirable and seek to avoid them. However it should be expected that a healthy pathway may have states with the checkpoint activated, otherwise the defense system could continuously attack healthy cells. When an intervention policy causes a BT to avoid states on a healthy pathway, a disruption in the execution of the task occurs and it could result in other diseases. In this work we use the terms healthy and desirable interchangeably, similarly to unhealthy

and undesirable.

Rather than avoiding states, it is possible to avoid BOAs with undesirable phenotypes. In this case, the goal is to make a BT reach any state within desirable BOAs and then let it naturally go to its attractor. In other words, in the absence of interventions and perturbations, once a BT is in a state of a desirable BOA, it follows a path towards a desirable attractor, where it remains until starting a task of interest. Unlike state-avoidance, BOA-avoidance tends to preserve pathways and phenotypes because the main goal is to shift a BT from undesirable to desirable BOAs and then letting it perform its tasks (BRYCE; VERDICCHIO; KIM, 2010).

The activity level of some entities may be difficult to measure due to complexity or cost of experiments, such as the oxidation of nuclear pores inside cells of the human senescence network (FURBER, 2019). Even though these entities are involved in the biological function being analyzed, their activity level cannot be observed by a learner. Consequently, it has to learn an intervention policy using only the information it can partially observe from the state signal.

In a partially observable BT, different states generate the same observation as the system does not perceive the value of not observed entities. When partial observability is not taken into account, a control system handles different states as a single one because it perceives them as a single observation. Since different states may require different actions, this strategy may not produce good results because the best action for a state may not be applied in another. For example, a state in a desirable pathway and a state in an undesirable attractor could generate the same observation. The former requires an action allowing the BT to follow a biological pathway, i.e., a non intervention action that does not apply any therapy or any drug. The latter requires an action forcing the BT out of the undesirable attractor, i.e., an intervention action that applies therapies or drugs capable of changing state transitions. If only one action is always performed in either case, or a disruption of a pathway occurs by always applying an intervention action in this observation or the BT does not exit the undesirable attractor by never intervening in it. Taking into account partial observability turns the problem more realistic and increases its complexity (BRYCE; VERDICCHIO; KIM, 2010).

These concepts provide the basis for the research described here, in which we

are concerned with developing a framework for intervening in a partially observable BT through machine learning techniques based on data, acquired from experiences. Our focus is on shifting a BT from undesirable to desirable BOAs in an efficient and effective way, by applying fewer interventions as possible to reduce the probability of side effects. Most research efforts are concentrated in model-based approaches, which are unfeasible for large BTs and require an accurate model. While experience-based proposals exist, they focus on state-avoidance problems or require BTs to constantly experiment actions on, which may be troublesome because BTs could be alive organisms. Thus, our research explores these and other opportunities available for improvements and provides solutions for open problems in the field.

1.1 Objectives

Our research efforts seek to combine biological networks and experience-based machine learning techniques to automatically generate an intervention policy capable of shifting partially observable biological networks from undesirable to desirable basins of attraction through a batch of experiences given in advance. As a result, the proposed approach must have all phases required for learning a control policy from experiences and without having prior knowledge regarding the network. The experience-based method should be data efficient, since we assume that there is no biological system available to interact with during learning and the number of experiences is limited. The computed policy should be effective by being concerned with applying fewer interventions as possible to avoid side effects, while quickly guiding a network towards desirable basins of attraction.

1.2 Contributions

In this thesis, we contribute a novel framework to shift a partially observable biological network from undesirable to desirable basins of attraction through experience-based techniques. Our framework is capable of automatically computing a control policy from experiences, without having prior knowledge about the network and without requiring a network to experiment on during learning. In particular, we use reinforcement learning techniques to learn an intervention policy from previously observed ex-

periences. The technique explored is batch reinforcement learning (BRL). For building this framework, we propose how to compute information regarding basins of attraction from a sequence of observations, which is then used throughout our framework to deal with partial observability. We propose a new BRL method that is concerned in shifting basins in partially observable biological networks. Lastly, our proposed policy seeks to reduce the number of interventions for a network already in desirable BOAs, where interventions are no longer necessary to improve an organism health and could result in side effects.

During this research, we published a number of articles in conferences, workshops and recognized journals (see Appendix B). To the best of our knowledge, our framework is the first one proposed to shift basins of attraction using experience-based approaches that does not require a biological network to experiment on during learning and is effective in reducing the number of applied interventions.

1.3 Organization

This first chapter (Chapter 1), *Introduction*, introduced our work, provided a context to the whole manuscript, described our objectives, and summarized the achieved contributions. In Chapter 2, *Background*, we cover the fundamental concepts that form the foundation for this research proposal, specifically biological networks, batch reinforcement learning, and the biological network intervention problem. In Chapter 3, *Related Work*, we describe related work, their advantages and disadvantages, and how our proposal seeks to fill their gaps. Chapter 4, *Proposal*, details our research proposal with the basic structure of our framework and the details of each stage. We then describe the conducted experiments and discuss the achieved results in Chapter 5, *Experiments*. In the final chapter (Chapter 6), *Final considerations*, we provide a conclusion of the conducted research and suggest future work on the topic.

In the appendix, we explain a proposed technique that is not part of the framework for generalization purposes (Appendix A) and academic achievements during the Ph.D. program (Appendix B).

2 BACKGROUND

In this chapter, we present the fundamental concepts, algorithms and methodologies that are important for the understanding of our proposal. Section 2.1 introduces the basic theory of biological networks, Section 2.2 provides an overview for batch reinforcement learning and Section 2.3 explains the theory behind the intervention of biological networks.

2.1 Biological Networks

Biological interactions among entities happen at every moment inside an alive organism as they are necessary for the completion of most biological tasks. For example, the digestion of food in complex organisms requires the interaction of many organs and the microbiome. In a molecular level, this is a complex process involving the production of many products that are used over the digestion process, e.g., for breaking food down and communication. While interactions in a higher level allow physicians to observe a patient's health in practical procedures, interactions in molecular levels are the basis for reactions and phenotypes. Both perspectives are important and a mix of them may provide a more robust understanding of how a biological system works.

A biological network (BT) models dependencies among entities that work together to accomplish a specific biological task. A BT can be represented by a directed graph as shown in Figure 1. It has entities as nodes and the edges represent the influence of a source node on a target node, which can be positive or negative to respectively promote or inhibit an activity of the target entity. Influences and interactions between entities can occur both directly and indirectly; indirect interactions are carried out by products of the source entities. For example, Figure 1 shows the yeast cell cycle (cell division) modelled by Li et al. (2004) with 11 of the most important genes involved in this function. While genes do not interact directly, they (source entities) produce products, such as proteins and RNA messenger, that can promote or inhibit the level of activity of other genes (target entities). This type of biological network is often called gene regulatory network, since it models how genes regulate each other.

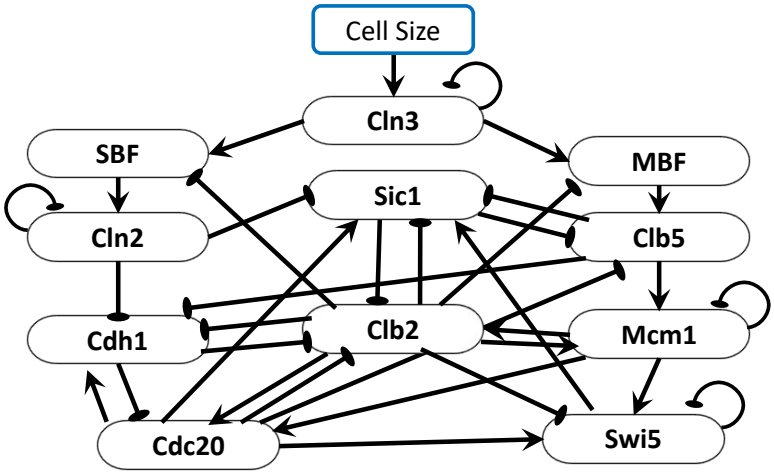


Figure 1: Biological network of the yeast cell cycle adapted from (LI et al., 2004). It has 11 entities interacting among themselves and one input signal (cell size) indicating the start of cell division task. Arrows indicate positive influences and oval arrows, negative.

Definition 1. *A biological network is a graph* $G = \langle X, E \rangle$ *composed of:*

- *A set* X *of* g *nodes, each representing an entity;*
- *A set* E *of edges, where a positive edge represents that a source entity influences the target entity to activate it and a negative edge to inhibit it.*

Several computational models have emerged to describe and analyze BTs. Karlebach and Shamir (2008) classify them into three classes: logical, continuous and single-molecule level. Logical models are concerned with qualitatively describing interactions and allow users to have a high-level understanding of the biological function being analyzed. Instead of using discrete values for entity activity, continuous models use real values that increase the precision for quantifying reactions. Continuous models are particularly useful when it is necessary to understand and simulate the effect of different concentrations of a molecule. The last class studies how stochasticity affects the relationship between entities.

Albeit being a simple model, discrete models have been shown to reproduce many BT dynamics and have modeled networks such as cancer pathway (FUMIÃ; MARTINS, 2013) and human senescence (FURBER, 2019). Next we explain the Boolean network (Subsection 2.1.1), which is one of the most used logical models for modelling and intervening in BTs. Then, we introduce what is a basin of attraction (Subsection 2.1.2) and how it is structured in a Boolean network. It is important to note that the objective

of our thesis is to intervening in a BT without requiring its model, and here Boolean networks are used to explain how this problem works and to execute and analyze experiments in simulations.

2.1.1 Boolean Network

Proposed by Kauffman (1969), Boolean network (BN) describes generic and essential relations in the form of Boolean rules between variables. It has an ordered set of g Boolean variables $x_1, x_2, ..., x_g$, whose values at the next time step are predicted by a list of Boolean prediction functions $f^{(1)}, f^{(2)}, ..., f^{(g)}$, one function for each variable. By modelling a biological system using a BN, each variable represents an entity and each prediction function describes how variables interact among themselves to define the value of a particular entity. In this modeling, the activity level of each entity x_i is discretized into two values: inhibited ($x_i = 0$) and active ($x_i = 1$), in which an active entity could be considered as being sufficiently active to influence the network (SHMULEVICH; DOUGHERTY, 2010).

At each discrete time step, BN synchronously updates the value of each entity x_i to x_i'. It applies the current value of all entities, known as state $\mathbf{x} = \langle x_1, ..., x_g \rangle$, on each prediction function $f^{(i)}$ that returns x_i' as a result. Consequently, a given network with $g = |\mathbf{x}|$ entities has 2^g possible states that exponentially grow with the number of entities. For convenience purposes, we represent the state tuple $\langle x_1, ..., x_g \rangle$ as $x_1..x_g$. For example, instead of state $\langle 0, 1, 0 \rangle$, we use state 010, where each binary number denotes the activation of its respective entities $x_1 x_2 x_3$. Note that it is possible to extend a BN to consider different degrees of discretization, which accounts for different degrees of entity activity.

Definition 2. *A Boolean network $BN = \langle X, F \rangle$ is composed of:*

- *A set X of g entities; $\mathbf{X}$ is an ordered array formed by all entities of X, $\mathbf{x} \in \mathbf{X}$ defines a possible BN state and $\mathbf{x}^t$ defines the network state at time step t.*
- *A list of prediction functions $F = \left\{ f^{(1)}, ..., f^{(g)} \right\}$ describing the relationship between entities. Each function $f^{(i)}$ predicts the next value x_i' of the i^{th} entity given the*

current BN state $\mathbf{x}$*:*

$$x_i' = f^{(i)} : \{0,1\}^g \rightarrow \{0,1\} \text{ and } x_i' = f^{(i)}(\mathbf{x}).$$

One way to model a BN from a BT is using influence values proposed by Li et al. (2004). Their proposal has a matrix of influence M_f, where each element m_{ij} has the influence value of an entity i over entity j:

- $m_{ij} = 1$, entity i activates entity j;
- $m_{ij} = -1$, entity i inhibits entity j;
- $m_{ij} = 0$, entity i does not influence entity j.

For computing the next value of an entity j, BT multiplies the current state with column j of M_f. If the dot product between the two sequences is positive, entity j will be 1 (active); if it is negative x_j will be 0 (inhibited); and if it is 0 then the value of x_j does not change between current and next time steps. Li et al. (2004) applied this idea to their proposed yeast cell cycle BT, shown in Figure 1, that has 11 key genes selected from ≈ 800 genes involved in this function. Although using all 800 genes would be more complete and biological meaningful, it would be unfeasible to determine the influence values for all ordered pairs by hand.

Instead of building a BT using biological knowledge, inference methods infer a BN from entity activity data composed of N state measurements $\mathbf{x^t}|t = 1, .., N$. The data set must be properly preprocessed when necessary, so it has only binary measurements for entities involved in the biological function being analyzed. Figure 2 shows an example of how to infer $f^{(1)}$ from time-series data set composed of three entities. For example, state $\mathbf{x} = 111$ appears three times, from $t = 8$ to $t = 10$, and in the next time step, $x_1^9 = 1$ and $x_1^{10} = 1$. Thus, when the current state is 111, the next value of x_1 is 0 in no case and x_1 is 1 in two cases. As $x_1^{t+1} = 1$ count is higher than $x_1^{t+1} = 0$ count, this method chooses 1 as the result of $f^{(1)}(111)$. Therefore, this method works as follows: for each entity x_i and for each possible state $\mathbf{x}$, the method counts how many times $x_i^{t+1} = 0$ and $x_i^{t+1} = 1$ for all $\mathbf{x}^t = \mathbf{x}$. The chosen result for $f^{(i)}(\mathbf{x})$ has the highest count, and when both values are tied it chooses randomly 0 or 1. After inferring $f^{(i)}$ for each entity $x_i \in X$, the inference process ends and the BN can be used for further analysis.

Time	1	2	3	4	5	6	7	8	9	10
x_1	0	0	1	0	0	1	1	1	1	1
x_2	0	1	0	0	1	0	1	1	1	1
x_3	0	1	1	1	0	0	0	1	1	1

	x_1^{t+1}		
$\boldsymbol{x}^t$	0	1	$f^{(1)}$
000	1	0	0
001	1	0	0
...	...	...	
110	0	1	1
111	0	2	1

Figure 2: Example of Boolean network inference method with 3 entities. Left table represents a data set composed of a sequence of state transitions ($\mathbf{x}^t | t = 1, ..., 10$) and right table has 0's and 1's counting for x_1 and its chosen $f^{(1)}$.

Once a BN is available, it is possible to compute and simulate the next state $\mathbf{x}^{t+1}$ of the network by applying its current state $\mathbf{x}^t$ on each prediction function $f^{(i)}$. Since the future state $\mathbf{x}^{t+1}$ depends only on the current state $\mathbf{x}^t$, the BN state transition has the Markov property (SUTTON; BARTO, 2018; SHMULEVICH; DOUGHERTY, 2010), which defines the independence of the future from the past given the present.

2.1.2 Attractors and Basins of Attraction

Starting from a given state, the next one is deterministically computed by F, which means it is possible to predict with certainty next and future states a network is going to pass by. Since there is a finite number of states, BN follows a path until reaching a previously visited state, that indicates the start of a cycle. This sequence of states a BN cycles is known as attractor and is linked to cell type and phenotype (KAUFFMAN, 1992) as they keep a stable pattern of entity activity, such as melanin in eye color.

Definition 3. *An attractor σ is a sequence of k ($1 \leq k \leq 2^g$) states $\sigma = (\mathbf{x}_{i1}, ..., \mathbf{x}_{ik})$, that forms a strongly connected component , where g is the number of entities and 2^g is the number of possible states.*

When a BN starts a biological function, entity activity varies as needed to perform the task and then returns to the same pattern defined by the attractor. For example, in the yeast cell cycle task represented in Figure 3, BN stays most of the time cycling in G_1 *stationary* (attractor) while waiting for a signal to start cell division. G_1 *stationary* indicates a stable cell growth, until it reaches cell size checkpoint and really commits to start the division process. This signal activates gene *Cln3* (first entity in state array $\mathbf{x}$) and BN goes from the G_1 *stationary* to excited G_1 state (Start) in Figure 3. Then BN

goes through a chain of states representing different phases of mitosis: *S* (DNA replication), *G2* (cell growth) and *M* (cell division), until returning to G_1 *stationary* where it stays for a long time, while waiting for a signal to restart the division process. This chain of states the BN goes through to perform a biological function is known as biological pathway and Figure 3 represents the yeast cell cycle pathway.

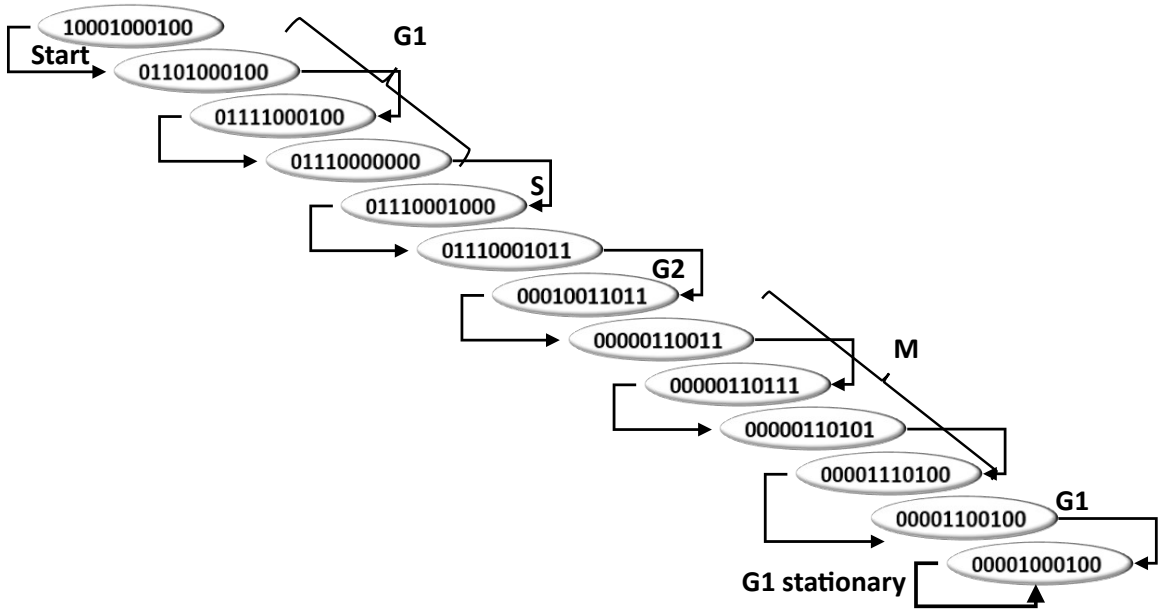


Figure 3: Biological pathway of the yeast cell cycle proposed by (LI et al., 2004). It starts at the excited G_1 state (10001000100) and finishes at G1 stationary (00001000100), where it cycles until receiving a new stimulus to restart this process.

In a different starting state, BN would follow a different chain of states until reaching an attractor. These different starting states that eventually reaches a particular attractor σ forms a set known as basin of attraction (BOA) of σ. In other words, these transient states that belong to the BOA of σ drive a BN towards σ. Figure 4 shows the state transition diagram of the yeast cell cycle, i.e., states and a link to their next state. Thiw figure shows seven disconnected subgraphs, each representing a BOA leading to a specific attractor (state with a self-loop). States and arrows in the biological pathway are highlighted and the desirable BOA has this pathway. Since the desirable BOA is the largest one, there is higher probability of a randomly selected starting state be in this BOA. However, if it is not, BN reaches an undesirable attractor that has an undesirable phenotype and remains there until this process restarts and a new starting state is selected.

Definition 4. *A basin of attraction is a set of* k *states* $\{\mathbf{x}_{i1}, .., \mathbf{x}_{ik}\}$ *that eventually reaches*

a particular attractor σ*, that is,* $\forall j, \forall t, (\mathbf{x}^t = \mathbf{x}_{ij}) \Rightarrow (\mathbf{x}^\infty \in \sigma)$.

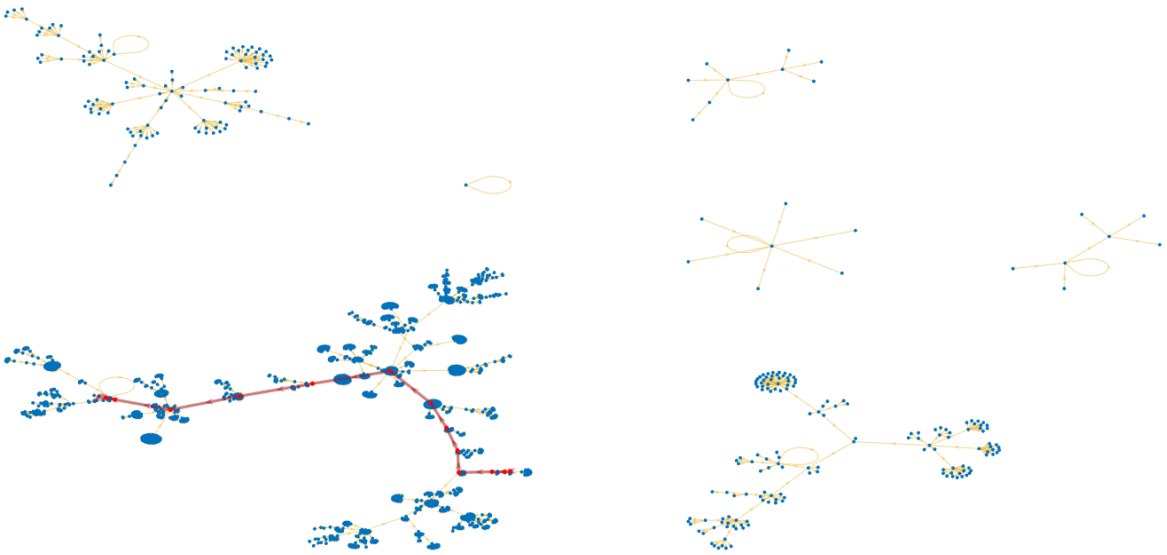


Figure 4: State transition diagram of the yeast cell cycle proposed by Li et al. (2004). Each dot represents a state and each arrow represents a transition from a state to its next state. It has 7 disconnected subgraphs, from which the yeast cell cycle pathway is represented by highlighted nodes and arrows.

2.1.3 Stochasticity

Although BN state transition is deterministic, entity activities inside BTs are intrinsically stochastic (ELOWITZ et al., 2002). Biological factors such as velocity of proteins accumulation, availability of nutrients and signal transmission may take a shorter or longer period of time (ZHANG et al., 2006), that is, the value of one or more entities may change before or after a discrete time step. Furthermore, external factors can interfere in entity activity level, such as temperature that can denature proteins and accelerate or decelerate reactions. For considering stochasticity, Zhang et al. (2006) modelled the budding yeast cell cycle using a temperature-like parameter characterizing the noise in the system. At low temperatures, this BT becomes almost as deterministic as the fission yeast cell cycle proposed by (LI et al., 2004).

Instead of considering a temperature parameter, an extension of BN called BN with perturbation (BNp) (SHMULEVICH; DOUGHERTY, 2010) simulates cell stochasticity by adding perturbations. With a small probability p a perturbation occurs and flips the value of a given entity from 0 to 1 and vice-versa. At each time step, a BNp tosses

a biased coin for each entity to account for perturbations. All perturbed entities have their values flipped and unperturbed ones keep the same value for the next time step. If no perturbations occurred, BNp acts as a BN, and uses prediction functions $f^{(i)}$ to update the value of each entity x_i.

Formally, perturbations are characterized by a Bernoulli random variable array $\zeta = (\zeta_1, \zeta_2, ..., \zeta_g), \zeta_i \in \{0, 1\}$, $P(\zeta_i = 1) = p$, and g is the number of entities (SHMULEVICH; DOUGHERTY, 2010). Given $\mathbf{x}$ and ζ:

$$x_i' = \begin{cases} f^{(i)}(\mathbf{x}), & \text{if } \zeta_i = 0 \ \forall \ i \\ x_i, & \text{if } \zeta_i = 0 \text{ and } \exists \ \zeta_j = 1 | j = 1, .., g \\ \overline{x_i}, & \text{otherwise.} \end{cases} \tag{2.1}$$

Definition 5. *A Boolean network with perturbations $BNp = \langle X, F, p \rangle$ is composed of:*

- *A set X of g entities; $\mathbf{X}$ is an ordered array formed by all entities of X, $\mathbf{x} \in \mathbf{X}$ defines a possible BNp state and $\mathbf{x}^t$ defines the network state at time step t.*
- *A list of prediction functions $F = \left\{f^{(1)}, ..., f^{(g)}\right\}$ describing the relationship between entities. In the absence of perturbations, each function $f^{(i)}$ computes x_i' from $\mathbf{x}$:*

$$x_i' = f^{(i)} : \{0, 1\}^g \rightarrow \{0, 1\} \text{ and } x_i' = f^{(i)}(\mathbf{x}).$$

- *A probability p of an entity suffering a perturbation flipping its value for the next time step.*

Usually, a biological function has a large BOA, making it stable towards perturbations in the system. For example, high temperature could denature a certain protein and the entity that produces it would be considered inhibited. This new state may belong to a different BOA with a different attractor (phenotype) and not necessarily desirable. When a desirable BOA has many states, there is a higher probability that this new state is also in the large BOA and the phenotype would not change. However, since perturbation makes all states reachable from all others, a BNp could eventually reach an undesirable BOA with an undesirable phenotype. For example, in Figure 4 a perturbation can make a BT leave the desirable BOA and go to an undesirable one. When this happens, one possibility is to wait for new perturbations that can take the

BT back to desirable BOAs, which is not an acceptable solution for many diseases. A better alternative is to learn how to effectively guide a BT back to desirable BOAs by applying interventions such as medicines and therapies. Therefore, in Figure 4, when the BT is out of the desirable BOA, it is important to find how to intervene in it in order to guide it back to the desirable BOA.

It is noteworthy that there are other Boolean networks that account for perturbations, such as the probabilistic Boolean network (SHMULEVICH et al., 2002). In this work we use BNp because attractors and BOAs are well defined and there is a straightforward way to compute them.

2.2 Learning from Experiences

When someone is sick, a physician analyzes the current situation and decides which action should be taken to treat the patient. Then, the patient's situation may change as a result of the applied treatment (action to intervene in a patient's health) or the absence of it (action to not intervene). In the next medical consultation, the physician analyzes the new situation and makes a new decision about what action should be taken. This process may repeat until the patient's health improves and the patient stops taking medicines and doing treatments (absence of interventions). When this process is translated to BT language, at each discrete time step, a decision maker selects what action should be taken in the current state in order to effectively guide a BT to desirable phenotypes.

Since BNp is a stochastic process that happens in a chain of events that has the Markov property, its decision process can be modeled as a Markov decision process (SUTTON; BARTO, 2018; SHMULEVICH; DOUGHERTY, 2010). When the BNp model [1] is available, it is possible to apply planning methods to solve this decision process. However, for complex organisms, a complete and accurate model is usually not available because a complex biological function with many variables is difficult to infer. When an accurate model is not available, it is possible to solve the Markov decision process (Subsection 2.2.1) by applying experience-based techniques such

[1] From the BNp model (Definition 5), it is possible to compute all possible states, all state transitions and their respective probability of happening, and a performance index indicating the desirability of performing a given action in a given state.

as reinforcement learning (Subsection 2.2.2). In this work, we apply a sub class of reinforcement learning called batch reinforcement learning (Subsection 2.2.3), which instead of learning by direct interacting with a BT, it learns a intervention strategy using a batch of past experiences given in advance.

2.2.1 Markov Decision Process

Proposed by Bellman (1957), a Markov decision process (MDP) is a framework modelling decision making on discrete time processes with stochastic transition probabilities. At each time step t, a process is at a state s, a decision maker chooses and applies an action a that makes the process probabilistically go into a follow up state s' at the next step and a reward r and a cost c return to the decision maker indicating how well the process performed towards the goal. Solving an MDP problem is to find a policy π that defines which action should be executed in each state.

Definition 6. *A Markov decision process (MDP) is described by a tuple* $\langle S, A, T, R, C \rangle$*:*

- S *is a finite set of possible states.*
- A *is a finite set of possible actions that the process can execute.*
- T *is a transition function* $T : (S, A, S) \rightarrow [0, 1]$ *indicating the probability of a process transitioning from state* $s \in S$ *to* $s' \in S$ *after executing action* $a \in A$ *in* s*, and* $\sum_{s' \in S} T(s'|s, a) = 1$*.*
- R *is a reward function* $R : (S, S) \rightarrow \mathbb{R}$ *that returns the reward of transitioning from* $s \in S$ *to* $s' \in S$*.*
- C *is a cost function* $C : (A) \rightarrow \mathbb{R}$ *that returns the cost of applying* $a \in A$*.*

An MDP runs over a number of time steps that defines the horizon of an MDP:

- Finite horizon: there is a finite and fixed number of time steps and the process terminates after completing the last one.
- Infinite horizon: in which there is no stopping condition and the process runs for an infinite number of time steps.

- Indefinite horizon: the process only stops when it reaches an absorbing goal state, therefore it runs over an arbitrary number of steps and eventually terminates.

In a finite horizon, there is a fixed number of time steps, in which the decision maker chooses and applies an action. In the BT intervention problem, it can be interpreted as the treatment window for a curable disease, in which a patient should be cured after the treatment, but when she is not, the treatment does not continue. Another possibility is an indefinite horizon with the goal of curing patients regardless of the treatment window. For not curable diseases, such as diabetes, the constant use of medicines or diet is necessary to keep the disease under control, which could be considered as an infinite horizon and it is important to keep a BT away from undesirable BOAs. An advantage of infinite horizon is the possibility of ensuring that a disease does not relapse, since preventive actions could be applied before symptoms reappear. For example, for people who needs a constant monitoring and application of insulin, there are continuous glucose monitoring devices that checks a patient glucose level from time to time and can notify her and/or apply a dose of insulin whenever necessary (ENGLER; ROUTH; LUCISANO, 2018). In this work, we choose to use infinite horizon because we consider perturbations and assume that diseases may come back and/or are not curable.

Rewards and costs implicitly specify the goal of a problem, thus reward and cost functions should indicate the desirability of an experience given the objective (SUTTON; BARTO, 2018). A policy that maximizes the expected value, represented by V, makes a system perform the task optimally. In other words, while reward and cost functions indicate the immediate desirability of an experience, a value function specifies long-term desirability. In the infinite horizon, V can be computed by:

$$V(s) = E\left[\sum_{t=0}^{\infty} \gamma^t (R(s, s') - C(a))\right], \tag{2.2}$$

where γ is a discount factor $(0 < \gamma < 1)$ that codifies the horizon in which rewards and costs are relevant. Using this formulation, rewards and costs acquired earlier are more relevant than acquired later, and this ensures that their sum is finite. This property is suitable for the BT intervention problem, as it favors actions treating a patient in the be-

ginning of a treatment rather than in the end. Future rewards and costs are dependent on what actions a decision maker will take, hence, value functions are defined when following a particular policy π.

Definition 7. *The value function of a policy π is a function $V^{\pi} : S \rightarrow \mathbb{R}$, in which $V^{\pi}(s)$ provides the expected value for a state s when π is followed:*

$$V^{\pi}(s) = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^{k}(r_{t+k} - c_{t+k})|s_t = s\right\}, \forall s \in S. \tag{2.3}$$

An important property of value functions is that they satisfy a particular recursive property (SUTTON; BARTO, 2018). For any policy π and any state s, the expression of Equation 2.3 can be recursively defined by:

$$V^{\pi}(s) = \sum_{s' \in S} T(s'|s, \pi(s))\Big[R(s, s') - C(\pi(s)) + \gamma V^{\pi}(s')\Big], \tag{2.4}$$

which is known as Bellman equation for evaluating $V^{\pi}(s)$ (BELLMAN, 1957; SUTTON; BARTO, 2018). It computes the expected value of a state as the immediate reward minus the immediate cost plus the discounted expected values of all possible next states weighted by the transition probabilities of reaching them.

In the case of state-action value, function $Q^{\pi} : S \times A \rightarrow \mathbb{R}$ denotes the expected value of starting at state s, executing action a and then following π in subsequent states:

$$Q^{\pi}(s, a) = \sum_{s' \in S} T(s'|s, a)\Big[R(s, s') - C(\pi(s)) + \gamma V^{\pi}(s')\Big]. \tag{2.5}$$

A policy π is considered optimal when it receives more rewards and fewer costs than any other policy. In other words, it maximizes the value function for each $s \in S$ and each $a \in A$ in Equation 2.5.

Definition 8. *Being Π the set of all policies for an MDP, a policy $\pi^* \in \Pi$ is optimal if $\forall \pi' \in \Pi, \forall s \in S$, $Q^{\pi*}(s, \pi^*(s)) \geq Q^{\pi'}(s, \pi'(s))$. Moreover, if Q^* is optimal then:*

$$\pi^*(s) = \arg\max_{a} Q^*(s, a). \tag{2.6}$$

When more than one action is optimal for a given state, any one can be selected in Equation 2.6.

2.2.2 Reinforcement Learning

Reinforcement learning (RL) is a learning paradigm, in which problems are usually formulated as an MDP (SUTTON; BARTO, 2018). In this paradigm, a learner does not have a model of the environment, hence does not know functions T and R and has to learn what actions will lead it to perform the task optimally by exploring the space of possible strategies in a given environment, receiving a reward and cost and deducing a policy from her observations. As shown in Figure 5, at each time step t a learner observes the current state s^t of the environment and performs an action a^t. As a result, the environment updates the state to the next state s^{t+1}, gives a reward signal r^{t+1} and a cost signal c^{t+1} indicating how well the process performed. Then, at the next time step the learner applies the next action a^{t+1} in s^{t+1}. Using this information, RL iteratively improves a policy and learns how to actuate in this environment.

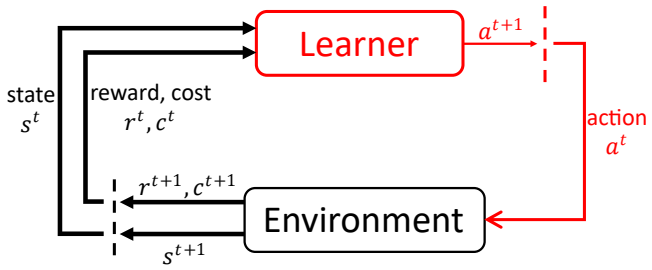


Figure 5: A standard Reinforcement Learning setup. After performing an action at the current state, a learner receives a reward, a cost and an updated state from the environment. Adapted from Sutton and Barto (2018)

One way to solve an MDP through RL is using a method called Sarsa (RUMMERY; NIRANJAN, 1994) that interacts with an environment and updates the quality of each state-action pair defined in the Q-function. At each time step, a process goes from a state-action (s, a) pair to the next state-action (s', a') pair, receives a reward r, a cost c, and then updates the Q-function:

$$Q_{t+1}(s, a) = (1 - \alpha)Q_t(s, a) + \alpha\left[r - c + \gamma Q_t(s', a')\right], \tag{2.7}$$

where α is a learning rate $(0 < \alpha < 1)$ that determines how much the newly acquired information replaces the previous one and γ is a discount factor $(0 < \gamma < 1)$. In this formulation, the newly acquired information $[r - c + \gamma Q_t(s', a')]$ is the immediate reward minus the immediate cost received in the interaction plus the discounted expected value for executing the next state-action pair at the next time step. After a certain

number of interactions with an environment, the value stabilizes in Q^* and a system computes an optimal policy using Equation 2.6.

Other common RL algorithm is Q-Learning (WATKINS; DAYAN, 1992), which updates $Q(s,a)$ without taking into account the policy being followed:

$$Q_{t+1}(s,a) = (1-\alpha)Q_t(s,a) + \alpha\left[r - c + \gamma \max_{a^{\#}} Q_t(s', a^{\#})\right]. \tag{2.8}$$

Therefore, while Equation 2.7 considers the action a' applied at s' (on-policy update), Equation 2.8 assumes that the optimal policy is being followed (off-policy update) (SUTTON; BARTO, 2018).

Equation 2.7 and Equation 2.8 show a local update of a Q-function for a single state-action pair (s,a). For this update to propagate to preceding state-action pairs, it is necessary to revisit (s,a), which is when the preceding state-action pair uses the updated $Q(s,a)$ in its computation (the newly acquired information). Thus, for an update to propagate further, it is necessary to revisit (s,a) several times. This problem, known as exploration overhead, slows the learning process and requires more interactions with an environment until convergence (LANGE; GABEL; RIEDMILLER, 2012). This means that during training many interactions are performed on a trial-and-error basis until Sarsa and Q-Learning can provide a good policy (SUTTON; BARTO, 2018).

In biological experiments, it is difficult to maintain an environment (BT) to interact with, mainly when many interactions are necessary and the environment is an alive organism. In the previous example given in Introduction (Chapter 1), there was a hard and slow ethical battle to decide how to verify whether oxygen was really causing blindness. Should the extra dose of oxygen be cut with the risk of increasing the rate of death? Or should they receive the extra dose with a risk of blindness? Which hospitals should do this experiment? How to inform parents about the risks? What about public opinion? Although scientists still do not know the optimal oxygen rate, current doses decreased blindness risks and increased the rate of survival in comparison to 1950's numbers (OWEN; HARTNETT, 2014). This is not an ideal scenario, but risking infants' life is far from desirable. Therefore, in biological problems, it may be vital to use data-efficient methods that require fewer interactions to produce high quality policies.

2.2.3 Batch Reinforcement Learning

Batch reinforcement learning (BRL) is a subfield of RL that also seeks to compute a policy maximizing the Q-function. The difference between the two lies in the setting, while conventional RL learns at each step while interacting with an environment, BRL learns a policy in an optimized way from a set of experiences given in advance (ERNST; GEURTS; WEHENKEL, 2005). Since it is more data-efficient than RL, BRL is more appropriate for cases in which there is a limited number of experiences samples to learn from (LANGE; GABEL; RIEDMILLER, 2012). Figure 6 shows the phases of a BRL problem, it starts with the collection of a set of experiences $\mathcal{F} = \{(s, a, r, c, s', a')\}$ given by a specialist or collected by interacting with an environment using an arbitrary policy. Then there is a learning phase when a BRL method learns a control policy from this set. More cycles of exploration-learning can be done and, in this case, the computed policy is an intermediate exploratory policy used to collect more experiences and further explore the environment. After a stopping condition, no experiences are collected anymore and the last computed policy is not further improved, Then, this final policy stays fixed and is applied for controlling the system in the application phase.

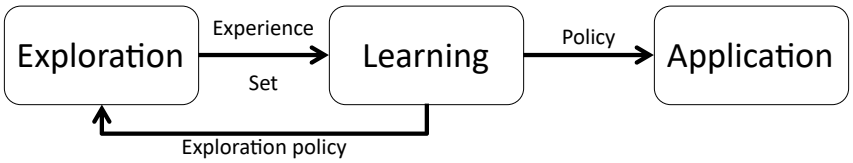


Figure 6: Batch reinforcement learning phases: Collecting experiences with an exploratory policy, learning a policy from the experiences through a BRL method, more cycles of experience collection and learning can be applied until the obtained policy is used in the environment as a fixed policy without further improvements. Adapted from (LANGE; GABEL; RIEDMILLER, 2012).

BRL was firstly designed to have a single cycle of exploration-learning and the learned policy is used to control the system without further improvements. This pure batch approach was later modified to incorporate several cycles of exploration-learning, which is known as growing batch (LANGE; GABEL; RIEDMILLER, 2012). In both cases, there is no guarantee the experience set really represents the environment dynamics, so the goal of BRL is finding the best possible policy from the available batch of experiences. Since this batch has a limited number of experiences, BRL methods should use it efficiently to extract as much knowledge as possible, thus they usually

apply an experience many times and use an approximation function to accelerate the learning process. Approximation functions, like regression tree or neural network, fits a curve on all points and can return values for points not observed yet (BUSONIU et al., 2010).

Convergence to a solution depends on the experience set and on the approximation function selected. If the function has high variance, convergence is not guaranteed. The quality of a solution depends on the quality of the experience set and if the approximation function is capable of approximating the optimal Q-Function with high accuracy (LANGE; GABEL; RIEDMILLER, 2012).

One of the most used BRL algorithms is Fitted Q-Iteration (FQI) (ERNST; GEURTS; WEHENKEL, 2005), which can be considered the batch version of Q-Learning. For each iteration h and for each experience in the experience set $\mathscr{F}$, FQI computes an update of the Q-value for a **particular** experience i $(s_i, a_i, r_i, c_i, s'_i, a'_i)$:

$$\tilde{q}(s_i, a_i) = r_i - c_i + \gamma \max_{a^{\#} \in A} \tilde{Q}^{h-1}(s'_i, a^{\#}_i). \tag{2.9}$$

Then, it mounts a training set $\mathcal{TS} = \{(s, a, \tilde{q}(s, a))\}$ composed of the state-action pair (s, a) and its updated estimate of the expected reward for a particular experience. When $\mathcal{TS}$ has the updated $\tilde{q}(s, a)$ for all experiences, a supervised learning method trains an approximation function $\tilde{Q}^h$ on the training set $\mathcal{TS}$. Supervised learning techniques are artificial intelligence methods that train a function to map inputs to outputs of a training set. In this case, the pair (s, a) is the input and the desirable output is $\tilde{q}(s, a)$. Thus, the supervised learning method trains an updated Q-function using the whole $\mathcal{TS}$ containing the new estimates of state-action pairs of all experiences. This generates an updated Q-function that represents the current estimate of the expected value for all possible state-action pairs.

For the batch version of Sarsa, called Fitted Q-Iteration Sarsa (FQI-Sarsa) (PEDNAULT; ABE; ZADROZNY, 2002), Equation 2.9 changes to:

$$\tilde{q}(s_i, a_i) = r_i - c_i + \gamma \tilde{Q}^{h-1}(s'_i, a'_i). \tag{2.10}$$

Algorithm 1 presents FQI-Sarsa, where in line 5 there is the update of $\tilde{q}(s_i, a_i)$ followed by its inclusion on training set $\mathcal{TS}^h$. In line 8 the supervised learning method trains an

approximation function $\tilde{Q}^h$ on $\mathcal{TS}^h$ that represents the updated estimate of the expected reward of a state-action pair for iteration h. Finally, after repeating this process for H times, FQI-Sarsa returns $\tilde{Q}^H$ that can be used to compute a deterministic policy using Equation 2.6.

Algorithm 1: Fitted Q Iteration Sarsa

Input : Discount factor γ, maximum number of learning iterations H, number of experiences N, experience set $\{(s_i, a_i, r_i, c_i, s'_i, a'_i) | i = 1, ...N\}$

$\tilde{Q}(s, a) = 0, \ \forall \ s \in S, a \in A$;
for *h = 1, 2, ..., H* **do**
  $\mathcal{TS}^h = \emptyset$;
  for *i=1,2,...N* **do**
    $\tilde{q}(s_i, a_i) = r_i - c_i + \gamma \tilde{Q}^{h-1}(s'_i, a'_i)$;
    $\mathcal{TS}^h \leftarrow \mathcal{TS}^h \cup \{(s_i, a_i, \tilde{q}(s_i, a_i))\}$;
  end
  Use supervised learning to train a function approximator $\tilde{Q}^h$ on the training set $\mathcal{TS}^h$;
end
Output: $\tilde{Q}^H$

2.3 Control of Biological Networks

Biological pathways that are not working properly may cause diseases since all the steps necessary to perform a task may not be followed correctly. Problems in biological pathways can occur due to mutations happening throughout an individual's life. For example, in colorectal cancer many mutations are responsible for the onset and progression of the disease (PINO; CHUNG, 2010). Another possibility is having a genetic variation inherited from parents that turns the carrier more susceptible to certain diseases. For example, one type of variation in the gene CLEC16A is associated with type-1 diabetes and multiple sclerosis (ZOLEDZIEWSKA et al., 2009). Although a gene turns the carrier more susceptible to a certain disease, it does not necessarily mean it is going to manifest it. For example, who has gene ApoE variant 4 has a higher risk of developing Alzheimer's disease than variants 2 and 3 (GENIN et al., 2011). While having one copy of variant 4 increases the risk of Alzheimer's disease in 5 times, having two copies increases the risk in 20 times (STRITTMATTER, 2012). However, genetics is not the only responsible for this disease and other factors also

contributes to the onset and progression of Alzheimer's disease, such as insulin resistance (ARNOLD et al., 2018; WILLETTE et al., 2015; MONTE, 2009) and inflammation (HEPPNER; RANSOHOFF; BECHER, 2015; HOLMES et al., 2009).

In complex organisms, many factors influence the onset of a disease, which can be modelled by a large network with many entities. In this case, figuring out when and what interventions to apply in order to guide a BT to desirable BOAs is not a trivial task. The random application of interventions can not only take a long time, but also require a large amount of interventions or even lead to unhealthy attractors. When dealing with human health, taking all these points into account is essential, after all, there are the cost of interventions, the risk of adverse reactions and a possible deterioration in patient's health. Therefore, in large and complex networks it is important to have automatic methods that can derive an effective policy from past experiences that is able to quickly shifting a BT from undesirable to desirable BOAs by applying fewer interventions as possible.

When no intervention is done, BNp may apply prediction functions F to compute the next state (see Definition 5). However, when an intervention is applied, a BNp acts in a different way, which can be modelled as a different BNp that dictates state transitions when its respective intervention is applied. As each action, that applies or not an intervention, has an BNp associated so the BT intervention problem can be modelled using multiple BNps based on an action control variable. Using a controlled Boolean networks with perturbations (CBNp) (SHMULEVICH; DOUGHERTY, 2010), when a given action is applied its respective BNp computes the next state of the CBNp. Therefore, depending on whether an intervention was applied or not and what intervention was applied, a BT may go to different states in different BOAs. The BT intervention problem seeks to discover when and what interventions to apply to make the network healthier. This discrete time decision making problem can be modelled as an MDP (SHMULEVICH; DOUGHERTY; ZHANG, 2002b), where an action can intervene or not intervene in a CBNp through the application of therapies. Note that it is possible to apply multiple therapies at the same time and their combined result should be modeled in a single BNp.

Definition 9. *The CBNp control problem is defined by* $\langle B, S, A, T, R, C \rangle$*, where:*

- *B is a set of BNp= $\langle X, F, p \rangle$ (Definition 5), each associated with a possible action that a system can execute. All BNp $\in B$ has the same X and p.*
- *S is a set of states and $S = \mathbf{X}$.*
- *A is a set of possible actions the system can execute and $|A| = |B|$.*
- *T is a transition function $T : (S, A, S) \rightarrow [0, 1]$ that indicates the probability of transitioning to state $s' \in S$ from $s \in S$ after executing action $a \in A$.*
- *R is a function that returns the reward $R : (S, S) \rightarrow \mathbb{R}$ of being at state $s \in S$, and transitioning to a next state $s' \in S$.*
- *C is a function that returns the cost $C : (A) \rightarrow \mathbb{R}$ of applying action $a \in A$.*

Given Definitions 9 and 5, S is a set of states composed of BNp states $\mathbf{X}$; A is the action set composed of a non intervention and one or more intervention actions; R and C are problem dependent and should be given by a specialist according to the desirability of each situation. And T can be computed using the prediction function $\mathbf{F}$ of each BNp and Equation 2.1.

For example, consider a BT composed of three entities, three BNp defined by Tables 1, 2 and 3, and $p = 0.1$, then $S = \{000, ..., 111\}$ and $A = \{a_1 =$ non intervention, $a_2 =$ intervention A, $a_3 =$ intervention B$\}$. By applying Equation 2.1 to F of BNp^1, it results in the transition diagram of Fig. 7. While Figure 7a shows the transition diagram in the absence of perturbations, Figure 7b shows all possible transitions, where all states are reachable from all others due to perturbations. From a given state, all others are reachable by a perturbation in a single entity ($p(1-p)^2$=0.081), two entities ($p^2(1-p)$=0.009) or all three entities (p^3=0.001). Therefore, in the absence of perturbations, BNp goes to the state indicated by prediction functions most of the time ($1-(1-p)^3$=0.729).

The resulting transition T for all actions is defined by Table 4, which is represented by Figure 8 for the case of absence of perturbations. We do not show a diagram for the perturbation case because it would be too polluted to visualize due to the great amount of arrows.

Regarding reward and cost functions, they should be given by a specialist based on the intervention problem being solved. When a specialist is not available, it is possible

Table 1: Example of a BNp with 3 entities for action a_1.

State	x_1	x_2	x_3	$f^{(1)}$	$f^{(2)}$	$f^{(3)}$
0	0	0	0	0	0	0
1	0	0	1	1	0	0
2	0	1	0	1	0	1
3	0	1	1	0	0	0
4	1	0	0	0	1	1
5	1	0	1	1	1	1
6	1	1	0	1	0	1
7	1	1	1	1	1	1

Table 2: Example of a BNp with 3 entities for action a_2.

State	x_1	x_2	x_3	$f^{(1)}$	$f^{(2)}$	$f^{(3)}$
0	0	0	0	0	0	0
1	0	0	1	0	0	0
2	0	1	0	1	0	1
3	0	1	1	1	0	1
4	1	0	0	0	1	1
5	1	0	1	0	1	1
6	1	1	0	1	0	1
7	1	1	1	1	0	1

to apply generic functions to solve the CBNp control problem:

$$R(s') = \begin{cases} \kappa & \text{if } s' \text{ is in a desirable BOA} \\ 0 & \text{if } s' \text{ is in an undesirable BOA} \end{cases}, \tag{2.11}$$

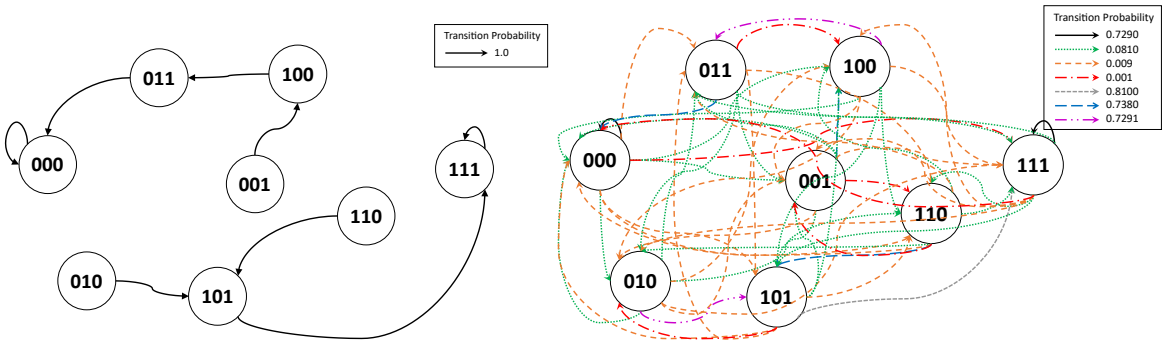
where κ is the reward for going to a desirable state. And for the cost function:

$$C(a) = \begin{cases} \psi & \text{if } a \text{ is an intervention action} \\ 0 & \text{if } a \text{ a non intervention action} \end{cases}, \tag{2.12}$$

where ψ is the cost for applying an intervention. For example, using $\kappa = 10$ and $\psi = 1$

Table 3: Example of a BNp with 3 entities for action a_3.

State	x_1	x_2	x_3	$f^{(1)}$	$f^{(2)}$	$f^{(3)}$
0	0	0	0	1	0	0
1	0	0	1	1	0	0
2	0	1	0	0	0	0
3	0	1	1	0	0	0
4	1	0	0	1	1	1
5	1	0	1	1	1	1
6	1	1	0	1	1	1
7	1	1	1	1	1	1



(a) Transitions by prediction functions (absence of perturbations).

(b) Transitions by prediction functions and $p = 0.1$.

Figure 7: State transition diagram of the BNp defined by Table 1.

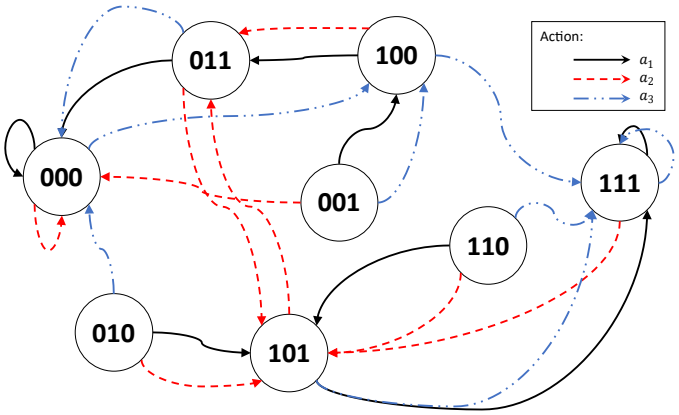


Figure 8: Transitions by prediction functions (absence of perturbations) with 3 actions.

results in Figure 9 for the absence of perturbations case.

2.4 Partial Observability

When specialists seek to build a complete and accurate model using biological knowledge, the more complex the biological function, the more difficult the modeling. For example, the human senescence network available at (FURBER, 2019) is a work in progress since 2000 that seeks to compile all elements responsible for the human aging process. It has around a hundred entities ranging from DNA to whole body physiology, from cellular activity to human pathology and even diet, sleep, fasting and meditation. It is updated when there is new evidences on the literature and this BT shows how hard it is to build an accurate model by hand. Therefore, it should be expected that some entities of a complex biological function are not going to be observed by a control system, which needs to learn an intervention policy by partially observing the state signal (BRYCE; VERDICCHIO; KIM, 2010).

Table 4: Transition probabilities of the example.

Action a_1

State	0	1	2	3	4	5	6	7
0	0.729	0.081	0.081	0.009	0.081	0.009	0.009	0.001
1	0.081	0.009	0.009	0.001	0.729	0.081	0.081	0.009
2	0.009	0.081	0.001	0.009	0.081	0.729	0.009	0.081
3	0.729	0.081	0.081	0.009	0.081	0.009	0.009	0.001
4	0.009	0.081	0.081	0.729	0.001	0.009	0.009	0.081
5	0.001	0.009	0.009	0.081	0.009	0.081	0.081	0.729
6	0.009	0.081	0.001	0.009	0.081	0.729	0.009	0.081
7	0.001	0.009	0.009	0.081	0.009	0.081	0.081	0.729

Action a_2

State	0	1	2	3	4	5	6	7
0	0.729	0.081	0.081	0.009	0.081	0.009	0.009	0.001
1	0.729	0.081	0.081	0.009	0.081	0.009	0.009	0.001
2	0.009	0.081	0.001	0.009	0.081	0.729	0.009	0.081
3	0.009	0.081	0.001	0.009	0.081	0.729	0.009	0.081
4	0.009	0.081	0.081	0.729	0.001	0.009	0.009	0.081
5	0.009	0.081	0.081	0.729	0.001	0.009	0.009	0.081
6	0.009	0.081	0.001	0.009	0.081	0.729	0.009	0.081
7	0.009	0.081	0.001	0.009	0.081	0.729	0.009	0.081

Action a_3

State	0	1	2	3	4	5	6	7
0	0.081	0.009	0.009	0.001	0.729	0.081	0.081	0.009
1	0.081	0.009	0.009	0.001	0.729	0.081	0.081	0.009
2	0.729	0.081	0.081	0.009	0.081	0.009	0.009	0.001
3	0.729	0.081	0.081	0.009	0.081	0.009	0.009	0.001
4	0.001	0.009	0.009	0.081	0.009	0.081	0.081	0.729
5	0.001	0.009	0.009	0.081	0.009	0.081	0.081	0.729
6	0.001	0.009	0.009	0.081	0.009	0.081	0.081	0.729
7	0.001	0.009	0.009	0.081	0.009	0.081	0.081	0.729

Depending on the biological task being analyzed, it may require interactions among many entities and some of them may not be measured due to limitations such as costs. These entities are left out of the entity's activity data set and the model's inference methods or experience-based control methods do not observe their values. Other cause of partial observability is the selection of entities to reduce the high number of entities due to limitations of space, computational power or number of experiences. For example, a BT with only 20 entities has 2^{20} possible states and a T function with $2^{40} \approx 10^{12}$ elements for each action. As the number of states grows exponentially with the number of entities, the user may select the most important ones to reduce the state space. For experience-based methods, the number of experiences is limited by time,

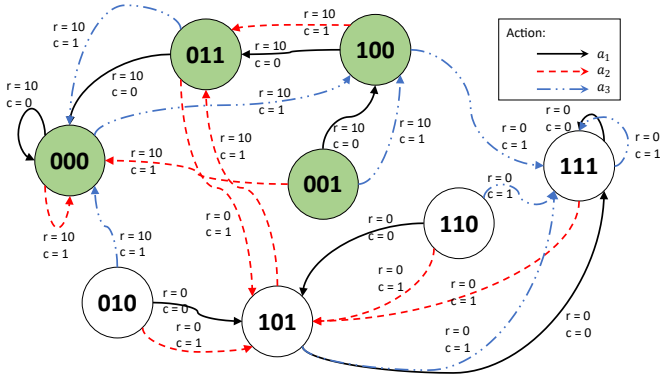


Figure 9: Example MDP with 3 entities and 3 actions in the absence of perturbations. Highlighted states are in a desirable BOA.

cost and difficulty to conduct laboratory experiments. For all these motives, the number of entities may be reduced by specialists using biological knowledge and/or automatic methods (TAN; ALHAJJ; POLAT, 2010b; VERDICCHIO; KIM, 2011).

In this case, the control system observes only n of g entities, where $1 \leq n \leq g$, so the system only perceives $O = \langle x_{i1}, ..., x_{in} \rangle$ instead of the full set of entities $S = \langle x_1, ..., x_g \rangle$. In other words, a CBNp in a given state generates an observation composed of n entities, with $n \ll g$, which is then perceived by the control system. For convenience purposes, we represent observation $\langle x_{i1}, ..., x_{in} \rangle$ as $x_{i1}...x_{in}$. By this partial state signal, a system does not know the true value of the current state and consequently it does not know in which BOA the network is in. For example, Figure 10 shows the transition diagram of Figure 8 when only the last two entities are observed by a control system. Note that some states generate the same observation, and some of these observations can be in a desirable BOA (highlighted states) or in an undesirable BOA (not highlighted states). For example, when a control system observes $o = 11$, it does not know which BOA this observation is in.

Given $o \in O$, the batch of experiences $\mathcal{F} = \{(o, a, r, c, o', a')\}$ is now composed of observations instead of states and the previous formulation of FQI-Sarsa (Alg. 1) can be changed to replace states for observations. However, a state in a desired BOA could generate the same observation as a state in an undesirable BOA, and a control system does not know how to differentiate them. This means that the deterministic policy computed by Eq. 2.6 indicates the same action for states that generate the same observation, even when they require different actions for an optimal performance. For example, in Figure 10, observation 11 can be a state in a desirable pathway (high-

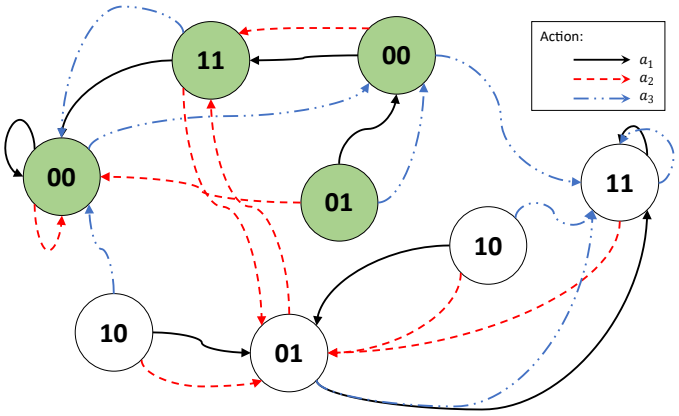


Figure 10: Transitions by prediction functions (absence of perturbations) of Tables 1, 2 and 3 when the first entity is not observed. Highlighted observations belong to a desirable BOA. An observation can belong to any BOA and a system cannot differentiate them.

lighted) or an undesirable attractor (not highlighted). While the former requires the application of a_1 or a_3 to keep following the pathway, the latter requires a_2 to exit the undesirable attractor. If only non interventions are applied in observation 11, the network does not exit the undesirable attractor. At the same time, if only a_2 is applied in observation 11, there is the disruption of a biological pathway (highlighted observation goes for a not highlighted one).

One way to indicate the possibility of applying different actions to the same observation is with the use of a stochastic policy $\pi : S \times A \Rightarrow [0, 1]$ that maps probabilistically actions to apply in each observation (JAAKKOLA; SINGH; JORDAN, 1995; SIRIN; POLAT; ALHAJJ, 2017):

$$\pi(a|o) = \frac{\tilde{Q}(o,a)}{\sum\limits_{a^{\#} \in A} \tilde{Q}(o,a^{\#})}, \tag{2.13}$$

where $\tilde{Q}$ is an approximation function of the Q function (see Algorithm 1). We must ensure that $\sum_{a \in A} \pi(a|o) = 1$ and $0 \leq \pi(a|o) \leq 1, \forall a \in A$. Thus, Equation 2.13 computes a probability distribution over actions proportional to Q-values and allows to probabilistically apply the best action for an observation. In this thesis, we refer to Equation 2.13 as a conventional stochastic policy. Other possibility is to apply the Boltzmann policy:

$$\pi(a|o) = \frac{e^{\tilde{Q}(o,a)/\tau}}{\sum\limits_{a' \in A} e^{\tilde{Q}(o,a')/\tau}}, \tag{2.14}$$

where $\tau > 0$ is a temperature parameter. A large τ makes the probabilities to be equiprobable, since $\frac{\tilde{Q}(o,a)}{\tau} \approx 0 \rightarrow e^{\tilde{Q}(o,a)/\tau \approx 0 \rightarrow} \approx 1$. A small τ greatly increases the differ-

ence between probabilities that differ in the value estimates, so $\tau \approx 0$ derives a policy almost deterministic (SUTTON; BARTO, 2018). Equation 2.14 computes a probability distribution proportional to the exponentials of the Q-Values.

2.5 Chapter Summary

The modeling of biological system is an important tool to elucidate scientists how an organism behaves in different situations, such as in good health, under distress and using different types of interventions. One way to model a BT is using a BNp that is a logical model describing generic and essential relations among entities. In a BNp, at each time step, each entity i has a value x_i indicating whether it is active ($x_i = 1$) or inhibited ($x_i = 0$). The ordered array composed of all entity values, known as state, indicates the activity level of all entities at a given time. During a biological task, BNp follows a chain of states that represents all the steps necessary to accomplish it. Then, when BNp finishes the task, it cycles a sequence of states known as attractor, that may represent phenotypes and cell types. Since perturbations can occur at any time, the BNp state may change for a different one that represents the starting state of a different path towards an attractor. A BOA of a particular attractor is this set of states that, once any of these states are reached by a BNp, the BNp eventually reaches that particular attractor in the absence of disturbances and interventions. Therefore, any state of a desirable BOA drives the BNp to a desirable attractor.

Organisms are open systems that receive nutrients, stimuli and other influences from its surroundings. Hence, everything an organism eats, breathes and interacts with can change entity activity in a certain way. Moreover, even when an organism does not eat, the entity's activity can change. Intermittent fasting has been shown to improve brain function, increase lifespan, induce autophagy and much more in animals (LONGO; MATTSON, 2014; ANTON et al., 2018). Intermittent fasting is a term used to describe eating patterns in which an organism does not consume or consume very few calories for at least 12 hours on a recurring basis. Although autophagy happens regularly within certain organisms, fasting can accelerate this process in which dysfunctional cell components are disassembled to build new and functional ones. At the same time, having more functional cell components has the potential to improve many

biological functions inside an organism.

Because a complex biological task has many entities, interactions and interconnected pathways, an accurate model is very difficult to infer or build. When accuracy is not guaranteed, the intervention policy derived from this model may have a poor performance in a real-world biological system. If an accurate model is not available, automatic methods can support scientists, since they can compute an intervention policy from experience samples and avoid model usage. Experience-based methods can compute an intervention policy from experiences obtained from interactions with a BT or from specialists. When it is possible to interact directly with a BT to collect experiences, methods like the RL algorithm Q-learning can be applied to compute a policy. Otherwise, it is possible to derive a policy from a batch of experience samples collected in advance that represents interactions with the BT. In this case, BRL methods are more appropriate, such as FQI and FQI-Sarsa.

It should be expected that a control system does not observe all entities that participate in a biological task, since they can be difficult to measure or their relation to the task are unknown to scientists. Therefore, taking into account partial observability turns the BT intervention problem more realistic and at the same time more challenging. Control methods should consider partial observability in their solution to correctly solve this intervention problem, otherwise they will only be applicable to specific cases.

Our proposed solution, called BOAConfe, is a framework that seeks to quickly shift partially observable biological system from undesirable to desirable BOAs. Since model accuracy in partially observable biological system is difficult to guarantee, BOAConfe applies BRL methods to derive an intervention policy from experiences given in advance. Although different solutions were proposed to solve the intervention problem, we show in the next chapter that none of them sought to shift between BOAs while dealing with partial observability and using experience-based methods.

3 RELATED WORK

Basically, there are two main strategies for intervening in BTs: structural intervention and external control. The former aims at making minimal and permanent changes on the structure of how entities interact (SHMULEVICH; DOUGHERTY; ZHANG, 2002a; BOUAYNAYA; SHTERENBERG; SCHONFELD, 2011; HU et al., 2016), e.g., intervene in the network to remove a particular inhibiting influence (remove a negative edge from the BT model). One of the main advantages of structural intervention is the application of a single intervention that changes the network dynamics into a more desirable one. The latter applies external stimulus such as lights and drugs that affect interactions (DATTA et al., 2003; PAL; DATTA; DOUGHERTY, 2006; IMANI; BRAGA-NETO, 2018a), so it does not change the interaction rules among entities. In contrast to structural intervention, external control may require repeated application of stimuli until a goal is reached or for maintaining a desirable behavior. We focus our efforts on external control, since structural intervention requires an accurate BT model.

In this chapter, we present and discuss the most relevant and similar work to our framework. In Section 3.1 we present model-based proposals and in Section 3.2 experience-based ones.

3.1 Model-based

The main focus of the external BT control problem is on model-based techniques. They offer several advantages, such as a model that can be analyzed and approved by specialists, simulations with the model that can be performed using different configurations and the computed intervention policy can be tested on the model to verify its performance. When the available model is accurate, an intervention policy that performs well on intervening in the model is likely to perform well on a real BT.

Datta et al. (2003) firstly solved the BT intervention problem using a dynamic programming technique in a fully observable environment. Since they considered finite horizon and a state-avoidance approach, their goal was to terminate the last step in a desirable state and to apply as fewer interventions as possible. Then, the authors

extended their previous work to consider partial observability (DATTA et al., 2004). Their proposed method expands a tree for all possible combinations of successive state-observation pairs after applying all possible actions at each time step. Since both methods explore all possibilities and choose the best one, they become unfeasible for networks with few dozen entities. After both proposals, several studies solved this control problem by applying different techniques, such as: partially observable Markov decision processes (ERDOGDU; POLAT; ALHAJJ, 2012; ERDOGDU; POLAT; ALHAJJ, 2017; IMANI; BRAGA-NETO, 2018b), factored Markov decision process (TAN; ALHAJJ; POLAT, 2010a; TISOVEC et al., 2015), linear programming (YOUSEFI; DOUGHERTY, 2013; KOBAYASHI; HIRAISHI, 2017), probabilistic model checker (HIRAISHI; KOBAYASHI, 2012; WANG et al., 2018).

Partial observability brings more complexity into this control problem, turning it more computational intensive. Imani and Braga-Neto (2018a) proposed to overcome this situation using experience-based methods together with a BT model. Based on an initial belief consisting of a probability distribution over states that indicates the probability that each possible state is the current state of a BT, their method applies this belief as state signal in a RL technique. After applying an action, the BT model updates the belief based on the action applied and new observation. Imani and Braga-Neto (2018a) showed that their method is more efficient than conventional partially observable control methods, however it still requires a BT model.

Regarding BOA-avoidance proposals, Choudhary et al. (2006) used dynamic programming to reach a desirable BOA over a finite number of steps in a fully observable BT. This time, the goal is to terminate in a state of a desirable BOA while applying as fewer interventions as possible. However, like in (DATTA et al., 2003), their proposed method is computational unfeasible for medium or large BTs (QIU et al., 2014). For dealing with larger networks, Hayashida et al. (2008), Akutsu et al. (2012) and Qiu et al. (2014) proposed methods to find a policy that maximizes a score indicating the closeness of a given state to a desirable attractor. While Hayashida et al. (2008) proposed to solve this problem using heuristic methods, Akutsu et al. (2012) and Qiu et al. (2014) applied integer programming. Although their methods are efficient and can be applied to medium sized BT, it requires a function providing a score of closeness from each entity to an attractor, which may not be simple to obtain.

Similarly, Taou, Corne and Lones (2018) proposed to use evolutionary algorithms to make a fully observable BT reach and stay in a desirable attractor. Evolutionary algorithms are heuristic-based approaches that optimize a given task by imitating some aspects of natural evolution. Since the authors considered an infinite horizon, making a BT reach and stay in an attractor could disrupt the biological pathway. For example, when the BT represented in transition diagram of Figure 8 starts a task, it goes to state 001. The best way to reach the goal of being at attractor 000 is to perform intervention 1, so state 001 goes directly to state 000. This intervention disrupts the pathway because the BT does not pass by states 100 and 011 before reaching attractor 000.

Bryce, Verdicchio and Kim (2010) proposed to apply the planning technique AO* to shift from a specific attractor to desirable BOAs in a partially observable BT over a finite number of steps. Like in (DATTA et al., 2004), their proposed method expands a tree with possible plans, and the difference between the two methods lies in how they choose to expand plans. The former expands all possible plans and uses dynamic programming to select the one that maximizes the performance index. The latter expands the plan that has the best expected value, therefore pruning plans to avoid exploring non optimal options, which it more computational efficient. However, when considering perturbations, AO* expands all possible options as in (DATTA et al., 2004) and becomes an unfeasible method for more than a few entities.

3.2 Experience-based

Most model-based methods focus on optimal stochastic control of a BT, which is a NP-hard problem (AKUTSU et al., 2007). In general, it is well established that optimal stochastic control problems suffer from the curse of dimensionality and are limited by the size of the state space (BERTSEKAS, 1995). For example, in a CBNp with 20 entities, there are 2^{20} possible states and a T function with $2^{40} \approx 10^{12}$ elements for each action. Although much can be done to reduce the space and computational requirements, model-based approaches still need an accurate model of the environment to compute a high quality intervention policy.

For dealing with the mentioned issues, Faryabi, Datta and Dougherty (2007) pro-

posed to apply the RL algorithm Q-Learning (WATKINS; DAYAN, 1992) in a fully observable family of BNps. The idea is to make Q-Learning interact with a simulator and incrementally learn an intervention policy capable of avoiding undesirable states. The simulator generates transition samples at each time step from the BNp models, an approach that requires a model of the environment, but not the complete T-function. Since Q-Learning is an experience-based method, it can also be used without a model, but instead of interacting with a simulator it would require an environment to interact with directly. However, Q-learning needs a high number of experiences to compute a high quality policy due to the exploration overhead, which would make this option impracticable for most real-world BTs.

Methods that use deep RL were proposed in (PETERSEN et al., 2019), (ENGELHARDT, 2019), (PAPAGIANNIS; MOSCHOYIANNIS, 2019) and (PAPAGIANNIS; MOSCHOYIANNIS, 2020) to reach a desirable attractor during a finite horizon. Deep RL allows to learn abstract representations of high-dimensional input data with multiple levels of representation from the first to higher levels of abstractions (LECUN; BENGIO; HINTON, 2015). For example, the method proposed in (PETERSEN et al., 2019) learned how to control an inflammation model simulator with a state space composed of a discrete 101×101 grid and 21 real valued state variables, from which they observed the real valued variables. Although (PETERSEN et al., 2019) and (ENGELHARDT, 2019) proposals consider partial observability, they do not address the issues raised from it. In other words, their frameworks work as if the BT is fully observable, hence instead of partially observable we consider their proposals as fully observable. Furthermore, they require an environment to interact with for too many time steps like in (FARYABI; DATTA; DOUGHERTY, 2007), which is usually not a feasible requirement in many biological experiments (CAMACHO et al., 2018).

Sootla et al. (2013) proposed a batch RL (BRL) method to solve the periodic reference tracking problem of a fully observable BT, in which they used continuous values to specify protein concentration. The idea is to find a policy that provokes a periodic sequence of values in certain entities. For example, in a 6 entity network, the protein concentration of entity 1 should follow a sinusoid with period 4, while the concentration of entity 2 a sinusoid with period 8. Their proposed method extends FQI to consider a priori knowledge about the reference trajectory and its periodicity in the state signal,

and the reward function is designed to use reference trajectory information by providing higher rewards when the distance to the reference is small. Although this problem could be extended to consider that a biological pathway is a reference trajectory, the distance between each state and a biological pathway is very difficult to compute without a BT model. Instead of a reference trajectory, Sootla et al. (2014) proposed a growing batch FQI to make a fully observable BT reach and keep a desired protein concentration, which could be considered as keeping a BT in a desirable attractor. However, since Sootla et al. (2014) considered an infinite horizon, keeping a BT in an attractor would prevent the execution of any biological tasks, like in (TAOU; CORNE; LONES, 2018). Hence, we do not consider this method as a BOA-avoidance solution.

Sirin, Polat and Alhajj (2013) proposed a framework to avoid undesirable states using BRL methods. Their framework considers full observability and is composed of 5 phases: binarization of continuous time-series entity activity data sets, assemble of experiences, feature selection, application of the BRL method FQI and computation of a policy. From a data set composed of continuous entity activity level, the first phase binarize them into 0 and 1. The authors assume that this data set was collected by measuring entity activity levels when no interventions were applied, which is the majority of available data set. An experience set composed of only non intervention samples cannot be used to generate a policy for intervention actions, so the second phase transforms some non-intervention actions in intervention actions. We explain this process in appendix A and propose a new experience assemble method. Then, they apply FQI with least-squares regression on this experience set and compute a deterministic policy. Later, Naderi and Mozayani (2019) extended this framework to improve FQI with least-squares regression using Gaussian features and FQI-Sarsa instead of FQI. Since they considered an experience in the form (s, a, r, s), they choose a' based on a greedy strategy, known as ϵ-greedy strategy: with probability ϵ action a' is randomly selected and with probability $1 - \epsilon$ action $a' = arg\ max_{a \in A} Q(s, a)$.

Sirin, Polat and Alhajj (2017) extended their previous framework (SIRIN; POLAT; ALHAJJ, 2013) to solve the partially observable BT intervention problem. Their framework differs in two steps from their previous proposal: a new BRL method and the application of a conventional stochastic policy. The new BRL method Least-Squares Fitted TD(λ) Iteration (LSFTDI) is a batch version of the algorithm TD(λ), which uses time se-

ries experiences to update Q-values. Instead of using Equation 2.9, it estimates the expected reward of the current observation-action pair based on subsequent rewards and an estimate of the expected reward of subsequent observation-action pairs. In other words, the estimation of the expected reward for the current observation-action pair is the discounted sum of rewards received from the current until the last observation-action pair being considered plus the estimation of the expected reward of this last pair. At the same time, the last observation-action pair being considered goes from the next pair until the last experience, which turns this method much more computational intensive than FQI.

3.3 Discussion

Model-based approaches have the benefit of having a model of a BT that was approved by specialists and is capable of describing its dynamics. While these methods can plan the best policy based on how the model is going to react in the future, an optimal policy requires simulation and analysis of all possible policies to select the best one, making them unfeasible for large models. Although it is possible to apply methods like heuristic or linear programming to solve high computational complexity issues, they still need a model of a BT, which can be difficult to obtain. As the complexity of a biological function increases, it also increases the number of entities involved and the number of connections between them. Consequently, the complexity of obtaining an accurate model and deriving an intervention policy from it greatly increases, mainly for partially observable BTs.

Experience-based methods can be applied to overcome this situation, as they use experiences to learn a control policy instead of doing simulation using the model. They usually do not use a model of the environment and rely on the quality of the gathered experiences, which can be collected by interacting directly with a BT or a batch collected in advance. While collecting experiences by interactions allows to explore all possibilities and learn the best one, it is usually impractical to keep an environment (possible an alive organism) to do many experiments on. On the other hand, BRL methods that use experiences gathered in advance rely on the quality of this batch to compute a high quality policy.

Although most methods seek to avoid undesirable states, it is more important to consider attractors and biological pathways to correctly treat diseases. Attractors concentrate BT dynamics and represent phenotypes, so a BT is expected to be cycling attractors most of the time. When it is outside an attractor, a BT is usually executing biological functions, which means it is following a biological pathway that has a sequence of steps to perform. However, state-avoidance approaches completely ignore this biological portion of the problem and may avoid states even when they are part of a biological function. Besides, transforming a state-avoidance to a BOA-avoidance problem may have benefits when there is at least one attractor that is a desirable state. In this case, the goal is to lead a BT towards their BOAs, where it eventually reaches the attractor with a desirable state and cycles it for a long period of time.

While methods solving a state-avoidance problem could be designed to avoid BOAs, they may not produce a good result. They try to maximize a performance index, which results in policies that seek to reach states having a higher probability of staying inside a desirable BOA. In other words, when perturbations occur, a state that is less likely to leave the desirable BOA is preferred to a state that is likely to leave it. This situation is even more severe in a partially observable BT using stochastic policies. A conventional stochastic policy may indicate interventions for observations in desirable BOAs that make a BT go to undesirable ones. Hence, an observation that is less likely to leave the desirable BOA is preferred to an observation that is likely to leave it because of perturbations or interventions.

In summary, existing proposals have some issues in regard to:

- **Model usage:** many approaches require an accurate BT model to compute a policy, which is difficult to obtain;
- **Data-efficiency:** some experience-based methods require too many experiences to learn a good policy;
- **Full observability:** some approaches consider that it is possible to observe all factors that are involved in a biological function, which is difficult to guarantee;
- **State avoidance:** most approaches avoid undesirable states, even though it is more important to avoid undesirable BOAs.

In Table 5 we show how proposals satisfy the mentioned requirements for solving the BT control problem.

The proposals of Sootla et al. (2014) and Sirin, Polat and Alhajj (2017) fulfils most of the mentioned specifications to solve this control problem. Therefore, we test both methods in the experimental evaluation (Chapter 5) and compare their results to our framework.

Table 5: Summary of the most relevant and similar work to our proposal.

Publications	Model/ Experience	Data Efficiency	Observability	State/BOA Avoidance	Horizon
(DATTA et al., 2003; TISOVEC et al., 2015; YOUSEFI; DOUGHERTY, 2013; KOBAYASHI; HIRAISHI, 2017)	Model	-	Full	State	Finite
(PAL; DATTA; DOUGHERTY, 2006; TAN; AL-HAJJ; POLAT, 2010a; YOUSEFI; DOUGHERTY, 2013; HIRAISHI; KOBAYASHI, 2012; WANG et al., 2018)	Model	-	Full	State	Infinite
(DATTA et al., 2004)	Model	-	Partial	State	Finite
(ERDOGDU; POLAT; ALHAJJ, 2012; ERDOGDU; POLAT; ALHAJJ, 2017; IMANI; BRAGA-NETO, 2018a)	Model	-	Partial	State	Infinite
(CHOUDHARY et al., 2006; HAYASHIDA et al., 2008; AKUTSU et al., 2012; QIU et al., 2014)	Model	-	Full	BOA	Finite
(TAOU; CORNE; LONES, 2018)	Model	-	Full	BOA	Infinite
(BRYCE; VERDICCHIO; KIM, 2010)	Model	-	Partial	BOA	Infinite
(FARYABI; DATTA; DOUGHERTY, 2007)	Experience	No	Full	State	Infinite
(PETERSEN et al., 2019; ENGELHARDT, 2019; PAPAGIANNIS; MOSCHOYIANNIS, 2019; PAPAGIANNIS; MOSCHOYIANNIS, 2020)	Experience	No	Full	BOA	Finite
(SOOTLA et al., 2013; SOOTLA et al., 2014)	Experience	Yes	Full	BOA	Infinite
(SIRIN; POLAT; ALHAJJ, 2013; NADERI; MOZAYANI, 2019)	Experience	Yes	Full	State	Infinite
(SIRIN; POLAT; ALHAJJ, 2017)	Experience	Yes	Partial	State	Infinite
(NISHIDA; BIANCHI; COSTA, 2020)	Experience	Yes	Partial	BOA	Infinite

4 PROPOSAL

In this chapter, we propose a framework for the biological network (BT) intervention problem. Our framework, called **B**asin **O**f **A**ttraction **Con**trol **F**ramework through **E**xperiences (BOAConFE), focus on shifting partially observable BTs from undesirable to desirable basins of attraction (BOA) through batch reinforcement learning (BRL) techniques. In addition, our framework is generic and can also be used for state-avoidance and/or full observability. According to the best of our knowledge, our proposal is the first one to solve BOA-avoidance problems in partially observable BT using BRL methods.

Figure 11 presents BOAConFE. It has 6 phases:

1. Preprocessing of a time-series of observations;
2. Computation of the probability P($o \in \mathcal{D}$) of an observation o being in desirable BOAs $\mathcal{D}$;
3. Preprocessing of an experience data set;
4. Reward assignment;
5. BRL method application and
6. Policy computation.

This dissertation focus on phases 2, 4, 5 and 6, since preprocessing phases have many solutions on the literature. It is noteworthy that it is possible to skip phases depending on the problem and available information.

Entity activity values (observations) may refer to continuous measurements for more than a dozen thousand entities. For example, gene activity data set is usually composed of RNA messenger concentration for any RNA messenger available in a cell sample. RNA messenger is one type of product of an active gene, hence a high concentration of RNA messenger indicates that its respective gene is active. When a data set has continuous measurements, first and third phases of our framework apply methods such as described in (HOPFENSITZ et al., 2012; ZHOU; WANG; DOUGHERTY,

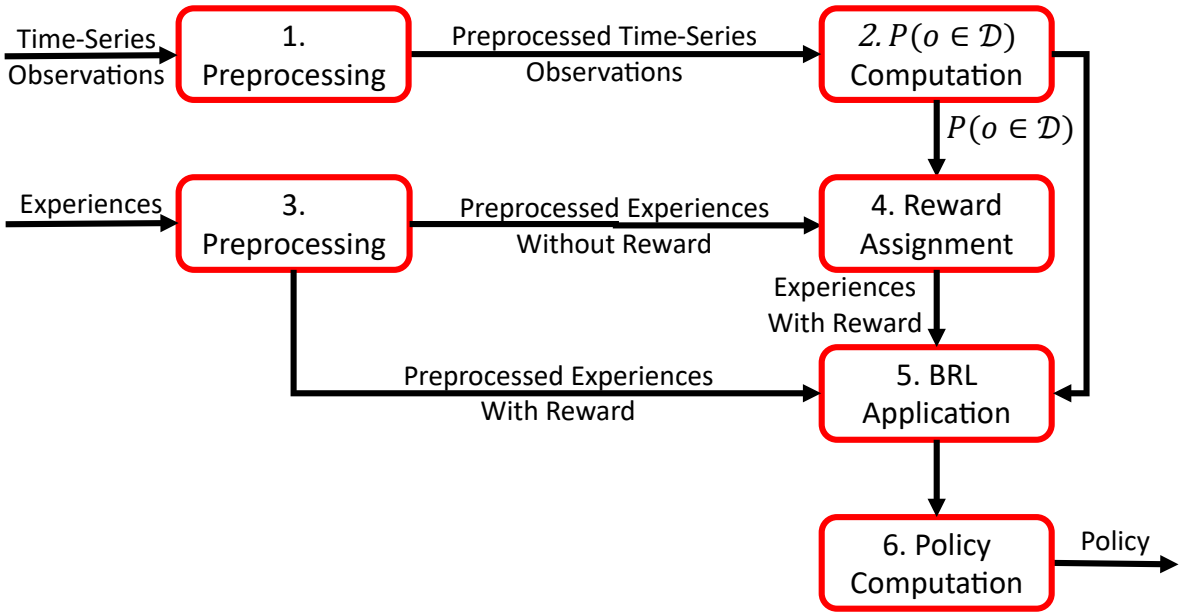


Figure 11: Proposed framework BOAConFE composed of 6 phases. This dissertation focus on phases 2, 4, 5 and 6.

2003; MAHANTA et al., 2012) to binarize all values. Then, data set is further preprocessed by passing through entity selection, where the most important entities of the biological task being analyzed are selected and unrelated ones are discarded. A specialist and/or automatic methods such as in (IVANOV; PAL; DOUGHERTY, 2007; TAN; ALHAJJ; POLAT, 2010a; VERDICCHIO; KIM, 2011) can perform this task.

Phase 2 computes $P(o \in \mathcal{D})$ that BOAConFE can use in phases 4, 5 or 6 to deal with partial observability in BOA-avoidance problems. Phase 4 assigns rewards to experiences when they are not available due to partial observability. Phase 5 applies a BRL method capable of solving the BT intervention problem at hand. Finally, the last phase computes an intervention policy: deterministic or stochastic.

Next sections present how we propose to solve phase 2 (Section 4.1), 4 (Section 4.2), 5 (Section 4.3) and 6 (Section 4.4) of BOAConFE. We also proposed a phase of experience assembly (NISHIDA; COSTA; BIANCHI, 2017) that is available in appendix A. However, since it is not generic and cannot be applied to all BRL methods, it was not included in our framework.

4.1 Phase 2: $P(o \in \mathcal{D})$ computation

When it is not possible to observe all state variables, a BT is in a certain state that generates an observation perceived by a control system instead of the state signal. In this case, from g state variables a control system perceives only n, $n \leq g$, so 2^{g-n} different states generate the same observation; consequently, a single observation represents 2^{g-n} different states. Given an observation, a control system does not know what state generated this observation, nor the actual BOA the BT is in. Sirin, Polat and Alhajj (2017) proposed to use observations as states in their BRL method and apply a stochastic policy to deal with partial observability. Although their proposal obtained reasonable results in Chapter 5 (Experiments), having more knowledge regarding BOAs allows to make informed decisions and derive a higher quality policy.

We propose to use $P(o \in \mathcal{D})$ defining the probability of an observation o is representing a state in desirable BOAs $\mathcal{D}$ (NISHIDA; BIANCHI; COSTA, 2018) in order to deal with partial observability in BOA-avoidance problems. Note that a BOA is a list of states generating their respective observations and $\mathcal{D}$ is composed of all states in any desirable BOA. In other words, $P(o \in \mathcal{D})$ indicates the probability of o is representing a state in a desirable BOA. It is noteworthy that $P(o \in \mathcal{D})$ depends on BT dynamics and it does not represent the fraction of states generating o that are in $\mathcal{D}$. For example, in Figure 10, states 011 and 111 generate observation 11. While 011 is a transient state in $\mathcal{D}$, 111 is an undesirable attractor. Attractors absorb the CBNp dynamics since it cycles an attractor most of the time, so this CBNp is usually in state 000 or in state 111, and rarely in the other states. When observing 11, it is probably representing state 111, since state 011 is rarely visited.

In phase 2, BOAConFE computes $P(o \in \mathcal{D})$ from a data set composed of time-series of observations collected without applying interventions. This requirement ensures that a BT in a given BOA reaches its attractor in the absence of perturbations. If observations were collected applying interventions, they could make the BT move to a different BOA and our method would not be able to identify this change. Being an attractor $\sigma = (o_1, o_2, ..., o_k)$ a sequence of k observations, Ω a set of σ and $\Omega_{\mathcal{D}}$ a set of desirable σ, our method requires $\Omega_{\mathcal{D}}$ and the probability $P(\sigma \in \Omega_{\mathcal{D}})$ as input, both provided by a specialist. Because different attractors can generate the same sequence

of observations, $P(\sigma \in \Omega_{\mathcal{D}})$ indicates the probability of an attractor σ being desirable. It should be noticed that perturbations can also change the current BOA and our method would not identify when it happens. In this case, this change is considered part of the stochastic environment in our framework.

Algorithm 2 explains our method that was proposed in (NISHIDA; BIANCHI; COSTA, 2020). For illustrating how it works, Fig. 12 shows an example with ten observations. It starts reading observations until it finds a sequence that repeats consecutively for at least $n_r = 3$ times. Since observation 00 repeats 3 times, it is marked as an attractor cycle $\sigma_1 = (00)$. $\sigma_1 \in \Omega_{\mathcal{D}}$ and it has probability $P(\sigma_1 \in \Omega_{\mathcal{D}}) = 1$ of being a desirable attractor, so Algorithm 2 assigns $P_o(o \in \mathcal{D}) = P(\sigma_1 \in \Omega_{\mathcal{D}}) = 1$ to all observations in the sequence towards σ_1, where $P_o(o \in \mathcal{D})$ is the probability of a particular observation being in $\mathcal{D}$. In other words, all past observations drove the BT towards σ_1, so they are in σ_1 BOA and also receives $P_o(o \in \mathcal{D}) = P(\sigma_1 \in \Omega_{\mathcal{D}})$ because a BOA is desirable if and only if its attractor is desirable. The observation after σ_1 is a new path towards a different attractor, hence our algorithm keeps following it until finding a new sequence that repeats for at least n_r times. It defines $\sigma_2 = (11)$ as an attractor and since $\sigma_2 \notin \Omega_{\mathcal{D}}$, it is an undesirable attractor. Consequently, our framework assigns $P_o(o \in \mathcal{D}) = 0$ for all observations in σ_2 and in the path towards it. After reading all observations, the average $P(o \in \mathcal{D})$ is computed for all $o \in O$:

$$\begin{aligned} \phi &= \{o_i | o_i = o, i = 1, ..., N\} \\ P(o \in \mathcal{D}) &= \frac{\left(\sum_{o_j \in \phi} P(o_j \in \mathcal{D})\right) + 0.5}{|\phi| + 1}, \end{aligned} \tag{4.1}$$

0.5 is added as a new term when computing the average, which accounts for uncertainty in the samples. Hence, when a particular o has many samples, adding 0.5 is negligible in the average computation. On the other hand, when there are few samples, it attributes a probability to cases that were not seen, but have a chance of happening. In the example, $P(11 \in \mathcal{D}) = (1.0 + 0.0 + 0.0 + 0.0 + 0.5)/5 = 0.3$.

$P(o \in \mathcal{D})$ can be easily extended to indicate the probability of an observation being in each BOA. For example, in the case of three BOAs, BOA_1, BOA_2 and BOA_3, each attractor σ has probabilities: $P(\sigma \in \Omega_{BOA_1})$, $P(\sigma \in \Omega_{BOA_2})$ and $P(\sigma \in \Omega_{BOA_3})$. Then, instead of computing $P(o \in \mathcal{D})$ our method computes $P(o \in BOA_1)$, $P(o \in BOA_2)$ and

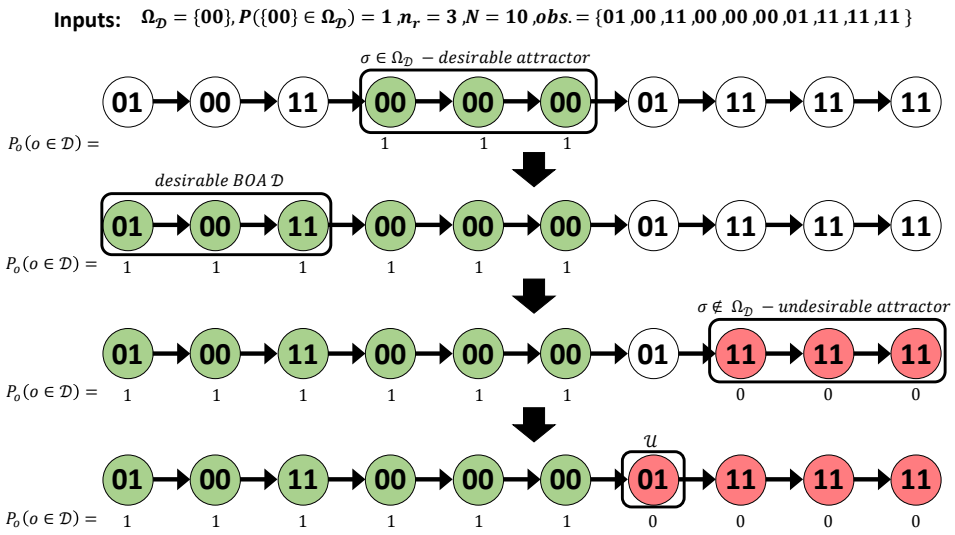


Figure 12: Example of how Algorithm 2 works. Each line represents a phase of the algorithm.

Algorithm 2: $P(o \in \mathcal{D})$ Calculation

Input : A set of desirable attractors $\Omega_{\mathcal{D}}$, Probability $P(\sigma \in \Omega_{\mathcal{D}})$, number of repetitions n_r, number of observations N, data set with time-series of observations $(o_1 ... o_N)$

1 $i = 1, j = 1;$
2 **while** $i \leq N$ **do**
3 **if** $o_i \in \sigma$ *that cycles consecutively for* n_r *times in* $(o_j, .., o_i)$ **then**
4 **while** $i \leq N$ *and* $o_i \in \sigma$ ▷ *find last* $o_i \in \sigma$ *in* $(o_i ... o_N)$ *that continues to cycle* σ **do**
5 $i = i + 1$
6 **end**
7 **if** $\sigma \in \Omega_{\mathcal{D}}$ **then**
8 $P_o(o_s \in \mathcal{D}) = P(\sigma \in \Omega_{\mathcal{D}})\ \forall o_s \in \{o_j, .., o_{i-1}\}$
9 **else**
10 $P_o(o_s \in \mathcal{D}) = 0\ \forall o_s \in \{o_j, .., o_{i-1}\};$
11 **end**
12 $j = i$
13 **else**
14 $i = i + 1;$
15 **end**
16 **end**
17 Calculate $P(o \in \mathcal{D}) \forall o \in O$ using Equation 4.1;

Output: $P(o \in \mathcal{D})$

$P(o \in BOA_3)$ in lines 7-11 and 17. However, when more variables ($P(o \in BOA)$ of several BOAs) must be learned from data, more observations are required to compute a reasonable $P(o \in BOA)$ and to compute a high quality policy.

When observability is poor and few state variables are observed, our method may wrongly identify attractors. In this case, many states generate the same observation and this could cause a sequence of observations repeating for n_r times that is not an attractor cycle. When the perturbation probability p is small, one way to overcome this situation is by increasing n_r. With a small p, CBNp has a higher probability of cycling the current attractor for a longer period of time and using a larger n_r our method would still identify attractor cycles, while reducing the number of false positives. For example, a sequence of states (1010, 1101, 1100, 1000, 0000, 0000, 0000) generates a sequence of observations (10, 11, 11, 10, 00, 00, 00) when the first two entities are observed. If $n_r = 2$, our method identifies (11) and (00) as attractors, since they repeat consecutively for at least two times. However, (11) it is not an attractor and was wrongly classified, since the pair (11, 11) comes from different states (1101, 1100). For $n_r = 3$, our method only identifies (00) as an attractor, which is a correct classification in this example.

In our proposed framework, $P(o \in \mathcal{D})$ can be used in phases 4, 5 and/or 6, for computing experience rewards, as a component of our proposed BRL method, and for computing our proposed stochastic policy, respectively. Otherwise, when none of the mentioned phases is going to use $P(o \in \mathcal{D})$, BOAConFE does not need to compute it. Next, we explain how to assign rewards to experiences when they are not available in advance.

4.2 Phase 4: Reward and cost assignment

Reward functions R and cost functions C should indicate the desirability of states or actions for a given objective (SUTTON; BARTO, 2018). When they represent a desirable situation a learner should receive higher rewards and lower costs. Therefore, the design of R and C is problem dependent and should be provided by a specialist based on prior knowledge. When a specialist is not available, R and C can be defined based on generic functions.

In works such as (PAL; DATTA; DOUGHERTY, 2006) and (SIRIN; POLAT; ALHAJJ, 2013), the authors proposed reward functions for general cases of a state-avoidance problem. While Pal, Datta and Dougherty (2006) proposed to give a positive reward

when the next state is desirable, Sirin, Polat and Alhajj (2013) proposed to reward when the current state is desirable. At the same time, both proposed to penalize each application of an intervention, since the objective is to reduce the number the interventions. These generic functions can be applied for BOA-avoidance problems, where a positive reward is given for visiting desirable BOAs instead of desirable states. However, we argue that there are more suitable functions for BOA-avoidance problems.

A BOA is composed of connected observations that once a CBNp reaches any observation in a BOA, it eventually reaches its attractor in the absence of perturbations and interventions. In a BOA-avoidance problem, the main goal is to reach and stay in a desirable BOA. At the same time, reaching and staying in an undesirable BOA is the worst-case scenario that should be highly penalized. Leaving an undesirable for a desirable BOA should be rewarded, but not as much as being and staying in a desirable BOA. And leaving a desirable for an undesirable BOA should be penalized, but less than the worst-case scenario. Therefore, current and next observations can provide a better indication of the desirability of an experience. We propose the following reward function (NISHIDA; COSTA; BIANCHI, 2017):

$$R(o,o') = \begin{cases} r_{\mathcal{DD}} & \text{if } o,o' \in \mathcal{D} \\ r_{\mathcal{UD}} & \text{if } o \in \mathcal{U}, o' \in \mathcal{D} \\ r_{\mathcal{DU}} & \text{if } o \in \mathcal{D}, o' \in \mathcal{U} \\ r_{\mathcal{UU}} & \text{if } o,o' \in \mathcal{U} \end{cases}, \tag{4.2}$$

where o and o' are the current and next observations, $\mathcal{U}$ is a set of states in undesirable BOAs that generates observations, $\mathcal{D}$ is a set of states in desirable BOAs that generates observations, $r_{\mathcal{DD}}$ is the reward for being and staying in desirable BOAs $\mathcal{D}$, $r_{\mathcal{UD}}$ is the reward for leaving $\mathcal{U}$ and reaching $\mathcal{D}$, $r_{\mathcal{DU}}$ the reward for leaving $\mathcal{D}$ and reaching $\mathcal{U}$ and $r_{\mathcal{UU}}$ for being and staying in $\mathcal{U}$. It should be emphasized that a reward can be a negative signal.

For the cost function $C(a) \rightarrow \mathbb{R}$, we use Equation 2.12., where ψ should be a number that balances positive and negative effects of intervention actions. This means that ψ should be small enough to encourage the application of interventions when a BT is in undesirable BOAs, while it should be high enough to discourage interventions when it is in desirable BOAs. Therefore, ψ should be able to avoid the application

of interventions when they do not bring a relevant gain and not compensate for side-effects and treatment costs.

Although our reward function is particularly useful for BOA-avoidance, it was originally proposed for solving state-avoidance problems in (NISHIDA; COSTA; BIANCHI, 2017). In this case, the goal is to keep a BT in a sequence of desirable states, while avoiding an undesirable sequence. Thus, Equation 4.2 can be used for both BOA and state avoidance problems.

For state-avoidance, when the goal is to keep an observable entity inhibited or activated, it is possible to automatically assign rewards. However, for a BOA-avoidance problem, Equations 2.11 and 4.2 can only be used when a specialist is able to define when an observation is either in $\mathcal{D}$ or $\mathcal{U}$. This laborious task is specially difficult in a partially observable environment, mainly when there are several thousands of experiences to classify.

In BOAConFE, the information that is able to tell the probability of an observation being in $\mathcal{D}$ is $P(o \in \mathcal{D})$. We changed Equation 4.2 to consider $P(o \in \mathcal{D})$ in a partially observable BOA-avoidance problem:

$$R(o,o') = r_{\mathcal{D}\mathcal{D}}P(o \in \mathcal{D})P(o' \in \mathcal{D}) + r_{\mathcal{U}\mathcal{D}}P(o \in \mathcal{U})P(o' \in \mathcal{D}) + r_{\mathcal{D}\mathcal{U}}P(o \in \mathcal{D})P(o' \in \mathcal{U}) + r_{\mathcal{U}\mathcal{U}}P(o \in \mathcal{U})P(o' \in \mathcal{U}). \tag{4.3}$$

This formulation works as the average reward a given sequence of observation-next observation is expected to receive. Although Equation 4.3 is not an optimal reward function since it does not differentiate when an observation is truly in $\mathcal{D}$ or not, it is capable of automatically assigning rewards when no prior knowledge is available. Note that Equation 2.12 remains the same because it depends only on performed actions that are always known.

After assigning rewards, the experience set $\mathcal{F} = \{(o, a, r, o', a')\}$ is complete and has all elements required by a BRL algorithm. In the next section, we explain and propose a new BRL algorithm to solve the BT intervention problem.

4.3 Phase 5: BRL application

The BRL application phase applies any BRL method for state or BOA avoidance problems. While for state-avoidance it is possible to apply FQI-Sarsa (Alg. 1) or LS-FTDI (SIRIN; POLAT; ALHAJJ, 2017), there are no BRL algorithms for BOA-avoidance problems that do not require BT information. As we mentioned before, state-avoidance methods can be designed to avoid BOAs, in which the reward function should consider observations in $\mathcal{D}$ as desirable. Although it would provide good results, exploiting BOA information could generate higher quality policies.

We propose a novel BRL algorithm called **m**ultiple **S**teps **B**asin **O**f **A**ttraction **F**itted **Q**-**I**teration (mSBOAFQI), whose focus is on producing higher quality policies in partially observable BTs and BOA-avoidance problems. For solving BOA-avoidance, mSBOAFQI explores how BOAs are structured in disconnected graphs, and for tackling poor observability, it uses information about a sequence of observation-action pairs available in an experience. Since both concepts are distinct from each other, we first explain how to explore BOA structure in an algorithm called BOAFQI (Subsection 4.3.1) and then we extend it to incorporate the other concept (Subsection 4.3.2).

4.3.1 Basin of Attraction Fitted Q-iteration

States in a given BOA may have some patterns like a certain entity inhibited most of the time or higher rewards for a given action, which could be explored when the system knows the current BOA. However, with partial observability this knowledge becomes uncertain, as the same observation may represent states in different BOAs. Although a system does not know what BOA a given observation is in, $P(o \in \mathcal{D})$ estimates the probability of an observation being in $\mathcal{D}$ and in $\mathcal{U}$.

If the current observation o has a high $P(o \in \mathcal{D})$, then there is a higher probability of finding desirable BOA patterns. For each experience, our proposed method **B**asin of **A**ttraction **F**itted **Q**-**I**teration (BOAFQI) (NISHIDA; COSTA; BIANCHI, 2018) throws a biased coin to decide whether o is classified as in $\mathcal{D}$ or not. Therefore, according to the coin, with a probability $P(o \in \mathcal{D})$ the experience is added to the desirable training set $\mathcal{F}_{\mathcal{D}}$, otherwise it is added to $\mathcal{F}_{\mathcal{U}}$ with probability $1 - P(o \in \mathcal{D})$. In average, experiences

are going to be probabilistically classified in the correct set.

Being $\tilde{Q}_{\mathcal{U}}(o,a)$ the expected value of an observation-action pair for undesirable BOAs $\mathcal{U}$ and $\tilde{Q}_{\mathcal{D}}(o,a)$ the expected value for desirable BOAs $\mathcal{D}$, the update of $\tilde{q}(o,a)$ for an experience i changes from Equation (2.7) to:

$$\tilde{q}(o_i,a_i) = r_i - c_i + \gamma(\tilde{Q}^{h-1}_{\mathcal{D}}(o'_i,a'_i) \times P(o_i \in \mathcal{D}) + \tilde{Q}^{h-1}_{\mathcal{U}}(o'_i,a'_i) \times (1 - P(o_i \in \mathcal{D}))). \tag{4.4}$$

This equation updates $\tilde{q}(o,a)$ using the weighted average of the expected value of desirable and undesirable BOAs. Then, $\{(o,a,\tilde{q}(o,a))\}$ is added to the training set $\mathcal{TS}_{\mathcal{D}}$ if this particular experience belongs to $\mathcal{F}_{\mathcal{D}}$, otherwise it is added to $\mathcal{TS}_{\mathcal{U}}$. After updating all experiences of both $\mathcal{F}_{\mathcal{D}}$ and $\mathcal{F}_{\mathcal{U}}$, BOAFQI trains a supervised learning method on $\mathcal{TS}_{\mathcal{D}}$ and $\mathcal{TS}_{\mathcal{U}}$, resulting on $\tilde{Q}_{\mathcal{D}}$ and $\tilde{Q}_{\mathcal{U}}$, respectively.

Algorithm 3 shows BOAFQI. It requires as input variables the discount parameter γ that the user defines based on the importance of past experiences over new ones, the maximum number of iterations H, in which there is the update and train of $\tilde{Q}$, the number of experiences N, the experience set $\mathcal{F}$ previously collected and the probability $P(o \in \mathcal{D})$. In lines 12 and 16 there is the update of $\tilde{q}(o,a)$ for each experience in $\mathcal{TS}_{\mathcal{D}}$ and $\mathcal{TS}_{\mathcal{U}}$, and after updating all experiences, lines 20 and 21 train the approximation functions $\tilde{Q}_{\mathcal{D}}$ and $\tilde{Q}_{\mathcal{U}}$ using a supervised learning method. After repeating this process for H times, BOAFQI returns $\tilde{Q}_{\mathcal{D}}$ and $\tilde{Q}_{\mathcal{U}}$, then BOAConFE uses them in the last phase to compute an intervention policy.

BOAFQI incorporates the idea of how BOAs are structured in disconnected directed graphs to find useful patterns. This concept is our proposed solution to tackle a BOA-avoidance problem and can be used for fully or partially observable BTs. However, when observability is poor, BOAFQI does not produce good results because it cannot differentiate similar observations generated by different states. Next, we propose how to extend BOAFQI to tackle poor observability by providing more information about the current observation.

4.3.2 Multiple steps basin of attraction fitted Q-iteration

In partially observable BTs, an observation represents 2^{g-n} states, where g is the number of entities and n the number of the observed ones, with $n \leq g$. As n decreases,

Algorithm 3: Basin of Attraction Fitted Q-Iteration (BOAFQI)

Input : Discount factor γ, max number of iterations H, number of experiences N, experience tuples $\mathscr{F} = \{(o_i, a_i, r_i, c_i, o'_i, a'_i) \mid i = 1, ...N\}$, probability $P(o \in \mathcal{D})$

$\tilde{Q}_{\mathcal{D}}(o, a) = 0,\ \tilde{Q}_{\mathcal{U}}(o, a) = 0\ \forall\ o \in O, a \in A$;
for $i = 1, 2, ..., N$ *in* $\mathscr{F}$ **do**
  if *Biased coin falls to* $\mathcal{D}$ *with probability* $P(o_i \in \mathcal{D})$ **then**
    $\mathscr{F}_{\mathcal{D}} \leftarrow (o_i, a_i, r_i, c_i, o'_i, a'_i)$;
  else
    $\mathscr{F}_{\mathcal{U}} \leftarrow (o_i, a_i, r_i, c_i, o'_i, a'_i)$;
  end
end
for $h = 1, 2, ..., H$ **do**
  $\mathcal{TS}_{\mathcal{D}} = \emptyset$;
  $\mathcal{TS}_{\mathcal{U}} = \emptyset$;
  foreach $(o_i, a_i, r_i, c_i, o'_i, a'_i) \in \mathscr{F}_{\mathcal{D}}$ **do**
    $\tilde{q}(o_i, a_i) = r_i - c_i + \gamma(\tilde{Q}_{\mathcal{D}}(o'_i, a'_i) \times (P(o_i \in \mathcal{D})) + \tilde{Q}_{\mathcal{U}}(o'_i, a'_i) \times (1 - P(o_i \in \mathcal{D}))$ (Equation 4.4);
    $\mathcal{TS}_{\mathcal{D}} \leftarrow \mathcal{TS}_{\mathcal{D}} \cup \{(o_i, a_i, \tilde{q}(o_i, a_i))\}$;
  end
  foreach $(o_i, a_i, r_i, c_i, o'_i, a'_i) \in \mathscr{F}_{\mathcal{U}}$ **do**
    $\tilde{q}(o_i, a_i) = r_i - c_i + \gamma(\tilde{Q}_{\mathcal{D}}(o'_i, a'_i) \times P(o_i \in \mathcal{D}) + \tilde{Q}_{\mathcal{U}}(o'_i, a'_i) \times (1 - P(o_i \in \mathcal{D})))$ (Equation 4.4);
    $\mathcal{TS}_{\mathcal{U}} \leftarrow \mathcal{TS}_{\mathcal{U}} \cup \{(o_i, a_i, \tilde{q}(o_i, a_i))\}$;
  end
  Use supervised learning to train a approximation function for $\tilde{Q}_{\mathcal{D}}(o, a)$ on the training set $\mathcal{TS}_{\mathcal{D}}$;
  Use supervised learning to train a function approximator $\tilde{Q}_{\mathcal{U}}(o, a)$ on the training set $\mathcal{TS}_{\mathcal{U}}$;
end
Output: $\tilde{Q}_{\mathcal{D}}, \tilde{Q}_{\mathcal{U}}$

the number of states an observation represents gets larger and the best action for a particular state may be executed in a smaller number of times. Therefore, for computing a high quality policy it is important to be able to differentiate what state an observation is representing and apply the best action more often.

In the **absence** of perturbations a CBNp works as a BN and deterministically follows a chain of states. This means that from a given state, a CBNp only reaches a particular state after passing through the **same** sequence of states. For example, in Fig. 8, when only non interventions are applied, state 011 reaches 000 after passing through states 100 and 011. When we add partial observability, there is a sequence of observations that might be different for some states that generate the same obser-

vation. For example, in Fig. 10 observation 11 is in $\mathcal{D}$ when the previous step is (00, non intervention), otherwise when it is (10, non intervention), (01, non intervention), (11, intervention) or (10, intervention), observation 11 is not in $\mathcal{D}$. As observability gets worse, the more steps in this sequence, the greater the probability of having different trajectories for reaching an observation generated by different states.

We propose the algorithm **m**ultiple **S**teps **B**asin **O**f **A**ttraction **F**itted **Q**-**I**teration (mSBOAFQI), which is a combination of BOAFQI (NISHIDA; COSTA; BIANCHI, 2018) and the usage of multiple steps (NISHIDA; BIANCHI; COSTA, 2020). Being a step an observation-action pair, mSBOAFQI considers m steps in an experience $\mathscr{F} = \left\{(o_i^{t-m+2}, a_i^{t-m+2}, o_i^{t-m+3}, a_i^{t-m+3}, .., o_i^{t-1}, a_i^{t-1}, o^t, a_i^t, r_i^t, c_i^t, o_i^{t+1}, a_1^{t+1})\right\}$. As it is possible to note, an experience has a sequence of $m-2$ observation-action pairs that precede the current pair (o^t, a_i^t) and the next one (o^{t+1}, a_i^{t+1}), making a total of m pairs.

Since the number of variables in an experience increased, mSBOAFQI requires a larger data set containing more experiences to provide good results. In order to mitigate this problem, instead of using $m-2$ precedent observations mSBOAFQI uses their discretized $P(o \in \mathcal{D})$. Thus, as $P(o \in \mathcal{D})$ is discretized in b bins, mSBOAFQI deals with b bins instead of 2^n observations that exponentially increase with the number of observed entities n. For example, $b = 2$ results in two groups of observations, 1 for probably in $\mathcal{U}$ and 2 for probably in $\mathcal{D}$. In addition to decreasing the required sample size to get a good result, discretization also fasten the learning process (GARCIA et al., 2013).

Being $Z = O^{m-1}$ the observation space of an observation and discretized $P(o \in \mathcal{D})$ of $m-2$ precedent observations, $z \in Z$, $U = A^{m-1}$ the action space of an action and $m-2$ precedent actions, $u \in U$; the experience set becomes $\mathscr{F} = \{z, u, r, c, z', u')\}$ and $\tilde{Q}(z, u)$ represents the estimation of the expected value of an observation-action pair and its $m-2$ precedent pairs. For example, Fig. 13 shows how an experience with $m = 3$ should be transformed in mSBOAFQI experience. First, the current observation 11 and its precedent observation 01, the current action a_1 and its precedent action a_2, the next observation 00 and its precedent observation 11, and the next action a_1 and its precedent action a_1 are separated to compose z, u, z' and u', respectively. All precedent observations are exchanged by their $P(o \in \mathcal{D})$ and then they are discretized in b bins. Because in this example $b = 2$, this process binarize them into 1 (undesirable BOA) and

2 (desirable BOA). Finally, this process mounts the mSBOAFQI experience by joining all parts in a tuple.

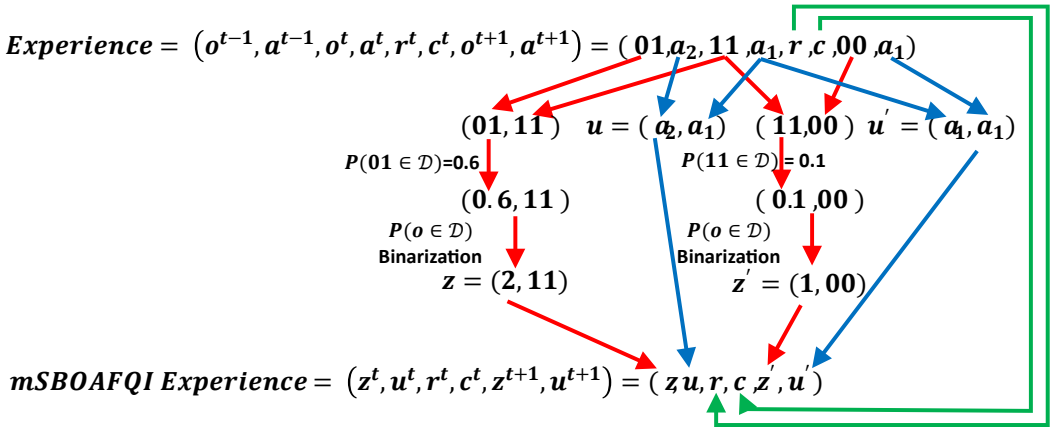


Figure 13: Example of how to transform an experience in mSBOAFQI experience.

Alg. 4 shows mSBOAFQI. It requires input parameters: the discount factor γ, the maximum number of iterations H, number of steps m in an experience, batch of experiences $\mathscr{F}$ collected in advance, number of experiences N, and the probability $P(o \in \mathcal{D})$. The parameter b should be given by a specialist based on the expected observability and size of the experience set. The time step t in an experience is added for generalization purposes, one experience is not necessarily time related to the other. In line 4, mSBOAFQI discretizes $P(o \in \mathcal{D})$ in b bins. Between lines 5 and 17 the input experience is converted to mSBOAFQI experience, which is then probabilistically classified in $\mathcal{D}$ or $\mathcal{U}$. In lines 21-24, mSBOAFQI estimates the expected value of an observation-action pair z-u for experiences classified as desirable. In line 22, each experience has its $\tilde{q}(z_i, u_i)$ updated, which represents the updated expected value of an observation-action and its $m-2$ precedent pairs of a particular desirable experience i. Then mSBOAFQI adds $(z_i, u_i, \tilde{q}(z_i, u_i))$ into the desirable training set $\mathcal{TS}_{\mathcal{D}}$. The same process happens for experiences classified as undesirable in lines 25-28. In lines 29 and 30, a supervised learning method trains approximation functions $\tilde{Q}_{\mathcal{D}}(z, u)$ and $\tilde{Q}_{\mathcal{U}}(z, u)$ using their respective training sets. msBOAFQI can use any supervised learning method and in this work we use random forest regression due to its automatic feature selection, low bias, efficiency and flexibility (BREIMAN, 2001). After repeating this process for H times, mSBOAFQI outputs $\tilde{Q}_{\mathcal{D}}$ and $\tilde{Q}_{\mathcal{U}}$ and BOAConFE can compute an intervention policy in the last stage.

It is noteworthy that when $m = 2$ mSBOAFQI is equal to BOAFQI. As m grows,

Algorithm 4: Multiple Steps Basin of Attraction Fitted Q-Iteration (mSBOAFQI)

Input : discount factor γ, max number of iterations H, number of steps m, number of experiences N, experience tuples $\{(o_i^{t-m+2}, a_i^{t-m+2}, .., o_i^t, a_i^t, r_i, c_i, o_i^{t+1}, a_i^{t+1}) \mid i = 1, ...N\}$, probability $P(o \in \mathcal{D})$, number of bins b.

$\tilde{Q}_{\mathcal{D}}(z,u) = 0, \tilde{Q}_{\mathcal{U}}(z,u) = 0 \; \forall \; z \in Z, u \in U$;
$\mathcal{F}_{\mathcal{D}} = \emptyset$;
$\mathcal{F}_{\mathcal{U}} = \emptyset$;
Discretize $P(o \in \mathcal{D})$ in b bins;
for *i = 1, 2, ..., N* **do**
  $z_i \leftarrow (P(o_i^{t-m+2} \in \mathcal{D}), ..., P(o_i^{t-1} \in \mathcal{D}), o_i^t)$;
  $u_i \leftarrow (a_i^{t-m+2}, ..., a_i^{t-1}, a_i^t)$;
  $z'_i \leftarrow (P(o_i^{t-m+3} \in \mathcal{D}), ..., P(o_i^t \in \mathcal{D}), o_i^{t+1})$;
  $u'_i \leftarrow (a_i^{t-m+3}, ..., a_i^t, a_i^{t+1})$;
  $r_i \leftarrow r_i^t$;
  $c_i \leftarrow c_i^t$;
  if *Biased coin falls to* $\mathcal{D}$ *with probability* $P(o_i^t \in \mathcal{D})$ **then**
    $\mathcal{F}_{\mathcal{D}} \leftarrow \mathcal{F}_{\mathcal{D}} \cup (z_i, u_i, r_i, c_i, z'_i, u'_i)$;
  else
    $\mathcal{F}_{\mathcal{U}} \leftarrow \mathcal{F}_{\mathcal{U}} \cup (z_i, u_i, r_i, c_i, z'_i, u'_i)$;
  end
end
for h *= 1, 2, ..., H* **do**
  $\mathcal{TS}_{\mathcal{D}} = \emptyset$;
  $\mathcal{TS}_{\mathcal{U}} = \emptyset$;
  foreach $(z_i, u_i, r_i, c_i, z'_i, u'_i) \in \mathcal{F}_{\mathcal{D}}$ **do**
    $\tilde{q}(z_i, u_i) = r_i - c_i + \gamma(\tilde{Q}_{\mathcal{D}}(z'_i, u'_i) \times P(o_i^t \in \mathcal{D}) + \tilde{Q}_{\mathcal{U}}(z'_i, u'_i) \times (1 - P(o_i^t \in \mathcal{D})))$;
    $\mathcal{TS}_{\mathcal{D}} \leftarrow \mathcal{TS}_{\mathcal{D}} \cup \{(z_i, u_i, \tilde{q}(z_i, u_i))\}$;
  end
  foreach $(z_i, a_i, r_i, c_i, z'_i, u'_i) \in \mathcal{F}_{\mathcal{U}}$ **do**
    $\tilde{q}(o_i, a_i) = r_i - c_i + \gamma(\tilde{Q}_{\mathcal{D}}(z'_i, u'_i) \times P(o_i^t \in \mathcal{D}) + \tilde{Q}_{\mathcal{U}}(z'_i, u'_i) \times (1 - P(o_i^t \in \mathcal{D})))$;
    $\mathcal{TS}_{\mathcal{U}} \leftarrow \mathcal{TS}_{\mathcal{U}} \cup \{(z_i, u_i, \tilde{q}(z_i, u_i))\}$;
  end
  Use supervised learning to train a approximation function for $\tilde{Q}_{\mathcal{D}}(z,u)$ on the training set $\mathcal{TS}_{\mathcal{D}}$;
  Use supervised learning to train a approximation function for $\tilde{Q}_{\mathcal{U}}(z,u)$ on the training set $\mathcal{TS}_{\mathcal{U}}$;
end
Output: $\tilde{Q}_{\mathcal{D}}, \tilde{Q}_{\mathcal{U}}$

there are more variables (observation-action pairs) to consider and more experiences are required to compute a good policy. In partially observable CBNp, knowing the current observation is not enough to compute the next one. The probability of going from o to o' depends on which state the CBNp really is in and not on o. For example, in Fig. 14 the probability $T(11, a_1, 00)$ of going from observation 11 to observation 00

after applying action a_1 depends on the current state. While the leftmost observation 11 goes to 00, the rightmost does not and the difference between the two is which state they are representing. The leftmost one is state 011 and the other one is 111, hence knowing the observation alone is not enough to determine the transition probabilities. This means that a partially observable CBNp is not Markovian in regard to observations and mSBOAFQI mitigates this problem by considering a $(m-1)^{th}$ order MDP, in which $P(O' = o'|O = o, ..., O^{2-m} = o^{2-m})$. Although it does not solve raised issues, mSBOAFQI can produce higher quality policies than other methods as we empirically demonstrate in the experimental evaluation (see Chapter 5).

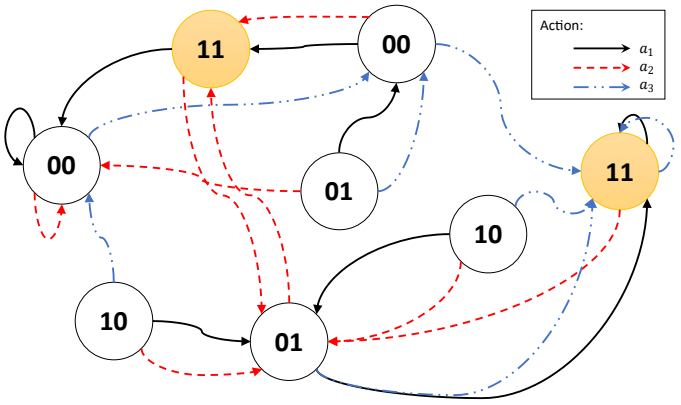


Figure 14: Transition diagram with three genes, where the first one is not observed. Even tough the highlighted observations are the same, the probability of going to the next one depends on which state the particular observation is representing.

4.4 Phase 6: Policy computation

The last stage of BOAConFE computes a policy to intervene in BTs. When Equation 2.6 or 2.13 is used to compute a policy, this stage provides a deterministic or stochastic policy, respectively. For mSBOAFQI, Equation 2.13 changes to:

$$\pi(a|z, a^{t-m+2}, .., a^{t-1}) = \frac{\tilde{Q}_{\mathcal{D}}(z,u) \times P(o^t \in \mathcal{D}) + \tilde{Q}_{\mathcal{U}}(z,u) \times (1 - P(o^t \in \mathcal{D}))}{\sum_{a' \in A} \begin{bmatrix} \tilde{Q}_{\mathcal{D}}(z, (a^{t-m+2}, .., a')) \times P(o^t \in \mathcal{D}) + \\ \tilde{Q}_{\mathcal{U}}(z, (a^{t-m+2}, .., a')) \times (1 - P(o^t \in \mathcal{D})) \end{bmatrix}}, \tag{4.5}$$

where $z = (o^{t-m+2}, ..., o^t) \in Z$ and $u = (a^{t-m+2}, ..., a) \in U$. Note that $P(o^t \in \mathcal{D})$ is based on current observation o^t and we must ensure that $\sum_{a' \in A} \pi(a|z, a^{t-m+2}, .., a^{t-1}) = 1$, which means that $\pi(a|z, a^{t-m+2}, .., a^{t-1})$ indicates the probability of applying action $a \in A$ in o^t given that $m-2$ precedent observation-action pairs already happened.

A stochastic policy returns the probability of applying each action, which in simulations it means rolling a biased dice. However, this approach would raise ethical concerns about how to choose an appropriate therapy in the real world to treat a patient. Although this subject deserves a deeper analysis, we assume that a stochastic policy indicates which action has a better chance of treating a patient, and a physician based on this information and her own knowledge decides an action. In any case, it is essential to design effective policies to support decision making.

In the ideal scenario, when a BT is already in $\mathcal{D}$, interventions are not applied because $\mathcal{D}$ will drive the BT towards its attractor with a desirable phenotype. An intervention in $\mathcal{D}$ means that a patient received extra treatments that could cause side effects and one of this side-effects could be shifting the BT to $\mathcal{U}$. Therefore, it is important to derive a policy applying as fewer interventions in $\mathcal{D}$ as possible. Equations 2.13 and 2.14 can derive stochastic policies that indicate too many interventions for observations in $\mathcal{D}$ because it does not consider BOAs. For example, it is possible to intervene in an observation $o \in \mathcal{D}$ and go to $o' \in \mathcal{D}$, the resulting $\tilde{Q}(o, a)$ could be high since the BT stayed in $\mathcal{D}$. Other possibility is $o \in \mathcal{D}$ going to $o' \in \mathcal{U}$, if o' can easily reach $o'' \in \mathcal{D}$ then $\tilde{Q}(o, a)$ may be high because $\tilde{Q}(o', a')$ is high. An intervention action a with a high $\tilde{Q}(o, a)$ could get a high probability $\pi(o, a)$ of being applied because they could get a high ratio in Equations 2.13, 2.14 or Equation 4.5. In the case of the Boltzmann policy in Equation 2.14, it also depends on the temperature parameter τ because it can turn the policy more equiprobable or more deterministic.

A stochastic policy is a way to overcome partial observability, but there should be a way to turn it less stochastic when a BT is already in $\mathcal{D}$. If the current observation has a high probability of being in $\mathcal{D}$, i.e. a high $P(o \in \mathcal{D})$, a policy should indicate a non intervention action with a higher probability. At the same time, with a small $P(o \in \mathcal{D})$ the computed policy should resemble Equation 2.13 stochastic policy because any action could be a good option for an observation in $\mathcal{U}$.

We propose a new policy called Basin of attraction policy (BOA policy) (NISHIDA; BIANCHI; COSTA, 2018) that modifies the Boltzmann policy (Equation 2.14) to con-

sider $P(o \in \mathcal{D})$:

$$\pi(a^{na}|o) = \frac{e^{\tilde{Q}(o,a^{na})/\tau} \times (1 + P(o \in \mathcal{D}))}{e^{\tilde{Q}(o,a^{na})/\tau} \times (1 + P(o \in \mathcal{D})) + (1 - P(o \in \mathcal{D})) \times \sum_{a' \in A^{in}} e^{\tilde{Q}(o,a')/\tau}}$$
$$\pi(a^{ia}|o) = \frac{e^{\tilde{Q}(o,a^{ia})/\tau} \times (1 - P(o \in \mathcal{D}))}{e^{\tilde{Q}(o,a^{na})/\tau} \times (1 + P(o \in \mathcal{D})) + (1 - P(o \in \mathcal{D}))) \times \sum_{a' \in A^{in}} e^{\tilde{Q}(o,a')/\tau}}, \quad (4.6)$$

where a^{na} is a non intervention action, A^{ia} is a set of intervention actions, $a^{ia} \in A^{ia}$ and τ is a temperature parameter.

Equation 4.6 formulation computes a Boltzmann like probability distribution that is proportional to the exponentials of the Q-Values and $P(o \in \mathcal{D})$. It increases the probability of choosing a non intervention action by $P(o \in \mathcal{D})$, and for an intervention action it decreases the application proportionally to $P(o \in \mathcal{D})$. This means that as uncertainty decreases for observations in $\mathcal{D}$ and $P(o \in \mathcal{D}) \approx 1$, this policy becomes less stochastic. For example, when $P(o \in \mathcal{D}) = 1$ and o is certainly in $\mathcal{D}$, BOA policy indicates only non intervention actions a^{na}. As $P(o \in \mathcal{D})$ decreases, the probability of applying a^{na} decreases until $P(o \in \mathcal{D}) = 0$ and BOA policy becomes equal to the Boltzmann policy. Note that BOA policy can be easily extended to consider multiple steps and $P(o \in BOA)$.

4.5 Chapter summary

Our proposed framework BOAConFE (Figure 11) seeks to provide all the necessary phases for the BT intervention problem. It is composed of six phases: 1) preprocessing of a data set composed of time-series of observations, 2) computation of $P(o \in \mathcal{D})$, 3) preprocessing of experience samples, 4) reward assignment, 5) BRL algorithm application and 6) intervention policy computation. First and third phases, preprocessing, binarize continuous values and select the most important entities for the biological task being analyzed. Our framework focus on the other phases (2, 4, 5 and 6), since preprocessing have many methods on the literature.

Phase 2 computes the probability $P(o \in \mathcal{D})$ of a given observation o being in desirable BOAs $\mathcal{D}$. Since $P(o \in \mathcal{D})$ is related to BOAs, it should only be computed when solving a BOA-avoidance problem and it is going to be used in phases 4, 5 and/or 6.

Algorithm 2 computes $P(o \in \mathcal{D})$ from a time-series of observations collected without applying intervention actions, so only perturbations change the current BOA. Our method identifies a sequence of observations that repeats for at least n_r times in a row as an attractor and all observations leading to it as belonging to its BOA. If the attractor is desirable, its BOA is desirable and our algorithm computes how many times a particular observation drove the BT to desirable attractors.

The process of defining the desirability of each experience is difficult in partially observable BTs. In this case, a specialist may not know if the observations are in $\mathcal{D}$ or $\mathcal{U}$, so she does not know the desirability of each experience. When rewards are not available in the experience set, BOAConFE assigns them in the third phase. Although our method does not know whether a given observation is in $\mathcal{D}$ or $\mathcal{U}$, it can assign a reward that represents the average reward an observation is expected to receive using $P(o \in \mathcal{D})$ of current and next observations. Regarding the cost function, the cost signal can be easily assigned because it only requires the action applied, which is known.

Phase 5 applies a chosen BRL method, which can be any method such as FQI (ERNST; GEURTS; WEHENKEL, 2005), FQI-Sarsa (PEDNAULT; ABE; ZADROZNY, 2002) or LSFTDI (SIRIN; POLAT; ALHAJJ, 2017). For BOA-avoidance problems we recommend to use our proposed method mSBOAFQI (Alg. 4) that uses information about BOAs to improve the learning process. mSBOAFQI explores how BOAs are structured in disconnected directed graphs in order to do a better estimate of the expected reward of an observation-action pair. Additionally, mSBOAFQI tackles poor observability by considering more observation-action pairs in an experience, which defines a trajectory to the current pair. Although it does not solve partial observability, it mitigates observability issues and provides better results.

Finally, the last phase computes any intervention policy, deterministic or stochastic. For partially observable BTs and BOA-avoidance problems, we recommend our proposed BOA policy (Equation 4.6) that can reduce the number of interventions applied in $\mathcal{D}$. BOA policy reduces the probability of applying interventions inversely proportional to $P(o \in \mathcal{D})$, i.e., it reduces the probability of applying an intervention action when a CNBp is going to reach a healthy phenotype without further application of therapies. In this case, applying unnecessary interventions could cause side effects or in a severe case, make a BT exit $\mathcal{D}$.

Although our framework focus on solving BOA-avoidance problems in partially observable BTs, it is generic and can also be used for full observability and/or state-avoidance problems. For full observability, phase 5 could apply BOAFQI instead of mSBOAFQI in BOA-avoidance problems, or FQI in state-avoidance problems. A deterministic policy (Equation 2.6) should be applied in a fully observable BT and a stochastic policy (Equations 2.13 or 2.14) in a partially observable BT state-avoidance problem. Still, we argue and empirically demonstrate in experiments (Chapter 5) that designing a state-avoidance problem as BOA-avoidance has benefits.

Next, we empirically demonstrate that our framework provides better results than Sirin, Polat and Alhajj (2017) and Sootla et al. (2014) proposals and verify each phase performance.

5 EXPERIMENTS

In this chapter we describe the conducted experiments, report their results and discuss overall performance. After executing all experiments, we expect to answer five main questions:

1. How do the results change when we increase the number of steps m in mSBOAFQI, given different degrees of observability (Section 5.2)?
2. How does the accuracy of the optimal and calculated $P(o \in \mathcal{D})$ affect the results (Section 5.3)?
3. Does the reward function affect the quality of a derived intervention policy (Section 5.4)?
4. How can different intervention policies impact results (Section 5.5)?
5. Can experience-based methods provide better results than those based on models in partially observable biological networks (Section 5.6)?
6. Can BOAConFE offer higher quality policies than previous proposals (Section 5.7)?

Question 1 focus on analyzing mSBOAFQI, while questions 2, 3 and 4 on evaluating phases 2 ($P(o \in \mathcal{D})$ computation), 4 (reward assignment) and 6 (policy computation), respectively. Question 5 compares model-based and experience-based methods in partially observable BTs. Lastly, question 6 evaluates how BOAConfe performs in comparison with proposals of Sirin, Polat and Alhajj (2017) and Sootla et al. (2014).

We use our proposed Basin of Attraction (BOA) policy (Equation 4.6) and reward function (Equation 4.3), unless stated otherwise. For the reward and cost functions, we select: $r_{\mathcal{DD}} = 100$, $r_{\mathcal{DU}} = -20$, $r_{\mathcal{UD}} = 20$, $r_{\mathcal{UU}} = -100$ and $\psi = 20$. This formulation highly discourage an action that keeps a biological network (BT) in $\mathcal{U}$, while highly encourage the selection of an action that keeps a BT in $\mathcal{D}$. At the same time, it discourage actions making a BT cycle between $\mathcal{D}$ and $\mathcal{U}$, since an intervention forcing a BT out of $\mathcal{D}$ receives $r_{\mathcal{DU}} - \psi = -40$ and an intervention making it reach $\mathcal{D}$ receives $r_{\mathcal{UD}} - \psi = 0$.

Finally, having $|r_{\mathcal{DU}}| = |r_{\mathcal{UD}}| = \psi$ avoids intervention actions that are not beneficial, while not preventing the application of beneficial ones. Because BOA policy uses the exponential of Q-values, we scalarized all rewards minus cost between -1 and 1, otherwise small differences in Q-values would have a huge impact.

Table 6 specifies all parameters used in the experimental evaluation. As mentioned before, we use random forest regression as supervised learning for algorithms mSBOAFQI, FQI and FQI-Sarsa, while for LSFTDI we use the same regression used by the authors, that is, least-squares regression. For LSFTDI we set λ=0, whose value Sirin, Polat and Alhajj (2017) reported that they got the best results in their experimental evaluation. Unless stated otherwise, we use a mid-range observability of 60% that provides good results.

Table 6: Parameters used in the experimental evaluation.

Parameter	Description
$b = 3$	Number of bins in mSBOAFQI (Algorithm 4)
$p = 0.005$	Perturbation probability
$\gamma = 0.9$	Discount factor
$\tau = 1$	Temperature parameter in BOA policy (Equation 4.6)
$H = 100$	Number of learning iterations
$r_{\mathcal{DD}} = 100$	Reward for going from $\mathcal{D}$ to $\mathcal{D}$.
$r_{\mathcal{DU}} = -20$	Reward for going from $\mathcal{D}$ to $\mathcal{U}$.
$r_{\mathcal{UD}} = 20$	Reward for going from $\mathcal{U}$ to $\mathcal{D}$.
$r_{\mathcal{UU}} = -100$	Reward for going from $\mathcal{U}$ to $\mathcal{U}$.
ψ = 20	Cost of an intervention action.

In this work, we simulate biological systems using CBNps. For all experiments, we use 100 synthetically generated CBNps to test a wide variety of BTs and in question 5 we also use four real based CBNps: melanoma (YOUSEFI; DOUGHERTY, 2013), yeast cell cycle (LI et al., 2004), T-Helper cell differentiation (MENDOZA; XENARIOS, 2006), and human aging (FURBER, 2019). It is noteworthy that frameworks BOAConFE, (SOOTLA et al., 2014) and (SIRIN; POLAT; ALHAJJ, 2017) do not require a model of the biological system to compute a policy. In this work, each CBNps generates a time-series of observations for BOAConFE and an experience set; all frameworks use them to calculate an intervention policy. Each derived policy is applied into this CBNp for 100 time steps to compute three measures: number of steps in desirable BOAs $\mathcal{D}$ (#Steps$_{\mathcal{D}}$), number of interventions applied (#a^{ia}) and interventions applied in

$\mathcal{D}$ per 100 steps in $\mathcal{D}$ ($\#a^{ia}_{\mathcal{D}}$). The goal is to increase $\#\text{Steps}_{\mathcal{D}}$, while decreasing $\#a^{ia}$ and $a^{ia}_{\mathcal{D}}$. All metrics are the average of 1,000 executions for each CBNp and each execution starts in an undesirable attractor arbitrarily chosen.

Regarding parameter p, when it is too large the CBNp cannot execute a biological function because it is perturbed too often. For example, when $p = 1$ CBNp jumps to a completely different state at each time step and the biological pathway is always disrupted. On the other hand, when p is small, CBNp does not leave the attractor and it becames difficult to evaluate a policy. For example, when $p = 0$, a CBNp reaches an attractor and with a policy that only applies a non-intervention action in it, the CBNp stays in the attractor until the end of the execution. In this case, CBNp visited a small portion of the state space so the evaluation is impaired. Therefore, when choosing p there is a trade-of between letting the CBNp follow a biological pathway and facilitating the evaluation of policies.

In order to compare results we used the statistical tests Kruskall-Wallis at 5% of significance and paired Wilcoxon signed rank at 5% of significance for all possible combination of pairs. When both tests indicate that an algorithm provides the best metric in comparison to all other algorithms, than its result is shown in bold.

5.1 Biological Systems

In this section we describe the real based CBNps and the parameters used to synthetically generate them. In both cases we use a perturbation probability $p = 0.005$, hence a 10-entity CBNp suffers a perturbation almost 5% of the time. A larger p would make a CBNp act much more randomly and a smaller p would make it rarely leave attractors, thus hampering the evaluation of all frameworks (SHMULEVICH; DOUGHERTY, 2010). We select entities to be observed using the method proposed by (VERDICCHIO; KIM, 2011) that computes the importance of each entity in defining $\mathcal{D}$ structure.

All CBNps in this section were originally designed with a single BNp that governs state transitions in the absence of interventions. In order to define BNPs for interventions, we apply the concept of control entities that we explain in Appendix A. We select the most important entity x_i as control entity using (VERDICCHIO; KIM, 2011) method,

and define 3 possible actions $A = \{$*non intervention, inhibition of* x_i, *activation of* $x_i\}$, thus, we expect to simulate how two targeted therapies would work when applied to inhibit or activate x_i.

5.1.1 Melanoma network

In a study of malignant melanoma composed of 8,067 genes, it was shown that gene WNT5A is a highly discriminating factor between cell lines with high and low competence for metastasis (BITTNER et al., 2000). This conclusion was later validated by Weeraratna et al. (2002), whose study showed via genetic engineering that an increase in WNT5A protein levels increased the competence of melanoma metastasizing, while an inhibition reduced its competence. Therefore, these studies suggest that keeping gene WNT5A inhibited helps preventing melanoma metastasizing.

In our experiments, we use the CBNp described in (YOUSEFI; DOUGHERTY, 2013), which formulation is shown in Table 7. Since it has 10 of the most important genes for this biological function, it has 1,024 possible states $S = \{0..0, .., 1..1\}$. After applying Table 7 prediction functions in each possible state, the result is the transition diagram shown in Figure 15. As it shows, there are 4 BOAs and only one of them has an attractor with WNT5A active, which can be considered the undesirable BOA $\mathcal{U}$. All BOAs have a mixture of desirable (yellow - WNT5A inhibited) and undesirable (blue - WNT5A active) states, so in this intervention problem it is important to keep the CBNp in the yellow ones.

Table 7: Regulatory functions of a metastatic melanoma CBNp from (YOUSEFI; DOUGHERTY, 2013).

Entity	Node	Observed	Predictor function
WNT5A	x_1	✓	$(x_3 \wedge x_5 \wedge \overline{x_6}) \vee (\overline{x_5} \wedge x_6)$
pirin	x_2	✓	$(\overline{x_1} \wedge \overline{x_3} \wedge x_5) \vee (x_1 \wedge \overline{x_3} \wedge \overline{x_5})$
S100P	x_3	✓	x_7
RET1	x_4	✓	$(\overline{x_1} \wedge x_2 \wedge x_4) \vee (\overline{x_2} \wedge x_4)$
MMP3	x_5	✓	$(x_4 \wedge x_9) \vee (\overline{x_9})$
PHOC	x_6	✓	$(\overline{x_4} \wedge \overline{x_7}) \vee (x_4 \wedge x_7 \wedge x_{10})$
MART1	x_7	✓	x_7
HADHB	x_8		$(x_1 \wedge x_5) \vee (\overline{x_5} \wedge \overline{x_9}) \vee (x_1 \wedge \overline{x_5} \wedge x_9)$
synuclein	x_9		$(\overline{x_1} \wedge \overline{x_7} \wedge \overline{x_{10}}) \vee (x_4 \wedge \overline{x_7} \wedge x_{10}) \vee x_7$
STC2	x_{10}		$\overline{x_3}$

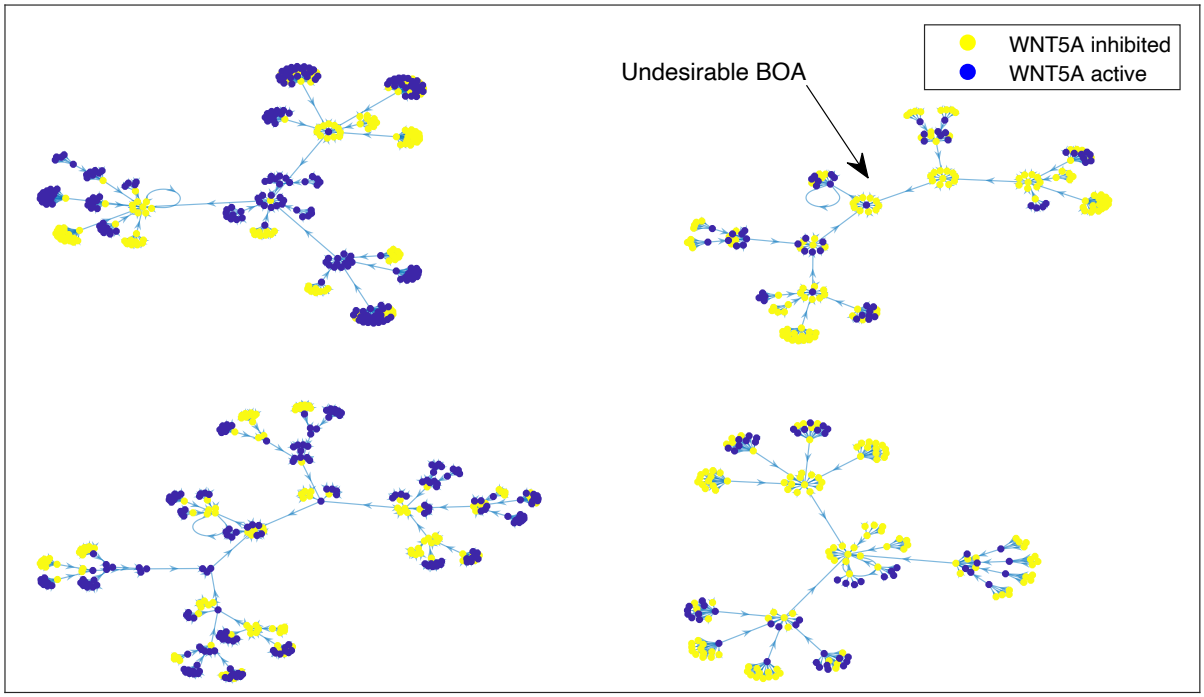


Figure 15: Transition diagram for the melanoma CBNp described in Table 7. It has 4 BOAs and only one of them has an attractor with WNT5A active (undesirable BOA).

In (SIRIN; POLAT; ALHAJJ, 2017) they use 7 observed entities, so we select 7 entities to be observed by applying the method proposed in (VERDICCHIO; KIM, 2011). This method also defined Mart1 as the most influential gene capable of shifting the CBNp between $\mathcal{U}$ and $\mathcal{D}$, so it is defined as control gene. Consequently, we define 3 possible actions $A = \{$*non intervention, Mart1 inhibition, Mart-1 activation*$\}$, where a *non intervention* action can always be applied, *Mart1-inhibition* only when Mart1 is active and *Mart1-activation* only when it is inhibited. Having a BNp for the *non intervention* action, we computed the transition probabilities as we did in Table 4 and applied the concept of control genes (see Appendix A) to compute the transition matrix T for all actions.

After calculating T, we have the MDP model composed of $<S, A, T, R, C>$, note that we defined R and C in Chapter 5. We assume the CBNp is at the undesirable attractor because it cycles it most of the time in a disease and use the MDP model to simulate 10,000 experiences by applying randomly selected actions. This experience set is composed with states, rewards and costs, so we convert states to observations, remove rewards and costs before providing them to all frameworks.

5.1.2 Yeast cell cycle network

The yeast (*Saccharomyces cerevisiae*) cell cycle proposed by Li et al. (2004) models how the most important genes involved in this function interact among themselves. As it was shown in Figure 1, it has 11 key genes and prediction functions described in Table 8 after applying the influence procedure (see Subsection 2.1.1). The transition diagram in Figure 4 shows 7 BOAs, from which the largest one is considered desirable because it has the yeast cell cycle pathway, and all others undesirable.

Table 8: Regulatory functions of a yeast cell cycle proposed by (LI et al., 2004)

Entity	Node	Observed	Predictor function
Cln3	x_1		0
MBF	x_2	✓	$(x_2 \wedge \overline{x_{10}}) \vee (x_1 \wedge \overline{x_{10}}) \vee (x_1 \wedge x_2)$
SBF	x_3	✓	$(x_3 \wedge \overline{x_{10}}) \vee (x_1 \wedge \overline{x_{10}}) \vee (x_1 \wedge x_3)$
Cln1	x_4		x_3
Cdh1	x_5	✓	$(\overline{x_4} \wedge x_7 \wedge \overline{x_8} \wedge \overline{x_{10}}) \vee (\overline{x_4} \wedge x_5\overline{x_8} \wedge \overline{x_{10}}) \vee (\overline{x_4} \wedge x_5 \wedge x_7 \wedge \overline{x_8}) \vee (\overline{x_4} \wedge x_5 \wedge x_7 \wedge \overline{x_{10}}) \vee (x_5 \wedge x_7 \wedge \overline{x_8} \wedge \overline{x_{10}})$
Swi5	x_6		$(\overline{x_{10}} \wedge x_{11}) \vee (x_7 \wedge \overline{x_{10}}) \vee (x_7 \wedge x_{11}) \vee (x_6 \wedge x_{11}) \vee (x_7 \wedge x_6)$
Cdc20	x_7		$x_{10} \vee x_{11}$
Clb5	x_8	✓	$(\overline{x_7} \wedge x_8 \wedge \overline{x_9}) \vee (x_2 \wedge \overline{x_7} \wedge \overline{x_9}) \vee (x_2 \wedge \overline{x_7} \wedge x_8) \vee (x_2 \wedge x_8 \wedge \overline{x_9})$
Sic1	x_9	✓	$(\overline{x_4} \wedge \overline{x_{10}} \wedge \overline{x_8} \wedge x_9) \vee (x_6 \wedge \overline{x_4} \wedge \overline{x_{10}} \wedge \overline{x_8}) \vee (x_6 \wedge \overline{x_4} \wedge \overline{x_{10}} \wedge x_9) \vee (x_6 \wedge \overline{x_4} \wedge \overline{x_8} \wedge x_9) \vee (x_6 \wedge \overline{x_{10}} \wedge \overline{x_8} \wedge x_9) \vee (x_7 \wedge \overline{x_4} \wedge \overline{x_{10}} \wedge \overline{x_8}) \vee (x_7 \wedge \overline{x_4} \wedge \overline{x_{10}} \wedge x_9) \vee (x_7\overline{x_4} \wedge \overline{x_8} \wedge x_9) \vee (x_7 \wedge \overline{x_{10}} \wedge \overline{x_8} \wedge x_9) \vee (x_7 \wedge x_6 \wedge \overline{x_4} \wedge \overline{x_{10}}) \vee (x_7 \wedge x_6 \wedge \overline{x_4} \wedge \overline{x_8}) \vee (x_7 \wedge x_6 \wedge \overline{x_{10}} \wedge \overline{x_8}) \vee (x_7 \wedge x_6 \wedge \overline{x_4} \wedge x_9) \vee (x_7 \wedge x_6 \wedge \overline{x_{10}} \wedge x_9) \vee (x_7 \wedge x_6 \wedge \overline{x_8} \wedge x_9)$
Clb1	x_{10}	✓	$(\overline{x_9} \wedge \overline{x_7} \wedge \overline{x_5} \wedge x_{10}) \vee (x_8 \wedge \overline{x_9} \wedge \overline{x_7} \wedge \overline{x_5}) \vee (x_8 \wedge \overline{x_9} \wedge \overline{x_7} \wedge x_{10}) \vee (x_8 \wedge \overline{x_9} \wedge \overline{x_5} \wedge x_{10}) \vee (x_8 \wedge \overline{x_7} \wedge \overline{x_5} \wedge x_{10}) \vee (x_{11} \wedge \overline{x_9} \wedge \overline{x_7} \wedge \overline{x_5}) \vee (x_{11} \wedge \overline{x_9} \wedge \overline{x_7} \wedge x_{10}) \vee (x_{11}\overline{x_9} \wedge \overline{x_5} \wedge x_{10}) \vee (x_{11} \wedge \overline{x_7} \wedge \overline{x_5} \wedge x_{10}) \vee (x_{11} \wedge x_8 \wedge \overline{x_9} \wedge \overline{x_7}) \vee (x_{11} \wedge x_8 \wedge \overline{x_9} \wedge \overline{x_5}) \vee (x_{11} \wedge x_8 \wedge \overline{x_7} \wedge \overline{x_5}) \vee (x_{11} \wedge x_8 \wedge \overline{x_9} \wedge x_{10}) \vee (x_{11} \wedge x_8 \wedge \overline{x_7} \wedge x_{10}) \vee (x_{11} \wedge x_8 \wedge \overline{x_5} \wedge x_{10})$
Mcm1	x_{11}	✓	$x_8 \vee x_{10}$

For selecting seven entities to be observed and a control entity, we applied the method proposed in (VERDICCHIO; KIM, 2011). It defined Mcm1 as the most influential entity capable of shifting the CBNp between $\mathcal{U}$ and $\mathcal{D}$, so it was chosen as control entity. Consequently, we define three possible actions $A = \{$*non-intervention, Mcm1-inhibition, Mcm1-activation*$\}$, where a *non-intervention* action can always be applied, *Mcm1-inhibition* can only be applied when Mcm1 is active and *Mcm1-activation* only when it is inhibited. We do six simulations, in each of them we assume the CBNp is at one of the six undesirable attractors and generate an expe-

rience set composed of $\frac{10,000}{6}$ experiences by applying randomly selected actions. All frameworks use the experience set composed of 10,000 experiences to derive an intervention policy.

5.1.3 T-Helper cell differentiation network

T-Helper cells are a type of T-lymphocytes that are part of the immune response. In the differentiation process, T-Helper Th0 precursor cells can change to effector Th1 or Th2 cells, which means that a T-Helper cell is at Th0 BOA and after differentiation it is either at Th1 or Th2 BOA, and does not come back to Th0. The network modelling the differentiation process is available in (MENDOZA; XENARIOS, 2006), it has 23 genes and three attractors, each characterizing a cell type: Th0, Th1 and Th2. Table 9 shows all entities and the ones that are observed, we do not show prediction functions due to their complexity. We choose to observe 16 entities to keep partial observability in 60% as in other BTs; hence, there are 2^{23} states and 2^{14} possible observations.

In the intervention problem we solve, it is important to intervene in this CBNp in order to guide the differentiation process from Th0 to Th1 or from Th0 to Th2. For evaluation purposes, we consider that perturbations can take the CBNp to any BOA, so we still consider an infinite horizon. We applied the method proposed in (VERDICCHIO; KIM, 2011) for selecting 14 entities to observe and it identified entity GATA3 as control entity. Therefore, we define three possible actions $A = \{$*non intervention, Gata3 inhibition, Gata3 activation*$\}$, where *non intervention* action can always be applied, *Gata3 inhibition* only when Gata3 is active and *Gata3 activation* only when it is inhibited. After computing the transition function T, the MDP model can simulate experiences. We assume the CBNp is at Th0 attractor and use the MDP model to generate 10,000 experiences by applying randomly selected actions.

5.1.4 Human aging network

The systems biology of human aging is a compiled chart organized by Furber (2019) based upon studies from many prominent biogerontologists. The chart is a complex network illustrating many of the leading theories on human senescence and is updated when new information is available. Bryce, Verdicchio and Kim (2010) for-

Table 9: T-Helper cell differentiation network proposed by (MENDOZA; XENARIOS, 2006).

Entity	Observed
TCR	✓
IL-18	✓
IL-12	✓
IFN-γ	
IL-4	✓
IL-10	
NFAT	
IL-18R	
IL-12R	
IFN-γR	
IL-4R	✓
IL-10R	
IFN-β	✓
IRAK	
STAT4	
JAK1	✓
STAT6	✓
STAT3	✓
IFN-βR	✓
SOCS1	✓
Tbet	✓
STAT1	✓
GATA3	✓

malized a portion of the system into a BN composed of 18 entities. As authors pointed out, it has three BOAs, from which the largest one is considered the unhealthy BOA, the second-largest the healthy BOA and the smallest one an average health. Here, we consider average health and unhealthy as undesirable BOAs.

Table 9 shows all entities and their observability. Prediction functions are available at (BRYCE; VERDICCHIO; KIM, 2010), we do not show them due to their complexity. While the other biological-based networks are composed only by genes, the human aging network has different biological components, such as mitochondria DNA and reactive oxygen species. Regarding observability and control entity, we use the same seven observed entities used by Bryce, Verdicchio and Kim (2010), which allows to observe 39% of all entities and the same control entity 'Dietary Antioxidants'. Hence, the goal is to control a person's diet to improve her aging process based on exams that shows the activity of 7 out of 18 entities.

We repeated the same process of melanoma and yeast CBNps to obtain the MDP and two sets of experiences to total 10,000 experiences. At each simulation the CBNp is in an undesirable attractor and then the MDP model simulates 5,000 experiences by applying randomly selected actions.

Table 10: Human aging network converted by Bryce, Verdicchio and Kim (2010).

Entity	Observed
Mitochondria	
Mitochondrial Metabolism	
Mitochondria proteins	
Mitochondria DNA	
Inner mitochondrial membrane	
Lysosome	
Lysosome Membrane	✓
Lon Protease	
Nucleus DNA	
Nucleus DNA repair	
Cellular Antioxidants	
Dietary Antioxidants	✓
Apoptosis	
Lipofuscin	✓
Ions	✓
Reactive Oxygen Species	✓
Hydrogen Peroxide	✓
Damaged proteins and dysfunctional mitochondria	✓

5.1.5 Synthetic generated CBNp

We used the simulator available in (YOUSEFI; DOUGHERTY, 2013) to generate 100 synthetic CBNps and simulate state transitions. For each Boolean function $f^{(i)}$, this simulator arbitrarily selects 0 or 1 to be its answer based on a Beta distribution. All CBNps have 10 entities ($g = 10$), at most three of them can influence the value of a single entity (nodes have a maximum in-degree of three edges in the BT graph), at least two BOAs and an attractor cycle composed of at most three states. The largest BOA is considered the desirable BOA, and all others undesirable that should be avoided.

The simulator generates S, A and T for each synthetic generated CBNp and we defined functions R and C in Chapter 5. We assume the CBNp is at an undesirable attractor and simulate experiences from this starting state. After repeating this simulation for all undesirable attractors, the experience set has 10,000 experiences and all

frameworks use it to compute an intervention policy.

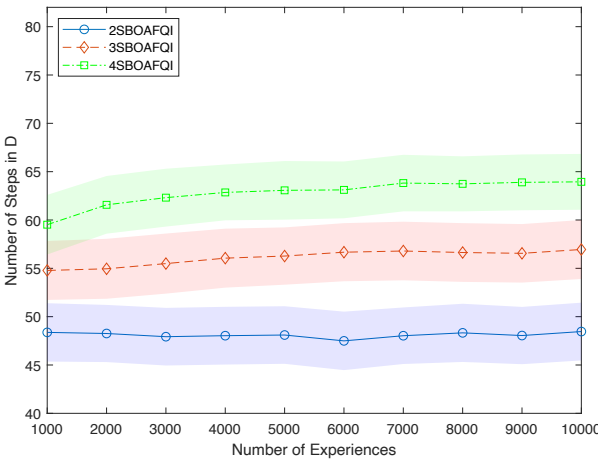
Although BOAConFE is scalable and can be used for a large CBNp, the discovery of attractor and BOAs is NP-hard (ZHANG et al., 2007). Therefore, using a large BNp would be computationally unfeasible in the experiments. Next, describe all conducted experiments and their results.

5.2 Number of steps m parameter

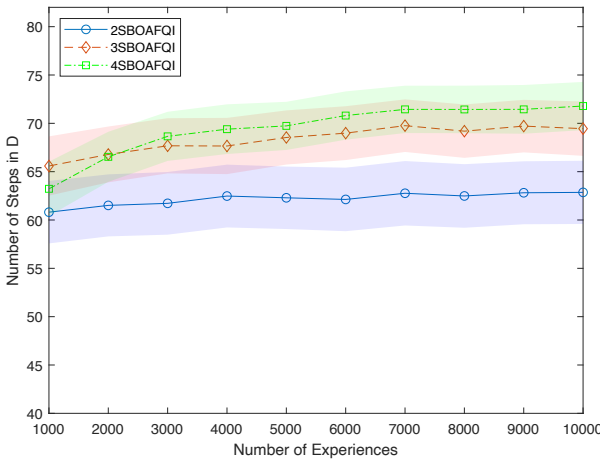
Our proposed BRL algorithm mSBOAFQI (Algorithm 4) considers m steps in an experience to tackle partial observability. For verifying whether having a greater m improves results, we test mSBOAFQI by varying m at different degrees of observability. We provide the same set of experiences to three versions of mSBOAFQI, each with a different m: $m = 2$ (2SBOAFQI), $m = 3$ (3SBOAFQI) and $m = 4$ (4SBOAFQI). Each synthetic CBNp provides the optimal $P(o \in \mathcal{D})$ and one experience set $\mathscr{F}$ with 10,000 experiences, from which we select a subset with size ranging from 1,000 to 10,000 and observability from 3 to 9 entities. Although the experience set is the same for all versions of mSBOAFQI, 2SBOAFQI uses only current and next step, 3SBOAFQI uses current, next and one previous step and 4SBOAFQI uses four steps. All versions use conventional stochastic policy (Equation 2.13), so the comparison between policies is more straightforward.

Figure 16 shows the results for #Steps$_{\mathcal{D}}$ when ranging observability and experience set size. Figure 16a shows #Steps$_{\mathcal{D}}$ when all algorithms have poor observability and can only observe 30% of all entities (3 out of 10 entities). In this case, 4SBOAFQI provides a higher quality policy than 3SBOAFQI and 2SBOAFQI for any number of experiences. This means that when observability is poor, having more steps provides more information about the current observation and improves decision making. On the other hand, Figures 16b and 16c show that as observability increases, 4SBOAFQI requires more experiences to compute a high quality policy. In other words, having more steps introduces more variables (observation-action pairs) to learn, which require more data to derive a good policy. Therefore, m should be chosen by a specialist based on the number of experiences and the expected observability.

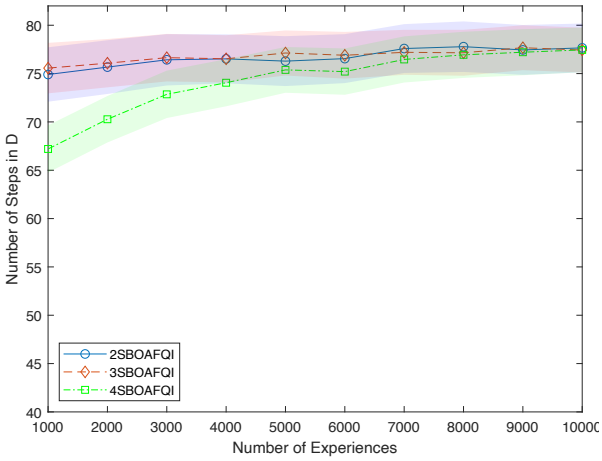
The metric #Steps$_{\mathcal{D}}$ shows whether mSBOAFQI is correctly solving the intervention



(a) 3 Observed entities.



(b) 6 Observed entities.

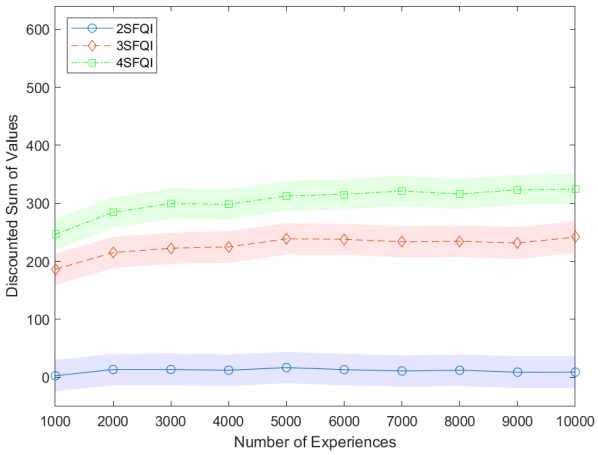


(c) 9 Observed entities.

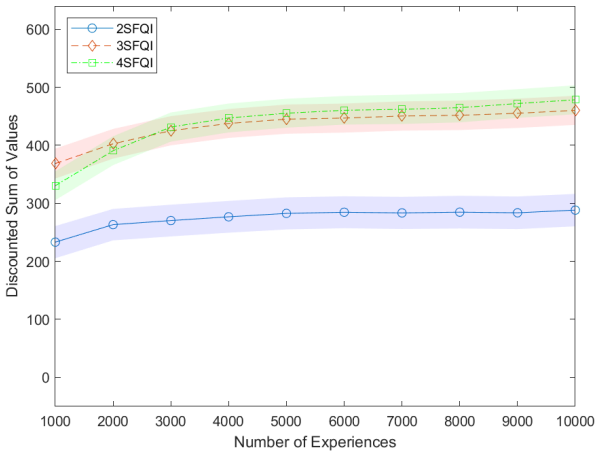
Figure 16: Average number of steps in desirable BOA in 100 CBNps for 2SBOAFQI($m = 2$), 3SBOAFQI($m = 3$) and 4SBOAFQI($m = 4$). The shaded area represents the 95% confidence interval. It is observed that m must be chosen by a specialist, taking into account the number of experiences and the observability of the system.

problem. At the same time, it is also important to show the discounted sum of values (rewards minus costs) to verify whether the BRL algorithm is working as expected. Figure 17 shows the results for the discounted sum of values when ranging observability and experience set size. It is possible to note that it has the same pattern as Figure 16, the discounted sum increases as the number of experiences and the number of observed entities increases. Since the results for #Steps$_{\mathcal{D}}$ and the discounted sum of values have the same pattern, in the next experiments we only show #Steps$_{\mathcal{D}}$.

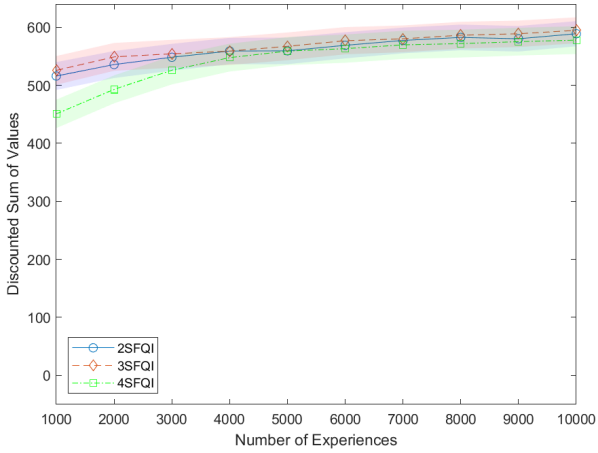
This experiment showed that considering more steps is advantageous in partially observable BTs. This is very clear in Figure 16a, since having more steps increased #Steps$_{\mathcal{D}}$ for all experience set sizes. In all other experiments, we use six observed



(a) 3 Observed entities.



(b) 6 Observed entities.



(c) 9 Observed entities.

Figure 17: Average of the discounted sum of values in 100 CBNps for 2SBOAFQI($m = 2$), 3SBOAFQI($m = 3$) and 4SBOAFQI($m = 4$). The shaded area represents the 95% confidence interval.

entities ($n = 6$), which is a mid-term observability that can provide good results. Figure 16b shows that learning curve of all algorithms grows slower after 7,000 experiences and 4SBOAFQI has a numerical advantage over 3SBOAFQI. Therefore, in all other experiments we use 4SBOAFQI and 7,000 experiences.

5.3 Solution quality with a computed $P(o \in \mathcal{D})$

In most real world cases we do not expect that an optimal $P(o \in \mathcal{D})$ is available, hence it is important to evaluate how the computed $P(o \in \mathcal{D})$ compares to the optimal one given by the CBNp model. For this experiment, we test the accuracy of $P(o \in \mathcal{D})$ in two scenarios: small and big data sets, with 1,000 and 10,000 samples, respectively. We set the number of cycles to $n_r = 5$ since observability is not severe and the size

of the observation set for computing $P(o \in \mathcal{D})$ is equal to the size of the experience set, so 1,000 experiences and 1,000 observations or 10,000 experiences and 10,000 observations.

Computed $P(o \in \mathcal{D})$ that was generated using 1,000 observations has an average mean square error of 0.14, while 10,000 observations has 0.07. Having more observations increases the accuracy of $P(o \in \mathcal{D})$ and using a more accurate $P(o \in \mathcal{D})$ can produce better results, which is demonstrated in Table 11 and Table 12. Using 1,000 observations for computing $P(o \in \mathcal{D})$, mSBOAFQI generated policies providing worse results than optimal $P(o \in \mathcal{D})$ for all metrics. Using 10,000 observations for computing $P(o \in \mathcal{D})$, computed and optimal $P(o \in \mathcal{D})$ generate statistically comparable results according to statistical test Kruskall-Wallis at 5% of significance and paired Wilcoxon signed rank at 5% of significance. This means that having more observations it is possible to compute a more accurate $P(o \in \mathcal{D})$ that can provide better results.

Table 11: Results for mSBOAFQI (m=4) using 1,000 experiences, optimal and computed $P(o \in \mathcal{D})$. Numbers are average from 100 CBNps with the 95% confidence interval. Numbers in bold correspond to the best performance achieved in each metric.

$P(o \in \mathcal{D})$	#Steps$_{\mathcal{D}}$	$\#a^{ia}$	$a^{ia}_{\mathcal{D}}$
Optimal	**67.3 ± 3.7**	**30.1 ± 2.4**	**22.3 ± 2.2**
Computed	61.3 ± 3.5	34.9 ± 2.3	26.3 ± 2.4

Table 12: Results for mSBOAFQI (m=4) using 10,000 experiences, optimal and computed $P(o \in \mathcal{D})$. Numbers are average from 100 CBNps with the 95% confidence interval. Numbers in bold correspond to the best performance achieved in each metric.

$P(o \in \mathcal{D})$	#Steps$_{\mathcal{D}}$	$\#a^{ia}$	$a^{ia}_{\mathcal{D}}$
Optimal	**72.1 ± 3.5**	**28.8 ± 2.3**	**19.8 ± 2.4**
Computed	**71.7 ± 3.3**	**29.1 ± 2.2**	**20.1 ± 2.4**

Although 1,000 observations is a small data set, it allowed to compute $P(o \in \mathcal{D})$ with an accuracy that is high enough to derive high quality policies. This is an important result because $P(o \in \mathcal{D})$ is used in stages four, five and six of BOAConFE and is a relevant part of our framework.

5.4 The impact of the reward function

For finding a good solution it is vital that the control system receives a reward signal that properly specifies the goal. Based only on the reward signal it receives from an

environment, the control system learns a policy to perform the task. In this section we do two experiments to verify which reward function is better and whether a reward can be assigned based on their expected average reward (Equation 4.2) when there is partial observability.

5.4.1 Comparison between reward functions

Here, we describe experiments using (PAL; DATTA; DOUGHERTY, 2006), (SIRIN; POLAT; ALHAJJ, 2013) and our reward function (Equation 4.2) when all rewards are given by a specialist. The reward function proposed by (PAL; DATTA; DOUGHERTY, 2006) is equal to Equation 2.11, so it rewards for a desirable next observation. The one proposed by (SIRIN; POLAT; ALHAJJ, 2013) rewards for a desirable current observation, and both (SIRIN; POLAT; ALHAJJ, 2013) and (PAL; DATTA; DOUGHERTY, 2006) penalize the application of interventions. For all functions with use the same values: $\kappa = 100$ and $\psi = 20$, where in (SIRIN; POLAT; ALHAJJ, 2013) function it receives κ for being in $\mathcal{D}$ and in (PAL; DATTA; DOUGHERTY, 2006) function it receives κ for going to $\mathcal{D}$. For a fair and unbiased evaluation, besides mSBOAFQI we use LSFTDI and FQI-Sarsa. We use optimal $P(o \in \mathcal{D})$ to have a straightforward comparison of the reward function for mSBOAFQI.

Table 13 shows the results. Our reward function has the best results for all evaluated algorithms and for all metrics, which indicate that considering both current and next observations improve the policy quality. The second best reward function is the one proposed by (PAL; DATTA; DOUGHERTY, 2006) that improved results for all algorithms in comparison with the function proposed by (SIRIN; POLAT; ALHAJJ, 2013). All comparisons were done using the statistical tests Kruskall-Wallis at 5% of significance and paired Wilcoxon signed rank at 5% of significance.

Results indicate that it is important to choose an appropriate reward function that can define the desirability of each experience. In this control problem, the goal is to make a CBNp reach and remain in $\mathcal{D}$, hence by considering current and next observation our reward function can better define the desirability of an experience.

Table 13: Results for FQI-Sarsa, LSFTDI and mSBOAFQI (m=3) using 7,000 experiences and optimal $P(o \in \mathcal{D})$. Numbers are average from 100 CBNps with the 95% confidence interval. Numbers in bold correspond to the best performance achieved in each metric.

Algorithm	Reward	#Steps$_{\mathcal{D}}$	#a^{ia}	$a^{ia}_{\mathcal{D}}$
FQI-Sarsa	(PAL; DATTA; DOUGHERTY, 2006)	59.2 ± 4.5	36.2 ± 2.6	27.2 ± 2.8
	(SIRIN; POLAT; ALHAJJ, 2013)	55.4 ± 4.1	37.2 ± 2.5	29.3 ± 3.0
	Proposed reward function	62.5 ± 3.5	35.4 ± 2.8	24.7 ± 3.0
LSFTDI	(PAL; DATTA; DOUGHERTY, 2006)	54.3 ± 4.0	36.5 ± 2.7	29.9 ± 3.0
	(SIRIN; POLAT; ALHAJJ, 2013)	55.4 ± 4.1	37.6 ± 2.4	30.8 ± 3.0
	Proposed reward function	56.9 ± 4.1	35.4 ± 2.7	28.8 ± 3.1
mSBOAFQI	(PAL; DATTA; DOUGHERTY, 2006)	70.5 ± 3.5	29.8 ± 2.3	21.3 ± 2.4
	(SIRIN; POLAT; ALHAJJ, 2013)	70.0 ± 3.4	31.2 ± 2.4	22.1 ± 2.5
	Proposed reward function	**72.6 ± 3.4**	**28.5 ± 2.4**	**19.6 ± 2.5**

5.4.2 Reward and partial observability

In partial observable BTs and BOA-avoidance problems, it is difficult to manually assign rewards to experiences, mainly for a large data set with several thousands of experiences. For state-avoidance problems the usual approach is to provide a high reward for experiences in which an observable entity is active or inhibited, but in BOA-avoidance problems this approach does not work. Here, we test if it is possible to assign the average reward an experience is expected to receive (Equation 4.3) instead of the real and unknown reward. We use our proposed reward function, since it has the best result overall in the previous experiment (see Subsection 5.4.1).

Besides comparing real rewards to assigning the average reward an experience expect to receive, we also compare automatically assigning rewards using optimal and computed $P(o \in \mathcal{D})$ with 7,000 observations. We use FQI-Sarsa algorithm instead of mSBOAFQI, because FQI-Sarsa does not use $P(o \in \mathcal{D})$ in its computation, so it will not further disseminate errors of the computed $P(o \in \mathcal{D})$.

Table 14 shows the average results for 100 synthetic CBNps. It shows that BOA-ConFE can automatically assign rewards in partially observable BTs, even when specialists are not available to classify an observation in either $\mathcal{D}$ or $\mathcal{U}$. Although using the real rewards produce higher quality policies, automatically assigning them based on $P(o \in \mathcal{D})$ is a good alternative when a specialist is not available. In the case of using optimal $P(o \in \mathcal{D})$ to compute rewards, it can derive policies with almost the same

quality as using real rewards. However, computed $P(o \in \mathcal{D})$ provided worse results, which shows the importance of computing an accurate $P(o \in \mathcal{D})$ to be able of assigning rewards that are close to the real ones. All comparisons were done for all possible combinations of pairs using the statistical tests Kruskall-Wallis at 5% of significance and paired Wilcoxon signed rank at 5% of significance.

Table 14: Results for reward assignment when it is not possible to know BOA desirability of observations. Numbers are average from 100 CBNps with the 95% confidence interval. Numbers in bold correspond to the best performance achieved in each metric.

Policy			#Steps$_{\mathcal{D}}$	#a^{ia}	$a^{ia}_{\mathcal{D}}$
Non intervention policy			31.1 ± 2.0	0.0 ± 0.0	0.0 ± 0.0
Arbitrary policy			40.2 ± 4.0	47.6 ± 2.7	51.7 ± 3.2
Reward	$P(o \in \mathcal{D})$		#Steps$_{\mathcal{D}}$	a^{ia}	$a^{ia}_{\mathcal{D}}$
Real (Equation 4.2)	-		**62.5 ± 3.5**	**35.4 ± 2.8**	**24.7 ± 3.0**
Computed (Equation 4.3)	Optimal		60.5 ± 4.2	36.9 ± 3.0	26.1 ± 3.3
Computed (Equation 4.3)	Computed observations	1.000	56.2 ± 4.6	38.9 ± 3.0	28.1 ± 3.3
Computed (Equation 4.3)	Computed observations	10.000	59.7 ± 4.4	37.3 ± 2.9	27.0 ± 3.3

5.5 Reducing the number of interventions with BOA policy

It is extremely important to derive an effective policy that can quickly shift a BT to $\mathcal{D}$ while reducing the number of interventions. A policy that takes a long time to shift a BT from $\mathcal{U}$ to $\mathcal{D}$ can have disastrous consequences in a patient's health, since diseases can worsen in a short period of time. For example, malaria quickly progress to severe forms and even lead to death without an effective treatment (ORGANIZATION, 2015). Furthermore, if it indicates many interventions, this policy can be too costly to implement in the real world as each intervention is a therapy or drug that may be expensive and may have side effects. In this experiment, we use FQI-Sarsa to verify how policy computation can influence all metrics. For the Boltzmann policy (Equation 2.14) we set $\tau = 1$.

Table 15 shows all results. Sirin, Polat and Alhajj (2017) proposed to use a conventional stochastic policy (Equation 2.13) to deal with partial observability, but it is worse than a deterministic policy (Equation 2.6) in average. A conventional stochastic

policy applies too many interventions and them can make a CBNp shift back to $\mathcal{U}$ or apply inappropriate actions. Boltzmann policy derived higher quality policies than a conventional stochastic policy, but it is worse than a determinitic policy. Deterministic policy indicates only one action for each observation, so it may indicate only low quality actions for some states. As results shows, it was the second best policy, with #Steps$_{\mathcal{D}}$ close to BOA policy, but in the biological-based BT experiments, we demonstrate that it can perform poorly in some cases. BOA policy provided the best results for all metrics, mainly for optimal $P(o \in \mathcal{D})$. BOA policy using computed $P(o \in \mathcal{D})$ provided the second best results and it shows that $P(o \in \mathcal{D})$ inaccuracy does not highly affect it. All comparisons were done for all possible combinations of pairs using the statistical tests Kruskall-Wallis at 5% of significance and paired Wilcoxon signed rank at 5% of significance.

Table 15: Results for FQI-Sarsa using different policies. Numbers in bold correspond to the best performance achieved in each metric.

Policy		#Steps$_{\mathcal{D}}$	#a^{ia}	$a^{ia}_{\mathcal{D}}$
Uncontrolled		31.1 ± 2.0	0.0 ± 0.0	0.0 ± 0.0
Arbitrary		39.8 ± 3.5	47.7 ± 2.1	50.3 ± 2.8
Deterministic (Equation 2.6)		76.2 ± 3.9	22.1 ± 5.5	19.3 ± 6.0
Boltzmann (Equation 2.14)		69.7 ± 3.6	31.1 ± 1.9	21.1 ± 2.3
Conventional Stochastic (Equation 2.13)		63.1 ± 4.3	35.4 ± 2.7	25.9 ± 3.1
Policy	$P(o \in \mathcal{D})$	#Steps$_{\mathcal{D}}$	#a^{ia}	$a^{ia}_{\mathcal{D}}$
BOA (Equation 4.6)	Optimal	**79.1 ± 3.7**	**12.6 ± 1.8**	**3.8 ± 1.0**
BOA (Equation 4.6)	Computed	77.1 ± 3.8	13.4 ± 1.9	4.6 ± 1.0

5.6 Comparison between model-based frameworks and BOAConFE

Model-based control frameworks infer a model and compute a policy from it. Although these framework can derive an optimal policy given the model, the quality of their solutions are related to the quality of the inferred model, hence a policy generated from an inaccurate mode cannot effectively control the real BT. Partial observability and perturbations can hinder the inference process and result in inaccurate models. In addition, it is worth noting that one of the most limiting aspects in the model-based BT control is scalability, since very large networks are intractable.

In order to verify the performance of model-based frameworks in partially observ-

able BTs, we generate 7,000/3 transition samples with six observed genes for each action. We infer a model using the Best-fit method (SHMULEVICH et al., 2003), compute BOAs and define the BOA that has attractor $\sigma = \Omega_{\mathcal{D}}$ as desirable. This means that inferred model has the same desirable attractor as original CBNp, but not necessarily the same BOA structure. Then, we apply the dynamic programming method called Value Iteration (BELLMAN, 1957; PAL; DATTA; DOUGHERTY, 2006) to compute an optimal deterministic policy (Equation 2.6) given the inferred model.

Table 16 shows the results of this experiment. The last metric $P(pathway)$ is the average probability of following all possible pathways, where a pathway is defined as a sequence between five and ten states. Inferring a 6-gene model from a series of observations generated by a 10-gene BT and then calculating an optimal policy from it has a far worse result than applying BOAConFE directly using past experiences. Value Iteration algorithm calculates an optimal policy for the inferred model, but the inferred model does not correctly describe how the actual BT works. This results show that when it is not possible to guarantee an accurate model, applying BOAConFE provides higher quality policies than using an inaccurate model. Still, inferring a model and applying Value Iteration provides better results than an arbitrary policy, even though they produce statistically the same #Steps$_{\mathcal{D}}$. Regarding $P(pathway)$, BOAConFE has a much higher probability of letting the BT follow a pathway, so in average it does not prevent the execution of biological tasks.

Table 16: Results for Value Iteration and BOAConFE. Numbers are average from 100 CBNp with a 95% confidence interval. Numbers in bold correspond to the best performance achieved in each metric.

Policy	#Steps$_{\mathcal{D}}$	#a^{ia}	#$a^{ia}_{\mathcal{D}}$	P(pathway)
Non intervention policy	37.3 ± 1.0	0.0 ± 0.0	0.0 ± 0.0	100.0 ± 0.0
Arbitrary policy	57.2 ± 1.8	48.4 ± 1.1	28.5 ± 1.2	0.0 ± 0.0
Framework	**#Steps$_{\mathcal{D}}$**	**#a^{ia}**	**#$a^{ia}_{\mathcal{D}}$**	**P(pathway)**
Value Iteration	58.4 ± 5.3	16.5 ± 3.6	9.4 ± 2.2	45.6 ± 9.5
BOAConFE	**78.6 ± 3.4**	**11.8 ± 1.6**	**4.6 ± 1.1**	**73.8 ± 5.4**

5.7 Comparison between framework proposals

In the previous experiments, we tested each phase of our proposed framework separately. Although each phase works well in comparison with other proposals, it is highly important to evaluate the complete framework. In this experiment, we compare BOAConFE with the ones proposed in (SIRIN; POLAT; ALHAJJ, 2017) and (SOOTLA et al., 2014), their differences are highlighted in Table 17.

For making all frameworks pursue the same goals, we applied the same reward function for all of them. We use our reward function (Equation 4.3) because it provided better results than other functions in Section 5.4 and can be used in partially observable BTs. We apply BOAConFE phases two and four to compute $P(o \in \mathcal{D})$ and rewards for all experiences using 7,000 observations. Then all frameworks receives the same batch of 7,000 experiences and compute an intervention policy from it. Note that the reward function is one of our proposals and in this section we are using only our proposed reward function. Therefore, all control methods have the same goal, which makes the comparison among them more straightforward and fair.

Table 17: Evaluated frameworks characteristics.

	BOAConFE	(SIRIN; POLAT; ALHAJJ, 2017)	(SOOTLA et al., 2014)
BOA-avoidance	Yes	No	No
Use $P(o \in D)$	Yes	No	No
Reward assignment	Automatic	Manual	Manual
BRL Method	mSBOAFQI	LSFTDI	FQI
Policy	BOA (Equation 4.6)	Conventional Stochasticy (Equation 2.13)	Deterministic (Equation 2.6)

Regarding reward assignment, we classified as automatic when a framework can assign rewards for partially observable BTs without requiring additional information or a specialist. Although the framework proposed in (SIRIN; POLAT; ALHAJJ, 2017) can automatically assign rewards for the problem they solve, it is a very specific case. Their framework assigns a high reward when the current observation has a given observed target entity as (in)active. Thus, when the target entity is not observed or in partially observable BOA avoidance problems, their framework cannot automatically assign rewards. The last framework can automatically assign rewards for fully observable prob-

lems, but not for partially observable ones.

5.7.1 Synthetic networks

Here, we compare all evaluated frameworks in a wide range of BTs using 100 synthetic CBNps. Table 18 shows the results of this experiment and it clearly demonstrates that all frameworks provide better results for most metrics than an arbitrary policy or not intervening in the CBNp. Not applying interventions when a CBNp is in $\mathcal{D}$ is the best approach and provides P(pathway) = 100%, but in partially observable BTs it is difficult to know with certainty whether a BT is in $\mathcal{D}$ or not. Hence, using a non intervention policy is not appropriate when a physician cannot attest the cure with certainty or for incurable diseases such as diabetes and high blood pressure.

Regarding the evaluated frameworks, the one proposed by Sirin, Polat and Alhajj (2017) provides the worst results for all metrics. There are two main reasons for this poor performance: 1) their BRL algorithm LSFTDI uses Least-Squares regression and it does not provide good approximations given the authors proposed features; 2) conventional stochastic policy (Equation 2.13) applies too many non-optimal actions. Sootla et al. (2014) has the second best result overall and much superior metrics in comparison to Sirin, Polat and Alhajj (2017). Because Sootla et al. (2014) uses a deterministic policy (Equation 2.6), when it indicates an intervention in any observation of the pathway it makes P(pathway) = 0. This makes its policy be either optimal in regard to following pathways or worse than an arbitrary policy. BOAConFE provides the best results in all metrics and is the only framework that CBNps can follow a pathway with an average probability higher than 50%. This is an important result because when a BT cannot follow a pathway, it cannot properly execute biological tasks.

The advantage of using synthetically generated CBNps is that it is possible to evaluate all frameworks on networks with different complexities. When $\mathcal{D}$ is large, it is easier to reach it and when $\mathcal{D}$ is small, it is more difficult. In part, the complexity can be noted by $\#\text{Steps}_{\mathcal{D}}$ when applying an uncontrolled and arbitrary policy. In this case, the smaller the numbers, the more complex the intervention problem is because it is not easy to reach $\#\text{Steps}_{\mathcal{D}}$ arbitrarily. For complex problems, an effective intervention policy is necessary for properly leading a CBNp to $\mathcal{D}$.

Table 18: Results for all evaluated frameworks. Numbers are average from 100 synthetically generated CBNps with the 95% confidence interval. Numbers in bold correspond to the best framework performance achieved in each metric.

Policy	#Steps$_{\mathcal{D}}$	#a^{ia}	$a^{ia}_{\mathcal{D}}$	P(pathway)
Uncontrolled	32.0 ± 2.0	0.0 ± 0.0	0.0 ± 0.0	100.0 ± 0.0
Arbitrary policy	42.2 ± 3.7	45.4 ± 2.5	47.3 ± 3.3	0.4 ± 0.3
Framework	#Steps$_{\mathcal{D}}$	#a^{ia}	$a^{ia}_{\mathcal{D}}$	P(pathway)
(SOOTLA et al., 2014)	66.4 ± 4.9	17.8 ± 4.0	11.9 ± 3.5	41.9 ± 9.1
(SIRIN; POLAT; ALHAJJ, 2017)	55.1 ± 4.4	35.9 ± 2.7	29.7 ± 3.2	9.4 ± 4.2
BOAConFe	**78.4** ± **3.4**	**11.9** ± **1.6**	**4.7** ± **1.1**	**73.6** ± **5.4**

5.7.2 Melanoma biological network

The goal of intervening in the melanoma CBNp is to avoid states with the gene WNT5A active, which is a highly discriminant factor for melanoma metastasizing. This type of goal describes a state-avoidance problem and state-avoidance frameworks such as (FARYABI; DATTA; DOUGHERTY, 2007) and (SIRIN; POLAT; ALHAJJ, 2017) can solve this intervention problem. Another option is to model a state-avoidance problem as BOA-avoidance. As Bryce, Verdicchio and Kim (2010) explained, state-avoidance problems can be modeled as BOA-avoidance when at least one of the attractors is composed of desirable states. These attractors are defined as desirable attractors, the BOAs leading to them are desirable and all others are undesirable BOAs. Then the goal changes from avoiding undesirable states to avoiding undesirable BOAs.

In this experiment, besides the comparison among all evaluated frameworks, we also evaluate how BOA-avoidance frameworks perform when solving state-avoidance problems. Table 19 shows the result of this experiment. The last column shows the number of steps in desirable states (#Steps$_{states}$). The framework proposed by Sootla et al. (2014) has the best results in all metrics. An interesting result is that their framework solving a BOA-avoidance problem has the highest #Steps$_{states}$, which is higher than their own framework solving a state-avoidance problem. This suggests that solving a state-avoidance problem as BOA-avoidance may produce better results. This is explained by the fact that a BT is expected to cycle in an attractor most part of the time, hence when this attractor is a desirable state, the BT is going to cycle a desirable state most part of the time.

BOAConFE solving a BOA-avoidance problem has the second best result overall. The main reason for BOAConFE losing for (SOOTLA et al., 2014) is the computed $P(o \in \mathcal{D})$ inaccuracy. While the optimal $P(o \in \mathcal{D})$ can indicate with certainty when a given observation is desirable or not, the computed one cannot. For the Melanoma BT, $P(o \in \mathcal{D})$ is 1 or 0 for most observations, so usually an observation has all its states in $\mathcal{D}$ or all in $\mathcal{U}$ and a deterministic policy can be highly effective in this case because the same action can be applied for all observations in $\mathcal{D}$: a non-intervention action. Consequently, it performs optimally once the CBNp reaches a desirable BOA. It is noteworthy that our proposal could be as good as (SOOTLA et al., 2014) with an optimal $P(o \in \mathcal{D})$. In this case, BOA policy would indicate only non-intervention actions for observations in $\mathcal{D}$ because $P(o \in \mathcal{D}) = 1$.

Table 19: Results for the melanoma biological system. Numbers are average from 30 executions with the 95% confidence interval. Numbers in bold correspond to the best performance achieved in each metric.

Policy		#Steps$_{\mathcal{D}}$	#a^{ia}	$a^{ia}_{\mathcal{D}}$	#Steps$_{states}$
Uncontrolled		43.0 ± 0.4	0.0 ± 0.0	0.0 ± 0.0	41.9 ± 0.0
Arbitrary policy		53.7 ± 2.9	45.7 ± 4.5	26.3 ± 2.1	64.9 ± 2.1
Type	Framework	#Steps$_{\mathcal{D}}$	#a^{ia}	$a^{ia}_{\mathcal{D}}$	#Steps$_{states}$
BOA Avoidance	(SOOTLA et al., 2014)	**97.4 ± 0.0**	**2.3 ± 0.2**	**0.3 ± 0.1**	**95.1 ± 0.1**
	(SIRIN; POLAT; ALHAJJ, 2017)	74.9 ± 5.6	32.3 ± 4.2	21.6 ± 1.6	80.2 ± 1.6
	BOAConFE	96.5 ± 0.1	2.8 ± 0.1	0.4 ± 0.0	94.5 ± 0.0
State Avoidance	(SOOTLA et al., 2014)	95.6 ± 2.5	4.2 ± 0.3	1.9 ± 0.3	94.0 ± 0.3
	(SIRIN; POLAT; ALHAJJ, 2017)	72.2 ± 6.1	32.8 ± 4.6	22.0 ± 1.9	78.0 ± 1.9

5.7.3 Yeast cell cycle biological network

In order to further evaluate the frameworks we apply them into the yeast cell cycle CBNp to shift it from $\mathcal{U}$ to $\mathcal{D}$, while letting the CBNp follow the pathway modelled by (LI et al., 2004). Table 20 shows the results of this experiment. This time, it is not straightforward to indicate the best framework because each one of them won in at least one metric.

The framework proposed by Sirin, Polat and Alhajj (2017) has the best #Steps$_{\mathcal{D}}$,

while having the worst results for the other three metrics. A P(pathway) = 0.0 is a terrible result, because the BT is seldom allowed to execute biological tasks. BOAConFE has the second best #Steps$_{\mathcal{D}}$ and P(pathway), and the best #a^{ia} and $a^{ia}_{\mathcal{D}}$. The Sootla et al. (2014) proposal has the best P(pathway) and the second best #a^{ia} and $a^{ia}_{\mathcal{D}}$. Therefore, Sootla et al. (2014) can always execute biological tasks with the disadvantage of applying more interventions than BOAConFE. The best result is problem dependent, for example, when the probability of perturbations is small Sootla et al. (2014) framework is the best approach because it is far more important to execute pathways. When it is higher, BOAConFE may be the best approach for reducing the number of interventions.

Table 20: Results for the yeast cell cycle biological network. Numbers are average from 30 executions with the 95% confidence interval. Numbers in bold correspond to the best performance achieved in each metric.

Policy	#Steps$_{\mathcal{D}}$	#a^{ia}	$a^{ia}_{\mathcal{D}}$	P(pathway)
Uncontrolled	49.4 ± 0.4	0.0 ± 0.0	0.0 ± 0.0	1.0 ± 0.0
Arbitrary policy	72.2 ± 0.9	51.4 ± 7.6	37.5 ± 6.7	0.0 ± 0.0
Algorithm	**#Steps$_{\mathcal{D}}$**	**#a^{ia}**	**$a^{ia}_{\mathcal{D}}$**	**P(pathway)**
(SOOTLA et al., 2014)	63.1 ± 2.3	21.1 ± 4.5	3.3 ± 3.5	**1.0 ± 0.1**
(SIRIN; POLAT; ALHAJJ, 2017)	**72.9 ± 1.0**	44.6 ± 1.2	34.9 ± 0.5	0.0 ± 0.0
BOAConFE	68.9 ± 0.4	**17.5 ± 0.9**	**2.0 ± 0.1**	0.9 ± 0.0

5.7.4 T-Helper cell differentiation network

The T-Helper cell differentiation network has 23 entities and $2^{23} = 8,388,608$ possible states, from which we observe 14 entities. In this experiment, our goal is to show that BOAConFE can be used in larger biological networks and provide good results. We divide this experiment in two parts: in the first one the goal is to guide the differentiation from Th0 to Th1 and in the second one from Th0 to Th2. In both tests we use the same set of observed entities and GATA3 as control entity.

Table 21 shows the results for the differentiation to Th1. The framework proposed by Sootla et al. (2014) has the best results for all metrics, closely followed by BOAConFE. Sirin, Polat and Alhajj (2017) has the worst results and even though it has the #Steps$_{\mathcal{D}}$ greater than 90, it still applied too many intervention actions.

Table 21: Results for the differentiation from Th0 to Th1. Numbers are average from 30 executions with the 95% confidence interval. Numbers in bold correspond to the best performance achieved in each metric.

Policy	#Steps$_{\mathcal{D}}$	#a^{ia}	$a^{ia}_{\mathcal{D}}$
Uncontrolled	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Arbitrary policy	64.7 ± 2.4	47.9 ± 1.3	31.9 ± 0.9
Algorithm	**#Steps$_{\mathcal{D}}$**	**#a^{ia}**	**$a^{ia}_{\mathcal{D}}$**
(SOOTLA et al., 2014)	**99.0 ± 0.0**	**1.0 ± 0.0**	**0.0 ± 0.0**
(SIRIN; POLAT; ALHAJJ, 2017)	94.4 ± 0.4	39.2 ± 0.7	35.8 ± 0.5
BOAConFE	98.0 ± 0.0	2.2 ± 0.0	0.6 ± 0.0

Table 22 shows the results for the differentiation to Th2. This time (SOOTLA et al., 2014) has the worst #Steps$_{\mathcal{D}}$ and the worst result overall. It has not applied a single intervention, which means that the visited observations, including the Th0 attractor, belongs to $\mathcal{D}$ and $\mathcal{U}$. In this case, (SOOTLA et al., 2014) considered that not applying interventions in them is the best action because they could be in $\mathcal{D}$. Indeed, when these observations **are in desirable BOAs**, not applying interventions provide the best long term rewards, but their framework does not know whether they are in $\mathcal{D}$ or not.

The difference between current and previous results lies in the observations and their desirability. For example, attractor Th0 generates observation o_0. When the goal is to reach Th1 BOA, all observations o_0 are in undesirable BOAs and a policy that applies an intervention action in o_0 seems reasonable. On the other hand, when the goal is to reach Th2 BOA, almost all observations o_0 are in $\mathcal{D}$. This time, choosing a single action for observation o_0 is not straightforward and (SOOTLA et al., 2014) chose to not apply interventions. Although this choice is correct in most cases, it makes a CBNp stay in Th0 attractor.

The framework proposed by (SIRIN; POLAT; ALHAJJ, 2017) has the second best results for all metrics. BOAConFE has the best result overall and it showed that it is robust in partially observable CBNps. The computed policy probabilistically indicated an intervention action in attractor Th0, which made network leave the undesirable attractor and it also indicated non intervention actions most of the time when the CBNp was in the desirable BOA of Th2.

Table 22: Results for the differentiation from Th0 to Th2. Numbers are average from 30 executions with the 95% confidence interval. Numbers in bold correspond to the best performance achieved in each metric.

Policy	#Steps$_{\mathcal{D}}$	#a^{ia}	$a^{ia}_{\mathcal{D}}$
Uncontrolled	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Arbitrary policy	25.8 ± 1.9	47.6 ± 1.4	12.7 ± 0.8
Algorithm	**#Steps$_{\mathcal{D}}$**	**#a^{ia}**	**$a^{ia}_{\mathcal{D}}$**
(SOOTLA et al., 2014)	0.0 ± 0.0	**0.0 ± 0.0**	**0.0 ± 0.0**
(SIRIN; POLAT; ALHAJJ, 2017)	84.5 ± 0.2	6.9 ± 0.1	0.5 ± 0.1
BOAConFE	**89.6 ± 0.3**	5.7 ± 0.1	0.3 ± 0.0

5.7.5 Human aging network

The human aging BT has 18 entities and three BOAs: unhealthy, average health and healthy. We apply all frameworks in two settings: starting from the unhealthy attractor and from the average health attractor. In this experiment, the observability is poor and each framework can only observe 7 out of 18 entities, so the state space composed of 2^{18} = 262,144 possible states is represented by 128 different observations. This makes each observation represent 2,048 different states and it may be more difficult to get high quality policies that apply the best action often.

Tables 23 and 24 show the result when the CBNp starts from an unhealthy and average health attractor, respectively. An interesting point to note is the difference between #Steps$_{\mathcal{D}}$ in both tables for the non intervention policy. It is much easier to shift from the average health attractor to healthy BOA (#Step$_{\mathcal{D}}$ = 48.3) than from unhealthy to healthy (#Step$_{\mathcal{D}}$ = 17.4) due to perturbations. This is explained by the size of each BOA and how they are connected. The unhealthy BOA is the biggest one and the hardest one to leave due to perturbations. The average health is the smallest one so it is easier to leave it and its states are more similar to the states in the healthy BOA. Note that a state can reach all others due to perturbations and similar states require perturbations in a small amount of entities. Another interesting point is that starting from an average health it is better to not intervene (#Step$_{\mathcal{D}}$ = 48.3) than to apply an arbitrary policy (#Step$_{\mathcal{D}}$ = 45.1).

The frameworks proposed by (SOOTLA et al., 2014) and (SIRIN; POLAT; ALHAJJ, 2017) have a worse performance when the CBNp starts from an average health. Since

Table 23: Results for the human aging network when it starts from the unhealthy BOA. Numbers are average from 30 executions with the 95% confidence interval. Numbers in bold correspond to the best performance achieved in each metric.

Policy	#Steps$_{\mathcal{D}}$	#a^{ia}	$a^{ia}_{\mathcal{D}}$
Uncontrolled	17.6 ± 0.3	0.0 ± 0.0	0.0 ± 0.0
Arbitrary policy	48.0 ± 2.2	48.6 ± 2.1	23.5 ± 1.0
Algorithm	#Steps$_{\mathcal{D}}$	#a^{ia}	$a^{ia}_{\mathcal{D}}$
(SOOTLA et al., 2014)	**93.2 ± 0.2**	**1.5 ± 0.0**	**0.0 ± 0.0**
(SIRIN; POLAT; ALHAJJ, 2017)	62.2 ± 1.4	44.1 ± 0.9	21.2 ± 0.5
BOAConFE	92.7 ± 0.2	3.9 ± 0.1	1.0 ± 0.0

Table 24: Results for the human aging network when it starts from the average health BOA. Numbers are average from 30 executions with the 95% confidence interval. Numbers in bold correspond to the best performance achieved in each metric.

Policy	#Steps$_{\mathcal{D}}$	#a^{ia}	$a^{ia}_{\mathcal{D}}$
Non intervention policy	48.1 ± 0.4	0.0 ± 0.0	0.0 ± 0.0
Arbitrary policy	45.8 ± 2.2	49.1 ± 2.1	22.4 ± 0.9
Algorithm	#Steps$_{\mathcal{D}}$	#a^{ia}	$a^{ia}_{\mathcal{D}}$
(SOOTLA et al., 2014)	60.7 ± 0.4	**0.5 ± 0.0**	**0.0 ± 0.0**
(SIRIN; POLAT; ALHAJJ, 2017)	57.7 ± 1.1	43.7 ± 1.0	19.6 ± 0.5
BOAConFE	**92.8 ± 0.2**	3.9 ± 0.1	1.1 ± 0.0

#a^{ia} of (SOOTLA et al., 2014) is 0.5, this indicates that in half of the executions it does not applied a single intervention and in the other half it applies one intervention. #Steps$_{\mathcal{D}}$ is primary due to perturbations that made the CBNp leave the undesirable attractor and reach a desirable BOA and the single intervention it performed increased #Steps$_{\mathcal{D}}$ in comparison to non intervention policy. (SIRIN; POLAT; ALHAJJ, 2017) has the worst performance in all metrics, but it is still better than an arbitrary policy. BOACOnFE has the best result overall, it has a high #Steps and low #a^{ia} and #$a^{ia}_{\mathcal{D}}$ in both cases, which demonstrates its robustness. Although it does not have the lowest #a^{ia} and #$a^{ia}_{\mathcal{D}}$, the interventions allowed the CBNp to quickly reach and stay in $\mathcal{D}$, which is preferable than waiting a long time for a random perturbation.

5.8 Further discussion

The experimental evaluation conducted in this chapter sought to answer five main questions:

1. How do the results change when we increase the number of steps m in mSBOAFQI, given different degrees of observability (Section 5.2)?
2. How does the accuracy of the optimal and calculated $P(o \in \mathcal{D})$ affect the results (Section 5.3)?
3. Does the reward function affect the quality of a derived intervention policy (Section 5.4)?
4. Can an intervention policy reduce the number of interventions (Section 5.5)?
5. Can experience-based methods provide better results than those based on models in partially observable biological networks (Section 5.6)?
6. Can BOAConFE offer higher quality policies than previous proposals (Section 5.7)?

The experiments in Section 5.2 showed that increasing m improves results, given that the experience set size is large enough to learn a high quality policy. However, as observability increases the number of variables (observation-action pairs) increases, which highly increases the requirement of the experience set size. A specialist should choose m given the expected observability and the size of the experience set.

In order to answer question 2, the experiment in Section 5.3 compared the result of mSBOAFQI when the optimal $P(o \in \mathcal{D})$ is available against when it is computed. As expected, an optimal $P(o \in \mathcal{D})$ provides better results and when the computed $P(o \in \mathcal{D})$ gets closer to optimal values, results improve. These results indicate that mSBOAFQI is robust when $P(o \in \mathcal{D})$ is not optimal and that our proposed method (Algorithm 2) can calculate $P(o \in \mathcal{D})$ with probabilities close to the optimum when a large enough number of observations are available.

Section 5.4 answers question 3, where we evaluated different reward functions. Our proposed reward functions provided the best results, which indicates that taking into account current and next observations desirability provides more information regarding the experience. When the true rewards are not available, it is possible to compute the average an observation is expected to receive using $P(o \in \mathcal{D})$. Although

the results get worse when using the computed $P(o \in \mathcal{D})$, it is a valid alternative to automatically compute rewards when they are not available.

Question 4 is concerned with the importance of how the policy is computed. In Section 5.5, we evaluated different control policies: deterministic, conventional stochastic and BOA policy. Conventional stochastic policy has the worst result because it has the lowest number of #Steps$_{\mathcal{D}}$ while the highest #a^{ia}. One of the main reasons for these results is how the policy is computed with Equation 2.13. When the optimal action for an observation has a Q-Value that is close to the worst action, the probabilities are also close. For example, Q-values of 3.0 and 2.0 for the best and worst actions, respectively, generate probabilities 0.6 and 0.4 of applying them. If instead of a conventional stochastic policy it was used the Boltzmann policy (Equation 2.14), the probabilities would be 0.73 and 0.27. Therefore, the probability of applying the best action increases as the difference between its Q-Value and the other is greater.

A deterministic policy is the best alternative when an observation only represents states in (un)desirable BOAs. In this case, the computed policy usually indicates a non intervention action when the CBNp is in $\mathcal{D}$. On the other hand, when observations in the pathway or the desirable attractor represent states in $\mathcal{U}$, the derived policy may always indicate an intervention action when it must not. Other requirement for a deterministic policy is not causing a cycle that does not exist. For example, when applying an intervention in an observation that makes the CBNp go to the same observation, it forms a cycle caused by the intervention.

When it is not possible to guarantee the deterministic policy requirements, applying a BOA policy is a good alternative. It is tied with deterministic policy in #Steps$_{\mathcal{D}}$ and has the lowest #a^{ia}. The main issue with this policy is stochasticity, because it is difficult to decide which action to choose and apply in a patient. It would be unethical to flip a biased coin, so a physician should be responsible for looking at the recommendation and choose an action. Therefore, BOA policy is an alternative to a deterministic policy when its requirements cannot be met.

The applicability of experience-based methods to the intervention problem of partially observable BTs is investigated in question 5. Model-based methods are the main technique used to solve this problem, regardless of how accurate is the reconstructed

model. Hence, it is important to verify whether experience-based methods can provide better results in partially observable biological networks. In Section 5.6, we reconstructed a model from a time-series of observations, applied a model-based method to compute a deterministic policy from this model, and then we compared the computed policy to the BOAConFE one. As expected, the model-based approach provided worse results in average than BOAConFE. Because the reconstructed model is not accurate due to partial obsevability, model-based methods derive a control policy that is usually not appropriate to the true biological network.

For answering the last question we used synthetic and biological inspired CBNps in a realistic scenario. Neither the optimal $P(o \in \mathcal{D})$ nor the true rewards were available. BOAConFE computed the experience rewards for all frameworks, otherwise it would not be possible to use the frameworks proposed by Sirin, Polat and Alhajj (2017) and Sootla et al. (2014). BOAConFE provided the best result overall in all metrics. Although for some biological inspired CBNps BOAConFE has not the best result, it provided the second best and clearly a robust result for any situation. While Sirin, Polat and Alhajj (2017) usually has the worst number of steps in#Steps$_{\mathcal{D}}$ and Sootla et al. (2014) a very bad result in some cases (T-Helper cell), BOAConFE demonstrated a consistent strong result over all experiments.

All experiments indicate that BOAConFE can be used in most situations, mainly when there is partial observability. In this case, discovering the true rewards is a difficult task, which requires a specialist to analyze each experience. BOAConFE provides a solution for this situation by computing the average reward an observation is expected to have. Moreover, when observability is poor, mSBOAFQI and BOA policy can compute a higher quality policy than previous proposals.

6 FINAL CONSIDERATIONS

In this last chapter, we give a conclusion of the conducted research (Section 6.1) and a brief discussion of potential future work (Section 6.2).

6.1 Conclusion

In this work, we investigated machine learning techniques to automatically generate intervention policies that are capable of quickly shifting partially observable BTs from undesirable to desirable BOAs. A major goal of BT intervention problems is to discover how to better treat diseases and make patients get healthier. An effective treatment can quickly cure or lead diseases into remission by applying few interventions as possible, and without causing severe side effects. Control methods can support this endeavour by facilitating decision making and deriving intervention policies that define when and what interventions to apply in order to treat diseases. As biological pathways are essential for the progression of biological tasks, solutions that disrupt them are not applicable in real world; otherwise, other diseases could rise and a patient would stay unhealthy or even get sicker.

One of the challenges of this control problem is partial observability, because factors that influence a task may not be observed. Since part of the entities are not perceived, it is difficult to infer an accurate BT model that can describe the real BT dynamics. Computing an intervention policy from an inaccurate model can lead to poor results in a real BT as the inferred model and real BT have different behaviours. Instead of using an inferred model to generate a policy, it is possible to apply techniques that learn an intervention policy from experiences.

In this work, we proposed a new framework for intervening in partially observable BTs that learns an intervention policy from a batch of experiences given in advance. Our framework, BOAConFE, explores the concept of BOAs and use their structure to properly solve this control problem. BOAConFE does not use information regarding the BT and compute the probability of each possible observation being in desirable BOAs $\mathcal{D}$. These probabilities are used in other phases of our framework to incorporate the

concept of BOAs in partially observable BT. Experiences are composed of a sequence of steps and a reward and cost signal defining how well the control system performed. When a specialist cannot provide rewards due to partial observability, our framework can compute them based on the probability of observations being in $\mathcal{D}$. Our proposed BRL method, mSBOAFQI, uses past steps to deal with trajectories and differentiate observations that should not be treated as a single one. In addition, mSBOAFQI uses BOA probabilities to exploit BOA structure and improve the learning process. Finally, BOA policy reduces the number of interventions applied when the BT is already in $\mathcal{D}$ and further application are not necessary to improve the health of a BT.

We empirically demonstrated that BOAConFE can quickly shift a partially observable BT from $\mathcal{U}$ to $\mathcal{D}$. It is capable of increasing the number of steps in $\mathcal{D}$, while decreasing the number of interventions, which means more cost-effective treatments and less side effects. Our experiments showed that generating policies from inaccurate models provide poor results when applied in real BTs. Therefore, generating intervention policies directly from experiences is a robust option for solving the intervention problem in partially observable BTs. When observability is poor, mSBOAFQI produces higher quality polices when more steps are available in an experience. However, as observability increases, having more steps requires more experiences and the advantages decrease.

Using the real rewards provides better results, since they accurately describe the desirability of each experience, but when they are not available BOAConFE can automatically compute them and still derive high quality policies. We also demonstrated that BOA policy can reduce the amount of interventions applied in $\mathcal{D}$, while increasing the number of steps in $\mathcal{D}$. Hence, our framework can automatically generate higher quality intervention policies than previous proposals for partially observable BTs and without requiring information regarding the BT modelling.

6.2 Future work

Future work on this framework may focus on gradual improvements in the process or even on various biological aspects. The following is an incomplete list of suggestions that might be interesting to extend this work on:

- Entity activity levels are continuous values that represent how active an entity is at a given time. Although discretization allows to have a high-level overview of the biological task, solving the intervention problem using continuous values may be important when it is important to consider various doses of a remedy and their impact on the concentration of different entity products. Thus, an important extension of our proposal would be to extend the formulation to continuous values.

- Since mSBOAFQI considers a trajectory to deal with partial observability, $P(o \in \mathcal{D})$ could be extended to use a trajectory and provide more accurate probabilities. For example, if the current observation o may be representing a desirable attractor and the last steps were in this same attractor, it is probably an attractor cycle and $P(o \in \mathcal{D}|z,u)$ should be much higher than $P(o^t \in \mathcal{D})$.

- Interventions may take time to take effect, their effect may last for more than a period of time, or they may be ineffective. In addition, some interventions may not be applied for some period, for example, several chemotherapy sessions cannot be applied in short periods of time. BOAConFE can be extended to consider different characteristics of the interventions.

- Another interesting aspect would be to see how to adapt treatments for patients whose BT have differences. For example, patients may have different genetic variations or a medicine may be more or less effective to them, or they cannot take a particular therapy. A growing batch data set could adapt to this differences, rules could be combined with reinforcement learning, or transfer learning (TAYLOR; STONE, 2009) could be used to share expert knowledge to new patients.

- Yet another approach could be to improve the learning of $P(o \in \mathcal{D})$ by applying machine learning techniques, such as supervised learning.

Finally, this area of research is very important and presents numerous challenges. We believe that our proposal offers an advance in the solution of this relevant problem of controlling interventions in BTs.

REFERENCES

AKUTSU, T.; HAYASHIDA, M.; CHING, W.-K.; NG, M. K. Control of Boolean networks: Hardness results and algorithms for tree structured networks. **Journal of Theoretical Biology**, Academic Press, v. 244, n. 4, p. 670–679, 2 2007. ISSN 0022-5193.

AKUTSU, T.; ZHAO, Y.; HAYASHIDA, M.; TAMURA, T. Integer Programming-Based Approach to Attractor Detection and Control of Boolean Networks. **IEICE Transactions on Information and Systems**, The Institute of Electronics, Information and Communication Engineers, E95.D, n. 12, p. 2960–2970, 12 2012. ISSN 0916-8532.

ANTON, S. D.; MOEHL, K.; DONAHOO, W. T.; MAROSI, K.; LEE, S. A. et al. Flipping the metabolic switch: Understanding and applying the health benefits of fasting. **Obesity**, Wiley Online Library, v. 26, n. 2, p. 254–268, 2 2018. ISSN 1930739X.

ARNOLD, S. E.; ARVANITAKIS, Z.; MACAULEY-RAMBACH, S. L.; KOENIG, A. M.; WANG, H.-Y. et al. Brain insulin resistance in type 2 diabetes and Alzheimer disease: concepts and conundrums. **Nature Reviews Neurology**, Nature Publishing Group, v. 14, n. 3, p. 168–181, 3 2018. ISSN 1759-4758.

BELLMAN, R. A Markovian decision process. **Journal of Mathematics and Mechanics**, JSTOR, p. 679–684, 1957.

BERTSEKAS, D. P. **Dynamic programming and optimal control**: Athena scientific Belmont, 1995. ISBN 1886529434.

BITTNER, M.; MELTZER, P.; CHEN, Y.; JIANG, Y.; SEFTOR, E. et al. Molecular classification of cutaneous malignant melanoma by gene expression profiling. **Nature**, Nature Publishing Group, v. 406, p. 536–540, 8 2000.

BOUAYNAYA, N.; SHTERENBERG, R.; SCHONFELD, D. Inverse perturbation for optimal intervention in gene regulatory networks. **Bioinformatics**, Oxford University Press, v. 27, n. 1, p. 103–110, 1 2011. ISSN 1367-4803.

BRAHMER, J. R.; TYKODI, S. S.; CHOW, L. Q. M.; HWU, W.-J.; TOPALIAN, S. L. et al. Safety and activity of anti–PD-L1 antibody in patients with advanced cancer. **New England Journal of Medicine**, Mass Medical Soc, v. 2012, n. 366, p. 2455–2465, 2012.

BREIMAN, L. Random Forests. **Machine Learning**, Springer, v. 45, n. 1, p. 5–32, 2001. ISSN 08856125.

BRYCE, D.; VERDICCHIO, M.; KIM, S. Planning interventions in biological networks. **ACM Transactions on Intelligent Systems and Technology (TIST)**, ACM, v. 1, n. 2, p. 11, 2010.

BUSONIU, L.; BABUSKA, R.; SCHUTTER, B. D.; ERNST, D. **Reinforcement learning and dynamic programming using function approximators**: CRC press, 2010. v. 39.

CAMACHO, D. M.; COLLINS, K. M.; POWERS, R. K.; COSTELLO, J. C.; COLLINS, J. J. Next-Generation Machine Learning for Biological Networks. **Cell**, Cell Press, v. 173, n. 7, p. 1581–1592, 6 2018. ISSN 0092-8674.

CAMAZINE, S.; DENEUBOURG, J.-L.; FRANKS, N. R.; SNEYD, J.; BONABEAU, E. et al. **Self-organization in biological systems**: Princeton University Press, 2001. v. 7. ISBN 9780691116242.

CHOUDHARY, A.; DATTA, A.; BITTNER, M. L.; DOUGHERTY, E. R. Intervention in a family of Boolean networks. **Bioinformatics**, v. 22, n. 2, p. 226–232, 1 2006. ISSN 1367-4803.

DATTA, A.; CHOUDHARY, A.; BITTNER, M. L.; DOUGHERTY, E. R. External Control in Markovian Genetic Regulatory Networks. **Machine Learning**, Springer, v. 52, n. 1, p. 169–191, 7 2003.

____. External control in Markovian genetic regulatory networks: the imperfect information case. **Bioinformatics**, Oxford University Press, v. 20, n. 6, p. 924–930, 4 2004. ISSN 1367-4803.

ELOWITZ, M. B.; LEVINE, A. J.; SIGGIA, E. D.; SWAIN, P. S. Stochastic gene expression in a single cell. **Science**, American Association for the Advancement of Science, v. 297, n. 5584, p. 1183–6, 8 2002. ISSN 1095-9203.

ENGELHARDT, D. Dynamic Control of Stochastic Evolution: A Deep Reinforcement Learning Approach to Adaptively Targeting Emergent Drug Resistance. **arXiv preprint arXiv:1903.11373**, 3 2019. Available from Internet: <http://arxiv.org/abs/1903.11373>.

ENGLER, R.; ROUTH, T. L.; LUCISANO, J. Y. Adoption barriers for continuous glucose monitoring and their potential reduction with a fully implanted system: results from patient preference surveys. **Clinical Diabetes**, American Diabetes Association, v. 36, n. 1, p. 50–58, 1 2018. ISSN 0891-8929.

ERDOGDU, U.; POLAT, F.; ALHAJJ, R. Partially Observable Gene Regulatory Network Control without a Boundary on Horizon. In: IEEE. **2012 IEEE 24th International Conference on Tools with Artificial Intelligence**, 2012. v. 1, p. 81–88.

____. Employing decomposable partially observable Markov decision processes to control gene regulatory networks. **Artificial Intelligence in Medicine**, v. 83, p. 14–34, 2017. ISSN 0933-3657.

ERNST, D.; GEURTS, P.; WEHENKEL, L. Tree-based batch mode reinforcement learning. **Journal of Machine Learning Research**, v. 6, p. 503–556, 2005.

FARYABI, B.; DATTA, A.; DOUGHERTY, E. R. On approximate stochastic control in genetic regulatory networks. **IET Systems Biology**, IET, v. 1, n. 6, p. 361–368, 2007.

FUMIÃ, H. F.; MARTINS, M. L. Boolean Network Model for Cancer Pathways: Predicting Carcinogenesis and Targeted Therapy Outcomes. **PLOS ONE**, Public Library of Science, v. 8, n. 7, p. e69008–, 7 2013.

FURBER, J. D. **Systems Biology of Human Aging - Network Model Wall Chart. (Rev 28 May 2019)**. 2019. Available from Internet: <http://www.LegendaryPharma.com/chartbg.html>.

GALLO, C. A.; CECCHINI, R. L.; CARBALLIDO, J. A.; MICHELETTO, S.; PONZONI, I. Discretization of gene expression data revised. **Briefings in bioinformatics**, Oxford University Press, v. 17, n. 5, p. 758–770, 2015.

GARCIA, S.; LUENGO, J.; SÁEZ, J. A.; LÓPEZ, V.; HERRERA, F. A Survey of Discretization Techniques: Taxonomy and Empirical Analysis in Supervised Learning. **IEEE Transactions on Knowledge and Data Engineering**, v. 25, n. 4, p. 734–750, 4 2013. ISSN 1041-4347.

GENIN, E.; HANNEQUIN, D.; WALLON, D.; SLEEGERS, K.; HILTUNEN, M. et al. APOE and Alzheimer disease: a major gene with semi-dominant inheritance. **Molecular Psychiatry**, Nature Publishing Group, v. 16, n. 9, p. 903–907, 9 2011. ISSN 1359-4184.

GROSS, F. What systems biology can tell us about disease. **History and philosophy of the life sciences**, JSTOR, p. 477–496, 2011.

HAYASHIDA, M.; TAMURA, T.; AKUTSU, T.; ZHANG, S.-Q.; CHING, W.-K. Algorithms and Complexity Analyses for Control of Singleton Attractors in Boolean Networks. **EURASIP Journal on Bioinformatics and Systems Biology**, Springer International Publishing, v. 2008, n. 1, p. 1–16, 2008. ISSN 1687-4145.

HEPPNER, F. L.; RANSOHOFF, R. M.; BECHER, B. Immune attack: the role of inflammation in Alzheimer disease. **Nature Reviews Neuroscience**, Nature Publishing Group, v. 16, n. 6, p. 358–372, 6 2015. ISSN 1471-003X.

HIRAISHI, K.; KOBAYASHI, K. Symbolic approach to verification and control of deterministic/probabilistic Boolean networks. **IET Systems Biology**, v. 6, n. 6, p. 215–222, 12 2012. ISSN 1751-8849.

HOLMES, C.; CUNNINGHAM, C.; ZOTOVA, E.; WOOLFORD, J.; DEAN, C. et al. Systemic inflammation and disease progression in Alzheimer disease. **Neurology**, American Academy of Neurology, v. 73, n. 10, p. 768–74, 9 2009. ISSN 1526-632X.

HOPFENSITZ, M.; MUSSEL, C.; WAWRA, C.; MAUCHER, M.; KUHL, M. et al. Multiscale Binarization of Gene Expression Data for Reconstructing Boolean Networks. **IEEE/ACM Transactions on Computational Biology and Bioinformatics**, IEEE, v. 9, n. 2, p. 487–498, 2012. ISSN 1545-5963.

HU, M.; SHEN, L.; ZAN, X.; SHANG, X.; LIU, W. An efficient algorithm to identify the optimal one-bit perturbation based on the basin-of-state size of Boolean networks. **Scientific Reports**, Nature Publishing Group, v. 6, p. 26247, 2016. ISSN 2045-2322.

IMANI, M.; BRAGA-NETO, U. M. Control of gene regulatory networks with noisy measurements and uncertain inputs. **IEEE Transactions on Control of Network Systems**, IEEE, v. 5, n. 2, p. 760–769, 2018.

____. Point-based methodology to monitor and control gene regulatory networks via noisy measurements. **IEEE Transactions on Control Systems Technology**, IEEE, v. 27, n. 3, p. 1023–1035, 2018.

IVANOV, I.; PAL, R.; DOUGHERTY, E. R. Dynamics Preserving Size Reduction Mappings for Probabilistic Boolean Networks. **IEEE Transactions on Signal Processing**, v. 55, n. 5, p. 2310–2322, 5 2007. ISSN 1053-587X.

JAAKKOLA, T.; SINGH, S. P.; JORDAN, M. I. Reinforcement learning algorithm for partially observable Markov decision problemst Importer. In: **Proceedings of the 7th International Conference on Neural Information Processing Systems**, 1995. p. 345–352.

KARLEBACH, G.; SHAMIR, R. Modelling and analysis of gene regulatory networks. **Nature Reviews Molecular Cell Biology**, Nature Publishing Group, v. 9, n. 10, p. 770–780, 2008.

KAUFFMAN, S. A. Metabolic stability and epigenesis in randomly constructed genetic nets. **Journal of theoretical biology**, Elsevier, v. 22, n. 3, p. 437–467, 1969.

____. The origins of order: Self-organization and selection in evolution. In: : Oxford University Press, USA, 1992. p. 61–100.

KOBAYASHI, K.; HIRAISHI, K. Optimization-Based Approaches to Control of Probabilistic Boolean Networks. **Algorithms**, Multidisciplinary Digital Publishing Institute, v. 10, n. 1, p. 31, 2 2017. ISSN 1999-4893.

LANGE, S.; GABEL, T.; RIEDMILLER, M. Batch reinforcement learning. In: **Reinforcement learning**: Springer, 2012. p. 45–73.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 5 2015. ISSN 0028-0836.

LI, F.; LONG, T.; LU, Y.; OUYANG, Q.; TANG, C. The yeast cell-cycle network is robustly designed. **Proceedings of the National Academy of Sciences of the United States of America**, National Academy of Sciences, v. 101, n. 14, p. 4781–4786, 2004.

LONGO, V. D.; MATTSON, M. P. Fasting: Molecular Mechanisms and Clinical Applications. **Cell Metabolism**, Cell Press, v. 19, n. 2, p. 181–192, 2 2014. ISSN 1550-4131.

MAHANTA, P.; AHMED, H. A.; KALITA, J. K.; BHATTACHARYYA, D. K. Discretization in gene expression data analysis: a selected survey. In: **Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology**, 2012. p. 69–75.

MENDOZA, L.; XENARIOS, I. A method for the generation of standardized qualitative dynamical systems of regulatory networks. **Theoretical Biology and Medical Modelling**, v. 3, n. 1, p. 13, 3 2006. ISSN 17424682.

MONTE, S. M. de la. Insulin resistance and Alzheimer's disease. **BMB reports**, v. 42, n. 8, p. 475–81, 8 2009. ISSN 1976-6696.

NADERI, R.; MOZAYANI, N. Gene Regulatory Networks Full Observable Cbased on Batch Reinforcement Learning: An Improved Policy. In: **2019 27th Iranian Conference on Electrical Engineering (ICEE)**: IEEE, 2019. p. 2004–2009. ISBN 978-1-7281-1508-5.

NELSON, D. L.; COX, M. M.; LEHNINGER, A. L. **Lehninger principles of biochemistry.**: WH Freeman, 2008.

NISHIDA, C. E. H.; BIANCHI, R. A. C.; COSTA, A. H. R. Batch Reinforcement Learning Stochastic Policy to shift Basin of Attractions in Partially Observable Biological Systems. In: **Workshop on Computational Biology 2018 Proceedings, Stockholm, Sweden**, 2018. Available from Internet: <https://drive.google.com/file/d/1mQb8gnDan0Yu7yqDo5xmxwDxi7FH9I6v/view?usp=sharing>.

NISHIDA, C. E. H.; BIANCHI, R. A. C.; COSTA, A. H. R. C. A framework to shift basins of attraction of gene regulatory networks through batch reinforcement learning. **Artificial Intelligence in Medicine**, Elsevier, v. 107, p. 101853, 2020.

NISHIDA, C. E. H.; COSTA, A. H. R. Fitted Q-Iteration Fatorado no controle de Redes de Regulação Gênicas. In: **Symposium on Knowledge Discovery, Mining and Learning (KDMiLe)**, 2016. p. 218–225.

NISHIDA, C. E. H.; COSTA, A. H. R.; BIANCHI, R. A. d. C. Controlling Gene Regulatory Networks with FQI-Sarsa. In: **2017 6th Brazilian Conference on Intelligent Systems (BRACIS)**, 2017. p. 216–221.

____. Control of Gene Regulatory Networks Basin of Attractions with Batch Reinforcement Learning. In: **2018 7th Brazilian Conference on Intelligent Systems (BRACIS)**, 2018. p. 127–132.

ORGANIZATION, W. H. **Guidelines for the treatment of malaria**: World Health Organization, 2015.

OWEN, L. A.; HARTNETT, M. E. Current concepts of oxygen management in retinopathy of prematurity. **Journal of ophthalmic & vision research**, v. 9, n. 1, p. 94–100, 1 2014. ISSN 2008-2010.

PAL, R.; DATTA, A.; DOUGHERTY, E. R. Optimal infinite-horizon control for probabilistic Boolean networks. **IEEE Transactions on Signal Processing**, IEEE, v. 54, n. 6, p. 2375–2387, 2006.

PAPAGIANNIS, G.; MOSCHOYIANNIS, S. Learning to control random boolean networks: A deep reinforcement learning approach. In: SPRINGER. **International Conference on Complex Networks and Their Applications**, 2019. p. 721–734.

____. Deep reinforcement learning for control of probabilistic boolean networks. **arXiv preprint arXiv:1909.03331**, 2020.

PEDNAULT, E.; ABE, N.; ZADROZNY, B. Sequential cost-sensitive decision making with reinforcement learning. In: **Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining**, 2002. p. 259–268.

PETERSEN, B. K.; YANG, J.; GRATHWOHL, W. S.; COCKRELL, C.; SANTIAGO, C. et al. Deep Reinforcement Learning and Simulation as a Path Toward Precision Medicine. **Journal of Computational Biology**, v. 26, n. 6, p. 597–604, 6 2019. ISSN 1557-8666.

PINO, M. S.; CHUNG, D. C. The chromosomal instability pathway in colon cancer. **Gastroenterology**, Elsevier, v. 138, n. 6, p. 2059–2072, 2010.

QIU, Y.; TAMURA, T.; CHING, W.-K.; AKUTSU, T. On control of singleton attractors in multiple Boolean networks: integer programming-based method. **BMC Systems Biology**, v. 8, n. 1, p. S7, 2014. ISSN 1752-0509.

RUMMERY, G. A.; NIRANJAN, M. **On-line Q-learning using connectionist systems**: University of Cambridge, Department of Engineering, 1994. v. 37.

SAWYERS, C. Targeted cancer therapy. **Nature**, Nature Publishing Group, v. 432, n. 7015, p. 294–297, 2004.

SHMULEVICH, I.; DOUGHERTY, E. R. **Probabilistic Boolean networks: the modeling and control of gene regulatory networks**: SIAM, 2010.

SHMULEVICH, I.; DOUGHERTY, E. R.; KIM, S.; ZHANG, W. Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks. **Bioinformatics**, Oxford University Press, v. 18, n. 2, p. 261–274, 2002.

SHMULEVICH, I.; DOUGHERTY, E. R.; ZHANG, W. Control of stationary behavior in probabilistic Boolean networks by means of structural intervention. **Journal of Biological Systems**, v. 10, n. 04, p. 431–445, 2002.

____. Gene perturbation and intervention in probabilistic Boolean networks. **Bioinformatics**, Oxford University Press, v. 18, n. 10, p. 1319–1331, 2002.

SHMULEVICH, I.; SAARINEN, A.; YLI-HARJA, O.; ASTOLA, J. Inference of genetic regulatory networks via best-fit extensions. **Computational and Statistical Approaches to Genomics**, Springer, p. 197–210, 2003.

SILVA, F. L. D.; NISHIDA, C. E.; ROIJERS, D. M.; COSTA, A. H. R. Coordination of electric vehicle charging through multiagent reinforcement learning. **IEEE Transactions on Smart Grid**, IEEE, 2019.

SILVERMAN, W. A. A cautionary tale about supplemental oxygen: the albatross of neonatal medicine. **Pediatrics**, American Academy of Pediatrics, v. 113, n. 2, p. 394–6, 2 2004. ISSN 1098-4275.

SIRIN, U.; POLAT, F.; ALHAJJ, R. Employing batch reinforcement learning to control gene regulation without explicitly constructing gene regulatory networks. In: **Proceedings of the 23rd International Joint Conference on Artificial Intelligence**, 2013. p. 2042–2048.

____. Batch Mode TD(λ) for Controlling Partially Observable Gene Regulatory Networks. **IEEE/ACM transactions on computational biology and bioinformatics**, IEEE, v. 14, n. 6, p. 1214–1227, 2017.

SOOTLA, A.; STRELKOWA, N.; ERNST, D.; BARAHONA, M.; STAN, G.-B. On periodic reference tracking using batch-mode reinforcement learning with application to gene regulatory network control. In: **Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on**, 2013. p. 4086–4091.

SOOTLA, A.; STRELKOWA, N.; ERNST, D.; BARAHONA, M.; GUY-BART, S. Toggling a genetic switch using reinforcement learning. In: **Proceedings of the 9th French Meeting on Planning, Decision Making and Learning**, 2014.

STRITTMATTER, W. J. Old Drug, New Hope for Alzheimer's Disease. **Science**, v. 335, n. 6075, 2012.

SUTTON, R. S.; BARTO, A. G. **Reinforcement learning : an introduction**: MIT press, 2018. ISBN 9780262039246.

TAN, M.; ALHAJJ, R.; POLAT, F. Automated large-scale control of gene regulatory networks. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, IEEE, v. 40, n. 2, p. 286–297, 2010.

____. Scalable approach for effective control of gene regulatory networks. **Artificial Intelligence in Medicine**, Elsevier, v. 48, n. 1, p. 51–59, 1 2010. ISSN 0933-3657.

TAOU, N. S.; CORNE, D. W.; LONES, M. A. Investigating the use of Boolean networks for the control of gene regulatory networks. **Journal of Computational Science**, Elsevier, v. 26, p. 147–156, 5 2018. ISSN 1877-7503.

TAYLOR, M. E.; STONE, P. Transfer learning for reinforcement learning domains: A survey. **Journal of Machine Learning Research**, JMLR.org, v. 10, p. 1633–1685, 2009.

TISOVEC, F. A. C.; BARROS, L. N. d.; DELGADO, K. V.; SILVA, C. F. d.; HASHIMOTO, R. F. Robust intervention on genetic regulatory networks using symbolic dynamic programming. In: **Workshop of Bioinformatics and Artificial Intelligence (BAI), Buenos Aires, Argentina**, 2015. Available from Internet: <https://bdpi.usp.br/item/002854449>.

VERDICCHIO, M. P.; KIM, S. Identifying targets for intervention by analyzing basins of attraction. **Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing**, p. 350–61, 2011. ISSN 2335-6936.

WANG, L.; FENG, T.; SONG, J.; GUO, Z.; HU, J. Model Checking Optimal Infinite-Horizon Control for Probabilistic Gene Regulatory Networks. **IEEE Access**, v. 6, p. 77299–77307, 2018. ISSN 2169-3536.

WATKINS, C. J. C. H.; DAYAN, P. Q-learning. **Machine learning**, Springer, v. 8, n. 3-4, p. 279–292, 1992.

WEERARATNA, A. T.; JIANG, Y.; HOSTETTER, G.; ROSENBLATT, K.; DURAY, P. et al. Wnt5a signaling directly affects cell motility and invasion of metastatic melanoma. **Cancer Cell**, Cell Press, v. 1, n. 3, p. 279–288, 4 2002.

WILLETTE, A. A.; BENDLIN, B. B.; STARKS, E. J.; BIRDSILL, A. C.; JOHNSON, S. C. et al. Association of Insulin Resistance With Cerebral Glucose Uptake in Late Middle–Aged Adults at Risk for Alzheimer Disease. **JAMA Neurology**, American Medical Association, v. 72, n. 9, p. 1013, 9 2015. ISSN 2168-6149.

YOUSEFI, M. R.; DOUGHERTY, E. R. Intervention in gene regulatory networks with maximal phenotype alteration. **Bioinformatics**, Oxford University Press, v. 29, n. 14, p. 1758–1767, 2013.

ZHANG, S.-Q.; HAYASHIDA, M.; AKUTSU, T.; CHING, W.-K.; NG, M. K. Algorithms for Finding Small Attractors in Boolean Networks. **EURASIP Journal on Bioinformatics and Systems Biology**, Hindawi Publishing Corp., v. 2007, p. 1–13, 2007. ISSN 1687-4145.

ZHANG, Y.; QIAN, M.; OUYANG, Q.; DENG, M.; LI, F. et al. Stochastic model of yeast cell-cycle network. **Physica D: Nonlinear Phenomena**, North-Holland, v. 219, n. 1, p. 35–39, 7 2006. ISSN 0167-2789.

ZHOU, X.; WANG, X.; DOUGHERTY, E. R. Binarization of Microarray Data on the Basis of a Mixture Model. **Molecular Cancer Therapeutics**, v. 2, n. 7, p. 679, 7 2003.

ZOLEDZIEWSKA, M.; COSTA, G.; PITZALIS, M.; COCCO, E.; MELIS, C. et al. Variation within the CLEC16A gene shows consistent disease association with both multiple sclerosis and type 1 diabetes in Sardinia. **Genes and immunity**, Nature Publishing Group, v. 10, n. 1, p. 15–17, 2009.

APPENDIX A – EXPERIENCE ASSEMBLY

Therapies that seek to treat pathway dysfunctions emerged in the last years mainly for treating cancers (SAWYERS, 2004). Several of these targeted therapies inhibit products produced by a targeted entity, reducing the availability of these products and consequently making this particular entity act as inhibited in the biological network (BT). These targeted entities, whose activity level is controlled by external influences such as therapies, are called control entities in the BT intervention problem (SHMULEVICH; DOUGHERTY, 2010). An example of a targeted therapy is the drug Nivolumab mentioned in Introduction (Chapter 1) that inhibits the immune system checkpoint to treat some types of cancer.

For modelling how targeted therapies works in a BT, the conventional approach is to consider that a therapy change the value of targeted entities and the new values are instantly used to compute the next state (SHMULEVICH; DOUGHERTY, 2010). For example, Table 1 shows the prediction function F of a BNp composed of 3 entities x_1, x_2 and x_3, for convenience of the reader, we repeat it in Table 25. Given a therapy inhibiting x_3, in a single step x_3 is inhibited and its new value $x_3 = 0$ is used to compute the next state. In this case, after applying an intervention in state 011, the third entity is inhibited and state 011 turns to 010, which is immediately used to compute the next state 101. This means that the result of applying an intervention in state 011 is a transition to 101, while without an intervention the BT would go to 000. On the other hand, when the system is in state 010 and an intervention is applied, x_3 is already inhibited, so the intervention action does not affect the BNp and it goes to state 101 as it would go without interventions. After applying this concept to all state, it results in the prediction functions shown in Table 26.

Definition 10. *Being a BNp given by Definition 5, composed of g entities and control*

Table 25: Example of regulatory functions with 3 entities for action non intervention

State	x_1	x_2	x_3	$f^{(1)}$	$f^{(2)}$	$f^{(3)}$
0	0	0	0	0	0	0
1	0	0	1	1	0	0
2	0	1	0	1	0	1
3	0	1	1	0	0	0
4	1	0	0	0	1	1
5	1	0	1	1	1	1
6	1	1	0	1	0	1
7	1	1	1	1	1	1

Table 26: Example of regulatory functions with 3 entities for intervention inhibiting x_3.

State	x_1	x_2	x_3	$f^{(1)}$	$f^{(2)}$	$f^{(3)}$
0	0	0	0	0	0	0
1	0	0	1	0	0	0
2	0	1	0	1	0	1
3	0	1	1	1	0	1
4	1	0	0	0	1	1
5	1	0	1	0	1	1
6	1	1	0	1	0	1
7	1	1	1	1	0	1

entity x_c, a prediction function $f^{(i)}$ of an intervention inhibiting x_c is given by:

$$f^{(i)}(x_1, .., x_c, .., x_g) = f^{(i)}(x_1, .., 0, .., x_g) \tag{A.1}$$

and an intervention action activating x_c is given by:

$$f^{(i)}(x_1, .., x_c, .., x_g) = f^{(i)}(x_1, .., 1, .., x_g) \tag{A.2}$$

A consequence of Definition 10 is that the transition probabilities for an intervention function inhibiting or activating a control entity can be generated from the transition probabilities of a non intervention action. The probability of going from state s to s' after applying an intervention action is equal to the probability of going from s'' to state s' after applying an intervention that transforms s into s'':

$$T(s, \text{intervention}, s') = T(s'', \text{non intervention}, s'). \tag{A.3}$$

For example, T(011, x_3 inhibition, 101) = T(010, non intervention, 101).

Most time-series entity activity data sets measure the evolution of entity activity when a particular biological function is being executed over a period of time. The idea is to understand how entities interact when performing the function, which allows to build models that explain these interactions. However, since no intervention actions are applied, this data set cannot be used in batch reinforcement learning (BRL) methods.

In order to overcome this situation, Sirin, Polat and Alhajj (2013) proposed an experience assembly method to build experiences (s, a, r, s') from a time series data set with N states $(s^1, .., s^t, .., s^N)$ collected without applying interventions. States s and s' are given by the data set, which are a state and its next state, respectively. For defining an action composed of n_c control entities $\boldsymbol{a} = (a_{i_1}, ..., a_{i_{nc}})$, they consider that a_{i_j} can change the value of entity x_{i_j}. For each control entity x_{i_j}, their method checks whether the value of x_{i_j} flipped from s to s':

$$a_{i_j} = \begin{cases} \textit{non intervention}, & \text{if } x_{i_j} = x'_{i_j} \\ x_{i_j}\textit{inhibition}, & \text{if } x_{i_j} \neq x'_{i_j} \text{ and } x'_{i_j} = 0 \\ x_{i_j}\textit{activation}, & \text{otherwise.} \end{cases}$$

Note that in their proposal, the next value of an control entity x_{i_j} is defined by action a_{i_j}:

$$x'_{i_j} = \begin{cases} x_{i_j}, & \text{if } a_{i_j} = \textit{non intervention} \\ 0, & \text{if } a_{i_j} = x_{i_j}\textit{inhibition} \\ 1, & \text{if } a_{i_j} = x_{i_j}\textit{activation}. \end{cases}$$

A.1 Proposed experience assemble method

The method proposed by (SIRIN; POLAT; ALHAJJ, 2013) considers that a non intervention action makes a control entity keep the same value between time steps, an inhibiting intervention inhibits the value of this entity in the next time step and an activation activates it in the next time step. This means that an entity is always being controlled by an external factor, because not necessarily a_{i_j} = *non intervention* keeps the same value between time steps. For example, in Table 25 and current state 010, the next state is 101, so regardless of the control entity, a non intervention action does not necessarily keep the value of the entities between time steps. In this case, it is

necessary to apply an appropriate therapy that is capable of maintaining the same value.

We propose a new experience assembly method that explores the control entity concept explained in Equation A.3 in its computation. It requires a time-series with N states $(s^1, .., s^t, .., s^N)$ collected without applying interventions and a set of n_c entities $\{x_{i_1}, ..., x_{i_{n_c}}\}$ that can be controlled by external stimuli. We consider an action $\boldsymbol{a} = (a_{i_1}, ..., a_{i_{n_c}})$, in which a_{i_j} can either be {*non intervention*, x_{i_j} *inhibition*, x_{i_j} *activation* }. A *non intervention* action does not change the value of x_{i_j}, an inhibition action turns x_{i_j} to 0 and an activation action to 1. This process is repeated for all control entities and the new value of all entities are used to compute the next state.

Algorithm 5 shows our proposed method. For illustrating how it works, Figure 18 shows an example with a sequence of three states (010, 101, 111) and x_3 as control entity. The first state in the sequence 010 is s and its next state 101 is s'. This sequence was collected without applying intervention actions, so state 010 goes to 101 when applying a *non intervention* action, which results in experience $(s, a, r, c, s') = (010, \textit{non intervention}, r_1, c_1, 101)$. Using Definition 10, applying action x_3 *inhibition* in state s''=011 it turns to state 010 and then 010 goes to state 101, which results in experience $(s'', a, r, c, s') = (011, x_3 \textit{ inhibition}, r_1, c_1, 101)$. Action x_3 *activation* cannot turn any s'' into 010 because the third entity is inhibited, so this action is skipped. The second state in the sequence 101 is s and its next state 111 is s' when applying a *non intervention* action, resulting in experience $(s, a, r, c, s') = (101, \textit{non intervention}, r_3, c_3, 111)$. Using Definition 10, applying action x_3 *activation* in state s''=100 it turns into state 101 and then 101 goes to state 111, resulting in experience $(s'', a, r, c, s') = (100, x_3 \textit{activation}, r_4, c_4, 111)$. Action x_3 *inhibition* cannot turn any s'' into 101 because the third entity is active, so this action is skipped.

After assembling the experience set $\mathscr{F}$, rewards should be assigned to each experience (see Section 4.2). Then, the experience set $\mathscr{F}$ is complete and can be used by BRL algorithms to derive a control policy. However, Algorithm 5 is not generic because it cannot be used to generate experiences with multiple steps.

Algorithm 5: Experience Assemble

Input : Time-series data set composed of discrete entity activity levels $\{s^1, ..., s^N\}$, number of measurements N, control entities $\{s_{i_1}, ..., s_{i_{n_c}}\}$, number of control entities n_c.

$\mathscr{F} = \emptyset$;
for $t = 1, 2, ..., N-1$ **do**
  $s' = s^{t+1}$;
  foreach $\boldsymbol{a} \in$ *Power set of (non intervention, inhibit, activate)*^{n_c} **do**
    $s = s^t$;
    /* Checks if action has no effect for a control entity. */
    if $(a_{i_j} ==$ *inhibit AND* $0 == x_{i_j})$ *OR* $(a_{i_j} ==$ *activate AND* $1 == x_{i_j})$ *OR* $(a_{i_j} ==$ *non intervention*$) \; \forall i_j$ **then**
      foreach $a_{i_j} \neq$ *non intervention* **do**
        $x_{i_j} = \overline{x_{i_j}}$ /* Flips entity value */
      end
      $\mathscr{F} = \mathscr{F} \cup (s, \boldsymbol{a}, 0, 0, s')$;
    end
  end
end
Output: $\mathscr{F}$

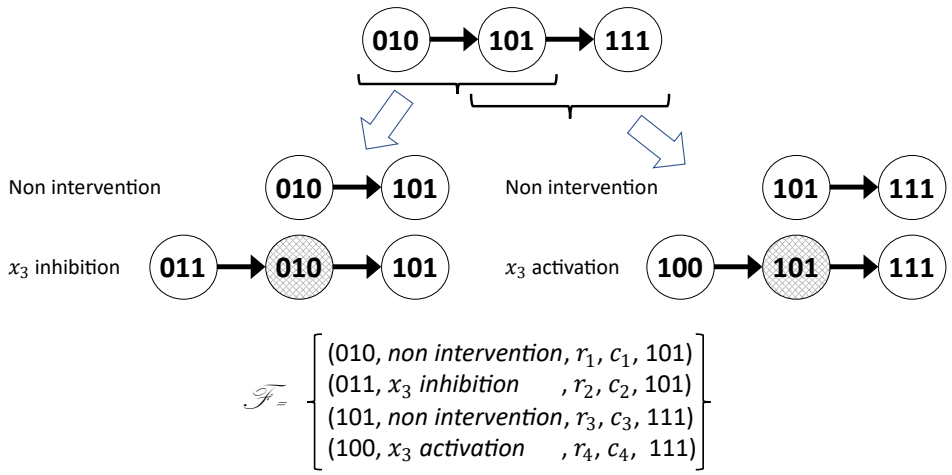


Figure 18: Example of how Algorithm 5 works with a sequence of 3 states = (010, 101, 111) and x_3 as control entity.

APPENDIX B – ACADEMIC ACHIEVEMENTS

This section describes the academic achievements and conducted activities during the course of the Ph.D. program.

B.1 Publications

The following list indicates the research productivity during the Ph.D. program:

A framework to shift basins of attraction of gene regulatory networks through batch reinforcement learning. Nishida, Bianchi and Costa (2020), In: Artificial Intelligence in Medicine, Elsevier, 2020. DOI: https://doi.org/10.1016/j.artmed.2020.101853.

Coordination of Electric Vehicle Charging through Multiagent Reinforcement Learning. Silva et al. (2019), In: IEEE Transactions on Smart Grid, IEEE, 2019. DOI: 10.1109/TSG.2019.2952331.

Control of Gene Regulatory Networks Basin of Attractions with Batch Reinforcement Learning. Nishida, Costa and Bianchi (2018), In: 2018 7th Brazilian Conference on Intelligent Systems(BRACIS). São Paulo, Brazil, 2018. IEEE. DOI: 10.1109/BRACIS.2018.00030.

Batch Reinforcement Learning Stochastic Policy to shift Basin of Attractions in Partially Observable Biological Systems. Nishida, Bianchi and Costa (2018), In:Joint ICML and IJCAI 2018 Workshop on Computational Biology Proceedings. Stockholm, Sweden, 2018.

Controlling Gene Regulatory Networks with FQI-Sarsa. Nishida, Costa and Bianchi (2017), In:2017 Brazilian Conference on Intelligent Systems (BRACIS). Uberlândia, Brazil, 2017. IEEE. DOI: 10.1109/BRACIS.2017.81.

Fitted Q-Iteration Fatorado no Controle de Redes de Regulação Gênicas. Nishida and Costa (2016), In:Symposium on Knowledge Discovery, Mining and Learning (KDMiLe). Recife, Brazil, 2016.

B.2 Disciplines

The minimum amount of credits (80) for the conclusion of the Ph.D. program has been acquired in full by attending twelve disciplines with a total of 82 credits and an average grade of A.

- **PCS5012-5/1 - Metodologia de Pesquisa Científica em Engenharia de Computação**. Workload: 120h, Credits: 8, **Grade: A**.
- **PCS5024-1/2 - Aprendizado Estatístico**. Workload: 120h, Credits: 8, **Grade: A**.
- **MAC5726-4/1 Biologia Computacional**. Workload: 120h, Credits: 8, **Grade: A**.
- **PSI5796-3/3 Processamento e Análise de Imagens e Vídeos**. Workload: 120h, Credits: 8, **Grade: A**.
- **MAC5727-5/1 Algoritmos de Aproximação**. Workload: 120h, Credits: 8, **Grade: C**.
- **MAC5784-2/2 Inteligência Artificial em Jogos de Computador**. Workload: 120h, Credits: 8, **Grade: A**.
- **MAC5912-2/2 Introdução a Redes Booleanas Probabilísticas**. Workload: 120h, Credits: 8, **Grade: A**.
- **PEE5000-5/1 Tópicos Avanąados em Engenharia Elétrica**. Workload: 30h, Credits: 2, **Grade: A**.
- **PCS5708-5/4 Tćnicas de Raciocínio Probabilśtico em Inteligência Artificial**. Workload: 120h, Credits: 8, **Grade: A**.
- **IBI5037-1/5 Algoritmos em Bioinformática**. Workload: 120h, Credits: 8, **Grade: A**.

- **PCS5118-1/1 Seminários em Engenharia de Computação**. Workload: 30h, Credits: 2, **Grade: A**.
- **PCS5119-1/1 Tópicos Avançados em Engenharia de Computação**. Workload: 30h, Credits: 2, **Grade: A**.

Additionally the prerequisite to participate in the teaching internship as demanded by the scholarship regulations has been acquired by a one semester teaching internship under the supervision of *Profa. Anarosa Alves Franco Brandão* in the discipline *PCS3110-1 - Algoritmos e Estruturas de Dados para Engenharia Elétrica* in the 2nd semester of 2018.