Università degli Studi di Udine
Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Corso di Laurea in Informatica

Bachelor Thesis

MOCATHE
MOdels for CAncer THErapies

Candidate
Cristian Stocco

Supervisor
Prof. Carla Piazza

Academic Year 2018/2019

INSTITUTE CONTACTS
Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Università degli Studi di Udine
Via delle Scienze, 206
33100 Udine — Italia
+39 0432 558400
`https://www.dmif.uniud.it/`

a Lei che le nostre strade possano *allinearsi*
ai parenti, agli amici e ai conoscenti che sono stati vicini per questo percorso
ai professori e alla comunità scientifica per il supporto accademico

Abstract

MOCATHE (**MO**del for **CA**ncer **THE**rapies) is a software improvement of **CIMICE**, a software which calculates the inference of cancer progression by its phylogenetic.

Today, tumors are one of the principal causes of death and understanding how the progression of cancer manifests is a human help. Of course, from complexity of these evolutions there are a lot of models based on different mathematical properties. Our goal is to enrich **CIMICE** to estimate the inference of the regression of cancers by applying a *Medicine* model.

Contents

1 Introduction **3**

2 Premise **5**
- 2.1 [DTMC] Discrete-Time Markov Chain . . . 6
  - 2.1.1 Example . . . 6
  - 2.1.2 Properties . . . 7
- 2.2 [CTMC] Continuous-Time Markov Chain . . . 9
  - 2.2.1 Example . . . 9
  - 2.2.2 Jump chain . . . 10
- 2.3 [CPMC] Cancer Progression Markov Chain . . . 11
  - 2.3.1 Example . . . 11
- 2.4 Applications of Markov Chain . . . 11
  - 2.4.1 Branching Processes in DTMC . . . 12
  - 2.4.2 Moran Model in DTMC . . . 12
  - 2.4.3 M/M/s queue in CTMC . . . 13
- 2.5 Data . . . 13

3 Models **15**
- 3.1 Cancer Model . . . 15
  - 3.1.1 Model . . . 15
  - 3.1.2 Edge Weights . . . 16
  - 3.1.3 Example . . . 17
  - 3.1.4 Assumptions . . . 20
  - 3.1.5 Complexity . . . 20
  - 3.1.6 Multiple Mutations . . . 20
  - 3.1.7 Tools Comparations . . . 20
- 3.2 Medicine Model . . . 25
  - 3.2.1 Model . . . 25
  - 3.2.2 LOBICO . . . 26
  - 3.2.3 Example . . . 27
- 3.3 Therapy Model . . . 28
  - 3.3.1 Vertices Expansion . . . 28
  - 3.3.2 Model . . . 29
  - 3.3.3 Self-Loops . . . 30
  - 3.3.4 Example . . . 30
  - 3.3.5 Graph Dimension . . . 32

4 Model's Properties **33**
- 4.1 Damage & Diverge Acyclicity Property . . . 33
- 4.2 Repair Cyclicity Property . . . 33
- 4.3 DTMC Properties . . . 34
- 4.4 Dimensional Bounds . . . 36

5 Synthetic Case Studies **37**
5.1 Example 1 38
5.2 Example 2 39
5.3 Example 3 40

6 Implementation **41**
6.1 Development 41
6.1.1 Analyzing the parameters 41
6.1.2 Analyzing the DOT graph 41
6.1.3 Setting up the graph settings 42
6.1.4 Validating the graph 42
6.2 Execution Example 42
6.2.1 CIMICE 42
6.2.2 MOCATHE with missing vertices 42
6.2.3 CIMICE re-run 43
6.2.4 MOCATHE re-run 44

7 Conclusion **47**

1

Introduction

The goal of this work is to add some functionalities to an initial version of the **CIMICE** software. This implementation is a novel method for cancer evolution inference, where we integrate the possibility to add a **Medicine** model in order to predict how the cancer reacts to the treatment. This version is briefly called **MOCATHE** (**MO**del for **CA**ncer **THE**rapies).

In these years, tumor phylogenetics was a growing research to prevent and took care of people affected by disease, where phylogenetics is a study of history and relationships among organisms. Even if the progress in this field is really going on it is far from optimal. This is mainly due to the cancer complexity and heterogeneity. [5]

Different cancer subtypes involves different mutations of nucleotides such as structural changes, deletions, duplications, inversions, and translocations of genomic segments and epigentic modifications. [3]

Cancer heterogeneity and its evolutions can be modeled through phylogenetic trees where multiple states and paths represent all the possible progressions.

Therefore the creation of different models was made and still are going on in order to capture different properties as well as complexity. Moreover cancers developed resistance against therapies and treatments that can be modeled by the *Darwin's Natural Selection Law* in fact it is a disease defined by a progression of molecular changes.

CIMICE model is created by a mutational matrix of genotypes, links between genomic alterations and genes. The structure is based on DAG (Directed Acyclic Graph) processing all possible pathways and estimating probabilities on each step between states, called edges. In **MOCATHE** the DAG property is dropped in some cases so our model is based on general graphs giving great information of regression of disease. The structure of this work is based on *Markov's Chain* property holding the same simply points of **CIMICE**:

- to have a minimal set of assumption
- low complexity of the model
- to have well defined limits

- to have a concentrated feature for other improvements

The application of **MOCATHE** is instinctively easy to understand because we take the DAG and add some edges and rebalance the Markov's Chain.

By the way, we imagine that these studies and implementations could help doctors to have an estimation of best therapy treatments, their efficacies and properties by applying medicines. It lets doctors understand which is the best treatments to apply to patients relying also by their experiences and knowledges.

The thesis is structured in different chapters. A briefly introduction in the first chapter. The second chapter explains the *Markov Chain* models and some applications. The third chapter contains the **CIMICE** model of the initial version, the **Medicine** model we apply to **CIMICE** and the resulting model, called **Therapy** which describes the parallel composition of these models. After, we add few theorems to validate model's properties and the examples which show the application of our model in the third and fourth chapters respectively. In the six chapter we briefly explain the implementation made.

2

Premise

The topic of this work is BioInformatic, an interdisciplinary field which merges biology, computer science, mathematics, statistic and information engineering with the aim of analysing biological data through automatic computations. More in detail, the tumor phylogenetics and cancer evolution inferences are specific sections mentioned here.

Biology is a natural science which examines life and living organisms. Here we focus the analysis on molecular interactions and evolutions of DNAs, exploiting mathematical tools with the aim of defining data structures and algorithms for computing the statistical results.

The DNA (and similarly RNA), is a molecule which keeps the genetic information of the organism in double helix form. This is made by a sequence of four nucleotides: cytosine (C), guanine (G), adenine (A) or thymine (T). For RNA, thymine nucleotide is replaced by uracil (U) and the form is a single helix letting the organisms use it to replicate of cell.

In mathematics, graphs are a set of vertices which represents objects or states and the edges are connections between them. In our context the vertices represent the possible states of the biological system, while the edges are showing state changes. So the graphs we consider describe the evolution of a biological system.

In particular the mathematical models we refer to are Discrete Time Markov Chains: a sequence of states which defines the history of a process. The biological systems we investigate on are represented as DNA mutations, i.e., each state of the system is a possible set of mutations. The transitions are walks representing alterations of the nucleotide sequence.

Our Markov's Chains model the tumor phylogenetic process. The tumor phylogenetic, consequently cancer evolution, is the evolutionary history of the DNA sequences described as a graph with multiple predecessors as in the example in the next Section. In the literature tree models called oncogenetic or mutagenetic tree where each vertex has at most one predecessor are usually considered and we will briefly describe them in the following chapters. We drop the assumption of a single predecessor and generalize from trees to graphs.

Let us formally define our models. A Markov Chain is a discrete state-space process in which the next state depends only on the present one where the set of states are finite and countable. When discrete time systems are considered Markov Chains can be modelled as follows.

Basically, there are two different types of Markov Chain: the key difference between them is that DTMC contains a probability distribution over the transitions. While CTMC describes the times of transitions.

In this work, CTMCs are shown in order to have a comparison with DTMCs but they are not being used since DMTCs are transformed into Cancer Progression models. In fact, the CTMCs has the property to express times instead of probabilities over the edges: that assumption is difficult to predict as state-of-art [10].

2.1 [DTMC] Discrete-Time Markov Chain

Definition 1 (DTMC)**.** *The Discrete-Time Markov Chain is a pair $M = (V, p)$ in which*

- *V is a finite set of vertices*
- *p describes a probability distribution*

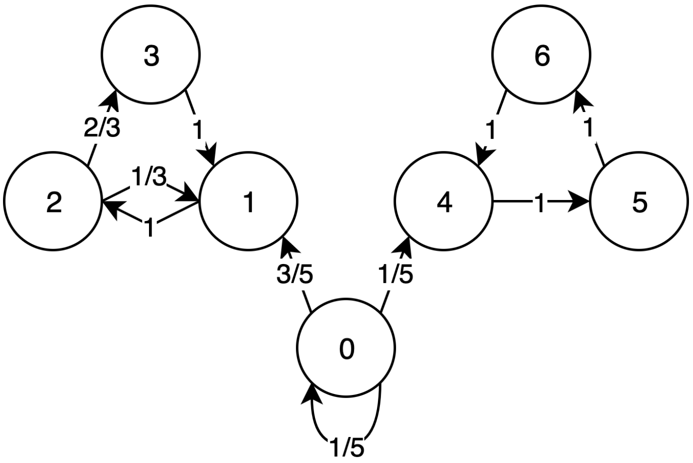
$$p : V \times V \to [0, 1]$$

- *for each vertex it holds probabilities over outcoming edges sum to 1*

$$(\forall u \in V)\left(\sum_{v \in V} p(\langle u, v \rangle) = 1\right)$$

2.1.1 Example

An example of DTMC on the left and the transition matrix on the right.



$$\begin{pmatrix}
\frac{1}{5} & \frac{3}{5} & 0 & 0 & \frac{1}{5} & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{3} & 0 & \frac{2}{3} & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0
\end{pmatrix}$$

2.1.2 Properties

We describe the main properties of *Markov Chains.*

Distributions

Given X as a model of a random state, $(X_n)_{n\geq 0}$ is a *Markov chain* with the *initial distribution* $\lambda \in \mathbb{R}^k_{\geq 0}$ and *transition matrix* $\mathbb{R}^{k\times k}_{\geq 0}$, where $k = |V|$ if

(i) $\mathbb{P}(X_0 = i_1) = \lambda_{i_1}$

(ii) $\mathbb{P}(X_{n+1} = i_{n+1}|X_0 = i_0, \dots, X_n = i_n) = p_{i_n i_{n+1}}$

We notice that for the definition of (ii) the distribution of the (i+1)-th state is independent from the random variables $X_{n\geq 0}$ for all $\{0, \dots, n-1\}$.

Discrete-Time Random Process

A Discrete-Time Random Process $(X_n)_{0\leq n\leq N}$ is a Markov(λ, p) if and only if for all $i_1, \dots, i_N \in I$ where I is the index variable.

$$\mathbb{P}(X_0 = i_1, X_1 = i_2, \dots, X_N = i_N) = \lambda_{i_1} * p_{i_1 i_2} * p_{i_2 i_3} * \cdots * p_{i_{N-1} i_N}$$

Reachability

The probability to reach a certain state in n steps could be calculated by the n-th power of the *transaction matrix p.* It's differentiated by using the *initial distribution* λ, so the starting vertex is chosen randomly.

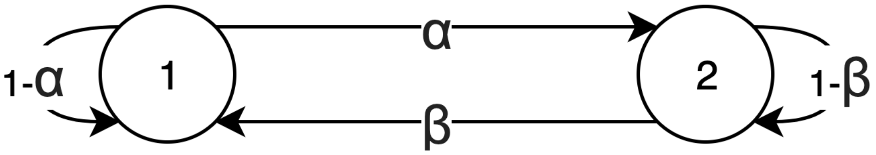
$$(\lambda P)_j = \sum_{i\in I} \lambda_i p_{ij}$$

By choosing a certain vertex i, all possible paths to target vertex j is defined as a Markov $(X_n)_{n\geq 0}$ and for all $n, m \geq 0$

(i) $\mathbb{P}(X_n = j) = (\lambda P^n)_j$

(ii) $\mathbb{P}_i(X_n = j) = \mathbb{P}(X_{n+m} = j \mid X_m = i) = (P^n)_{ij} = p^{(n)}_{ij}$

Reachability-Example: Two-state chain Given the diagram and the transition matrix



$$\begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix}$$

We manipulate the relation $P^{n+1} = P^n P$ to write that the probability to stay in the 2-th state is the probability to cross the edge $\langle 1, 2\rangle$ with probability

$$p^{(n+1)}_{12} = p^{(n)}_{12} * (1-\beta) + p^{(n)}_{11} * \alpha$$

In order to keep the previous equation with a unique variable, we apply the following invariant $p_{11}^{(n)} + p_{12}^{(n)} = 1$, so the previous equation is transformed as following

$$\begin{aligned} p_{12}^{(n+1)} &= p_{12}^{(n)} * (1-\beta) + p_{11}^{(n)} * \alpha \\ &= p_{12}^{(n)} * (1-\beta) + (1 - p_{12}^{(n)}) * \alpha \\ &= p_{12}^{(n)} * (1-\alpha-\beta) + \alpha \\ x_{n+1} &= x_n * c_0 + c_1 \end{aligned}$$

With the constant solution $x_{n+1} = x_n = x$ means that the $(n+1)$-th state is the same (n)-th, we therefore write $x = x * c_0 + c_1$ and for any $c_0 \neq 1$ the solution is $x = \frac{c_1}{1-c_0}$. Given the formula $y_n = x_n - \frac{c_1}{1-c_0}$ satisfies $y_{n+1} = c_0 * y_n$ representing the application of the n state where $y_n = c_0^n * y_0$.

$$x_n = \begin{cases} x_0 + n * c_1 & \text{if } c_0 = 1 \\ c_0^n * y_0 + \frac{c_1}{1-c_0} & \text{otherwise} \end{cases}$$

Therefore, applying c_0 and c_1 to $p_{12}^{(n)}$:

$$p_{12}^{(n)} = \begin{cases} 0 & \text{if } \alpha = \beta = 0 \\ (1-\alpha-\beta)^n \frac{\beta}{\alpha+\beta} + \frac{\alpha}{\alpha+\beta} & \text{otherwise} \end{cases}$$

$$\text{with } x_0 = 1 \text{ since it's the starting vertex}$$

$$y_0 = x_0 - \frac{c_1}{1-c_0} = \frac{(1-c_0)x_0 - c_1}{1-c_0} = \frac{(\alpha+\beta)x_0 - \alpha}{\alpha+\beta} = \frac{\beta}{\alpha+\beta}$$

Class Structure Whenever Markov Chains are big graphs, it could be rived into small pieces to understand the whole system. To group up single vertices into sets, we say that i *leads to* j, writing $i \rightarrow j$ if

$$p_{ij}^{(n)} > 0 \quad \text{for some } n \geq 0$$

Furthermore, i *communicates with* j, writing $i \leftrightarrow j$, when it holds that $i \rightarrow j \wedge j \rightarrow i$. We could see that this is the symmetric property of *communicating classes*, which is considered an equivalence relation on I, the state-space. The reflexive property satisfies intuitively $i \leftrightarrow i$ as well and the transitivity it's given by the previous equation, in fact when $p_{ij}^{(n)} > 0$ holds exists a generic path $i \rightsquigarrow j$, so exists a vertex k where $i \rightarrow k \wedge k \rightarrow j$.

Definition 2 (Communicating Classes)**.** *A Communicating Class is a set of vertices $C = \{1, \ldots, n\}$, substantially it could be compared to Strongly Connected Components, where*

$$\begin{aligned} \forall i \in [1, \ldots, n] &\qquad i \leftrightarrow i \\ \forall i, j \in [1, \ldots, n] &\qquad i \leftrightarrow j \wedge j \leftrightarrow i \\ \forall i, k, j \in [1, \ldots, n] &\qquad i \leftrightarrow k \wedge k \leftrightarrow j \end{aligned}$$

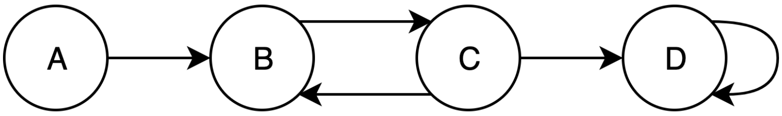
We define also the *closed class* C (a class where there is no escape) if

$$i \in C \wedge i \to j \text{ implies } j \in C$$

When $C = \{i\}$ the state i is called *absorbing* (a sink).

Recurrence & Transience Recurrence and transience are properties describing respectively that a state is belonging to a cycle, thus it keeps coming back while a transient state is a state that when it is left, it will not appear anymore.

Recurrence & Transience-Example In this example, A is a transient state, while B, C, D are recurrents and D absorbing. The *Communicating Classes* are $\{A\}$, $\{B, C\}$, $\{D\}$.



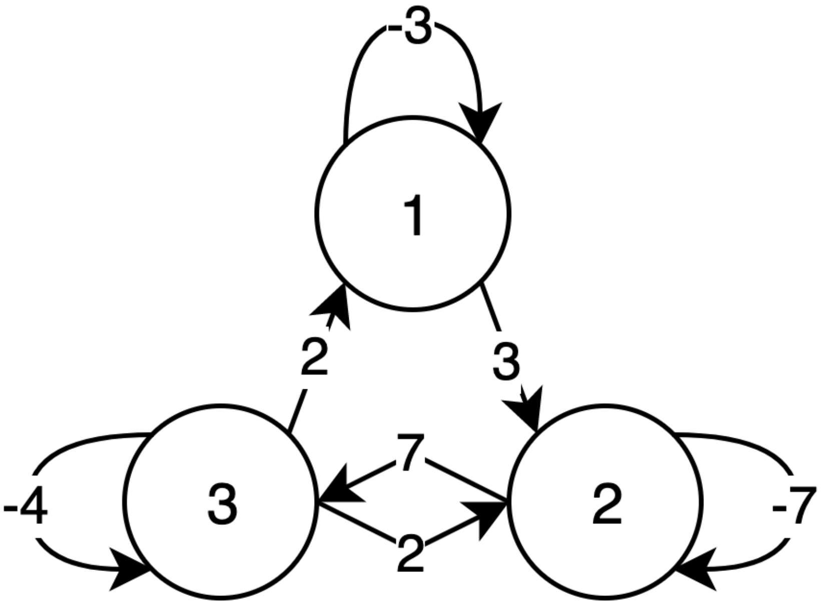
2.2 [CTMC] Continuous-Time Markov Chain

Definition 3 (CTMC)**.** *The Continuous-Time Markov Chain is a matrix Q in which $q_{ij} \in Q$:*

$$\begin{aligned} \forall i \quad & 0 \le -q_{ii} < \infty \\ \forall i,j \quad & i \ne j \to q_{ij} \ge 0 \\ \forall i \quad & \sum_{j \in I} q_{ij} = 0 \end{aligned}$$

2.2.1 Example

An example of CTMC on the left and the transition matrix on the right.



$$\begin{pmatrix} -3 & 3 & 0 \\ 0 & -7 & 7 \\ 2 & 2 & -4 \end{pmatrix}$$

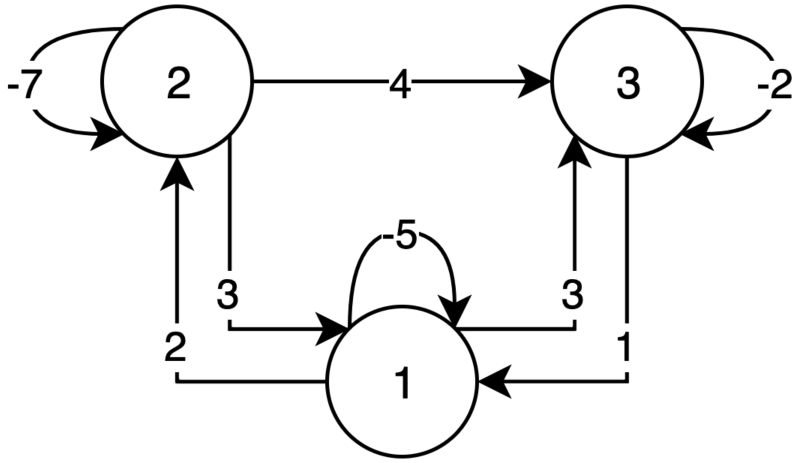
2.2.2 Jump chain

So far, the **CTMC** and **DTMC** are models based on same properties with the difference on the edge weights. It's possible to commute a **CTMC** into a **DTMC** with a transformation rescaling through a formula explained below. Before doing that, a notation $q_i = -q_{ii}$ is added to write a simplified version for *jump matrix* $\Pi = (\pi_{ij} : i, j \in I)$ of $Q = (q_{ij} : i, j \in I)$, with I a countable set.

$$\pi_{ij} = \begin{cases} q_{ij}/q_i & \text{if } i \neq j \wedge q_i \neq 0 \\ 0 & \text{if } i \neq j \wedge q_i = 0 \end{cases}$$

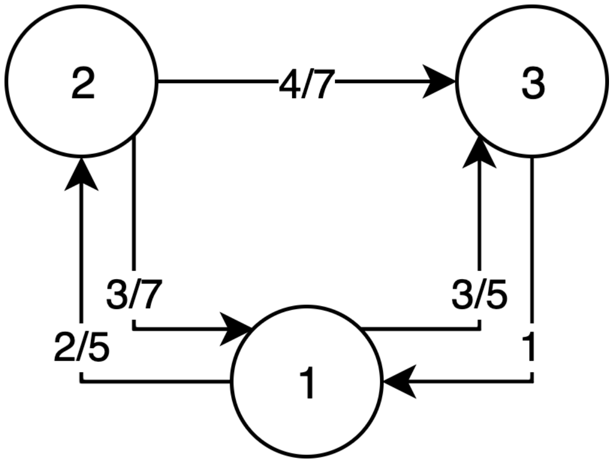
$$\pi_{ii} = \begin{cases} 0 & \text{if } q_i \neq 0 \\ 1 & \text{if } q_i = 0 \end{cases}$$

Thus, given a **CTMC** $(X_t)_{t\geq 0}$ on I with *initial distribution* I and *generator matrix* Q, its *jump chain* $(Y_n)_{n\geq 0}$ is a **DTMC** with *initial distribution* I and *probability distribution* Π.



$$\begin{pmatrix} -5 & 2 & 3 \\ 3 & -7 & 4 \\ 2 & 0 & -2 \end{pmatrix}$$

Figure 2.1: Continuous-Time Markov Chain Example



$$\begin{pmatrix} 0 & \frac{2}{5} & \frac{3}{5} \\ \frac{3}{7} & 0 & \frac{4}{7} \\ 1 & 0 & 0 \end{pmatrix}$$

Figure 2.2: Discrete-Time Markov Chain Example

2.3 [CPMC] Cancer Progression Markov Chain

In biological context genotypes describes a particular set of genes subject to mutations. In our models we will consider DTMCs whose nodes are genotypes while the edges represent the probabilities of acquiring new mutations.

Definition 4 (Genotype)**.** *Given a set of genes* $G = \{g_1, \ldots, g_n\}$*. The set of genotypes* $S = \{S_0, \ldots, S_m\}$ *is a set of* S_i *which is a subset of genes, where* $S_0 = \emptyset =$ *"clonal cell".*

Definition 5 (CPMC)**.** *The Cancer Progression Markov Chain is a pair* $C = (S, p)$ *in which*

- $S = \{S_1, \ldots, S_m\}$ *is a finite set of genotypes over a set of genes* $G = \{g_1, \ldots, g_n\}$

  $S_1 = \emptyset =$ *"clone" reaches any other genotype of the chain*

- *for each* $i, j \in [1, m]$ $i \neq j \rightarrow p(S_i, S_j) > 0$ *if and only if* $S_i \subseteq S_j$ *and*
  for each $k \neq i, j$ $(S_i \subseteq S_k \subseteq S_j) \rightarrow p(S_i, S_j) = 0$

The initial distribution λ is considered implicitly as the starting state is the clonal cell so $\lambda_{i_1} = 1$ and for all other vertices, its probability is 0: $\forall j \in \{2, \ldots, |S|\}\lambda_{i_j} = 0$.

2.3.1 Example

We show an example of a graph representing a genealogy tree of cells and a Markov Chain over genotypes. The graph on the left is showing how the cell is mutating starting from the "clone" cell, the initial state, and being modified to other genotypes. In the same way, on the right it's shown how the graph could be enriched to become a Markov Chain.

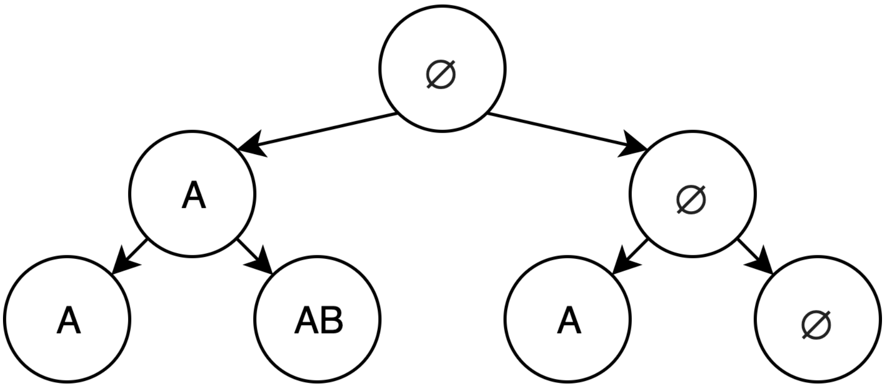


Figure 2.3: Genealogy tree of mutations

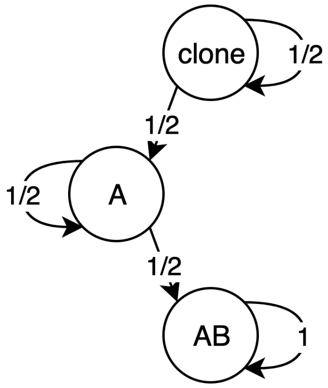


Figure 2.4: CPMC

The probability of transition from "clone" to A represents the probability that a normal cell generates a cell with mutation A.

2.4 Applications of Markov Chain

Markov chains is a useful model for any problem with random-processes such as biological models, queueing models, resource management model, Markov decision processes and Markov chain Monte

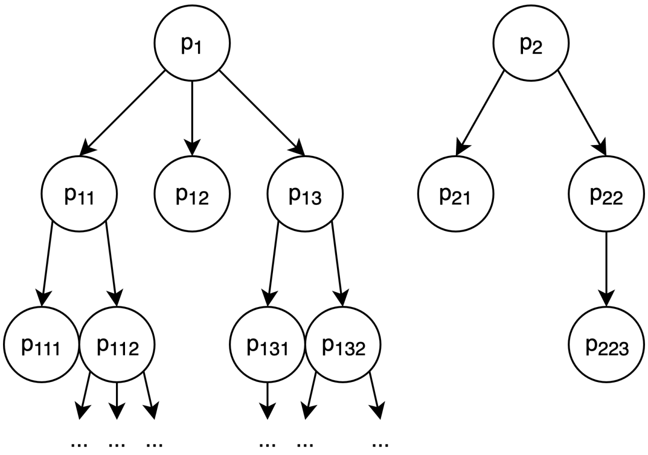
Carlo. Of course some examples in biology are explained for DTMC and a queueing model is described for CTMC.

DTMC Biology models contain high complexity so randomness is good to make distribution on genes hierarchy (like CIMICE), population growth, chain reactions. Other types are epidemics evolution through descendants and new generations, and inheritance of genes within families. These types of problems are studied by DTMC $(X_n)_{n\geq 0}$ since generations or evolutions are all discrete states n.

CTMC Queueing is a basic mathematical model for producer/consumer problem with server that are able to serve some tasks in FIFO (First-Input First-Output) mode. Since tasks are given as random processes at a certain time t exponentially distributed, it turns out that this distribution is handled by CTMC $(X_t)_{t\geq 0}$. In this field, it's computed the distribution on waiting times for tasks, stability on queues and queue length.

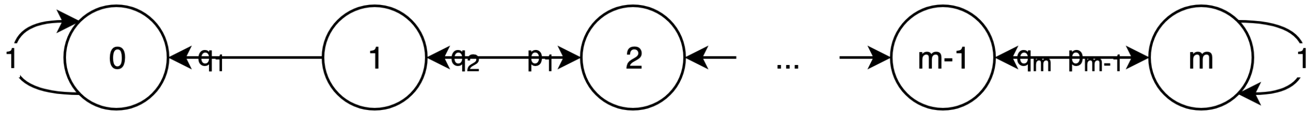
2.4.1 Branching Processes in DTMC

Galton and Watson developed the initial branching processes in 1870s for analysing the disappearance of family names even if the population was still growing. So having the assumptions that p_k is the probability of having k sons, the goal is to calculate after n generations the probability that there are no male descendants. This type of problem links branching processes with random walks, used also for chain reactions in chemistry and nuclear fission.



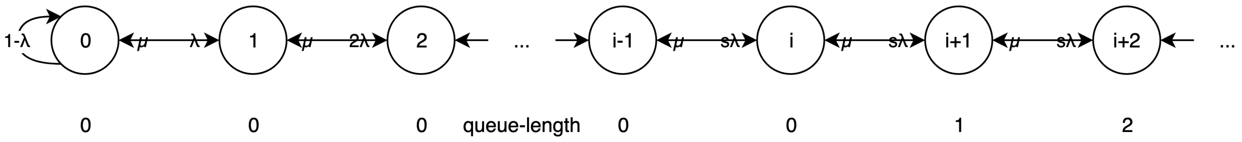
2.4.2 Moran Model in DTMC

The Moran Model is the birth-and-death chain on $\{0, 1, \ldots, m\}$ states for a population with two types of elements 'A' and 'B'. Without loss of generality, a state j has j elements of type 'A'. The population on time $n+1$ is given by replication of an element and deletion of another element in time n, which creates a Discrete-Time Markov chain with a transition matrix $\mathbb{P}$. That structure has obviously absorbing states on *0* and m because applying the $n+1$ rule at the population containing elements of the same type maintains the same population. Furthermore transient class $\{1, \ldots, m-1\}$ is also a communication class such as the absorbing states do.



2.4.3 M/M/s queue in CTMC

The problem here is explained with a queue which contains tasks handled by s servers. The queue size is handled by a CTMC at time t. The arrival rate of a process to be executed is λ and μ is the rate of each server. Thus, the total service rate (and maximum) is $\mu \times s$ with s server busy, in this case the queue has often some waiting task in the queue.



2.5 Data

Our probabilistic models are based on single cell sequencing experiments and data are displayed as *Mutational Arrays D*. These type of data are basically matrices $m \times n$ in which m is the number of samples and n is the number of genes. For each row, the number of ones represents the active genes for that cell so in short it defines the genotype.

There are some assumptions for the data:

ONE the analyzed cells are taken at the same time representing a snapshot of the cancer

POP the analyzed cells reflect the genotype distribution of the entire population

By the second assumption **POP** it means that the analyzed cells represents the global population of that type of cancer. In fact, three types of experiments are: **cross-selectional** where *patients* are samples, **regional** where *tissues* are samples and **single cell** where *cells* are samples, where samples are taken with next generation sequencing techniques like RNAseq or DNAseq.

Definition 6 (Dataset)**.** *The input dataset D is a matrix* $M^{m \times n}$.

$$D = \{gnt_i \mid i \in [m], n = |gnt_i|\}$$

The matrix is used to define the the distribution of initial lattice elements and their composition. Therefore, this simplex matrix is used to build the output by CIMICE and MOCATHE.

3

Models

In this chapter we formalize the models. We have **Cancer Models** as defined also in **CIMICE**, **Medicine Models**, and their composition which we call **Therapy Models** or briefly **MOCATHE**.

3.1 Cancer Model

The **Cancer Models** describe the probability to reach a certain mutational state starting from the initial state with no mutations, also called "clonal". The following definitions are used to create a progression of states and the model itself.

3.1.1 Model

The desease context is managed by already-implemented CIMICE model based on Markov's Chains to perform *single cell sequencing* with a minimal set of assumptions. Probabilities on transaction edges are describing the probability of cell mutation, without mentioning times because that property is difficult to predict as state-of-art [10]. They may vary between different people, body-regions, feedings, environments, and so on. For these reasons, the self-loop is not considered except in leafs.

Definition 7 (Observed Genotype)**.** *Given a set of genotypes* $S = \{S_1, \ldots, S_m\}$*, the observed genotypes is a probability function for the genotypes* S *in the dataset* D

$$P_D(gnt) = \frac{\#(gnt, D)}{|D|}$$

Definition 8 (Cancer Model)**.** *Given* S *be a finite set of genotypes, the cancer model* $A = \langle V_A, E_A, W_A \rangle$ *is a graph in which:*

$$V_A = \{\langle gnt, P_D(gnt)\rangle \mid gnt \in D \land P_D(gnt) > 0\} \cup \{\langle \emptyset, P_D(\emptyset)\rangle\}$$
$$E_A = \{\langle u, v\rangle \mid u, v \in V_A \land u[0] \subset v[0] \land |v[0]| - 1 = |u[0]|\}$$
$$W_A(\langle u, v\rangle \in E_A) : \textit{Section } 3.1.2$$

3.1.2 Edge Weights

The weights on edges are divided in four steps in order to make the transition probabilities:

- *UpWeights* extraction: defining the probabilities of a genotype to have a certain history
- *UpWeights* normalization: normalizing the *UpWeights* extraction on incoming edges
- *DownWeights* extraction: defining the probabilities of a genotype to evolve following a certain path using *UpWeights* data
- *DownWeights* normalization: normalizing the *DownWeights* extraction on outcoming edges

With some additional notations to create a definition for parents of v, children of u and the probability of a vertex v to be in the dataset D respectively

$$\begin{aligned} \Pi_v &= \{u \mid \langle u, v \rangle \in E_A\} \\ \Lambda_u &= \{v \mid \langle u, v \rangle \in E_A\} \\ p(v) &= v[1] \quad v \in V_A \end{aligned}$$

UpWeights Extraction & Normalization

In these steps, the empirical assumption is based by the most probabile origin of a genotype is the most observed one. So the formula is based of $p(u)$ and the recursion in the ancestor of u, divided by $|\Lambda_u|$ to give the same *UpWeight* number to histories with equal observed probability and filtering a node with many children.

$$W_{up}(\langle u, v \rangle \in E_A) = \frac{1}{|\Lambda_u|}\Bigg(p(u) + \sum_{w \in \Pi_u} W_{up}(\langle w, u \rangle)\Bigg)$$

$$\overline{W}_{up}(\langle u, v \rangle \in E_A) = \begin{cases} 1 & \text{if } u[0] = \emptyset \\ \frac{W_{up}(\langle u,v \rangle)}{\sum_{w \in \Pi_v} W_{up}(\langle w,v \rangle)} & \text{else} \end{cases}$$

DownWeights Extraction & Normalization

Here, the estimation is based on the observations of reachability based on successors of a node. While the normalization is expressing the evolution from linked genotypes having a proportionality from groups of genotypes and their observed probability. Normalization is also the final value for the weights on edges.

$$W_{down}(\langle u, v \rangle \in E_A) = \overline{W}_{up}(\langle u, v \rangle)\Bigg(p(v) + \sum_{w \in \Lambda_v} W_{down}(\langle v, w \rangle)\Bigg)$$

$$W_A(\langle u, v \rangle \in E_A) = \overline{W}_{down}(\langle u, v \rangle \in E_A) = \frac{W_{down}(\langle u, v \rangle)}{\sum_{w \in \Lambda_u} W_{down}(\langle u, w \rangle)}$$

3.1.3 Example

Given the entry table for the dataset the following pictures describe the applications of $UpWeights$ and $DownWeights$ both Extraction and Normalization functions.

$s \setminus g$	A	B	C	D
cmp1	1	0	0	0
cmp2	1	0	0	0
cmp3	0	1	0	0
cmp4	1	1	0	0
cmp5	1	1	0	0
cmp6	1	0	1	0
cmp7	1	0	1	0
cmp8	1	0	1	0
cmp9	0	1	0	1
cmp10	1	1	0	1
cmp11	1	1	0	1
cmp12	1	1	1	1

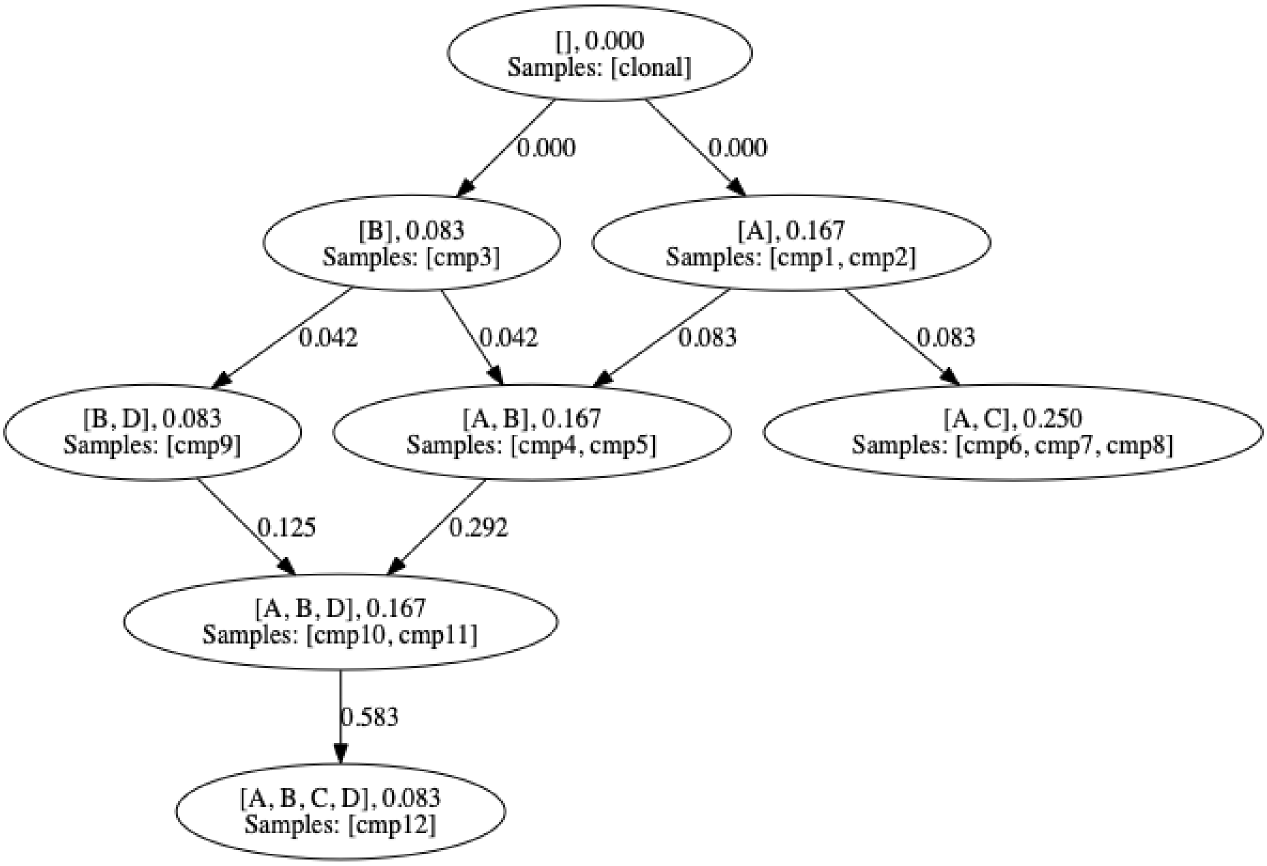

[], 0.000
Samples: [clonal]
0.000
0.000
[B], 0.083
Samples: [cmp3]
[A], 0.167
Samples: [cmp1, cmp2]
0.042
0.042
0.083
0.083
[B, D], 0.083
Samples: [cmp9]
[A, B], 0.167
Samples: [cmp4, cmp5]
[A, C], 0.250
Samples: [cmp6, cmp7, cmp8]
0.125
0.292
[A, B, D], 0.167
Samples: [cmp10, cmp11]
0.583
[A, B, C, D], 0.083
Samples: [cmp12]

Figure 3.1: Application of *UpWeights Extraction*

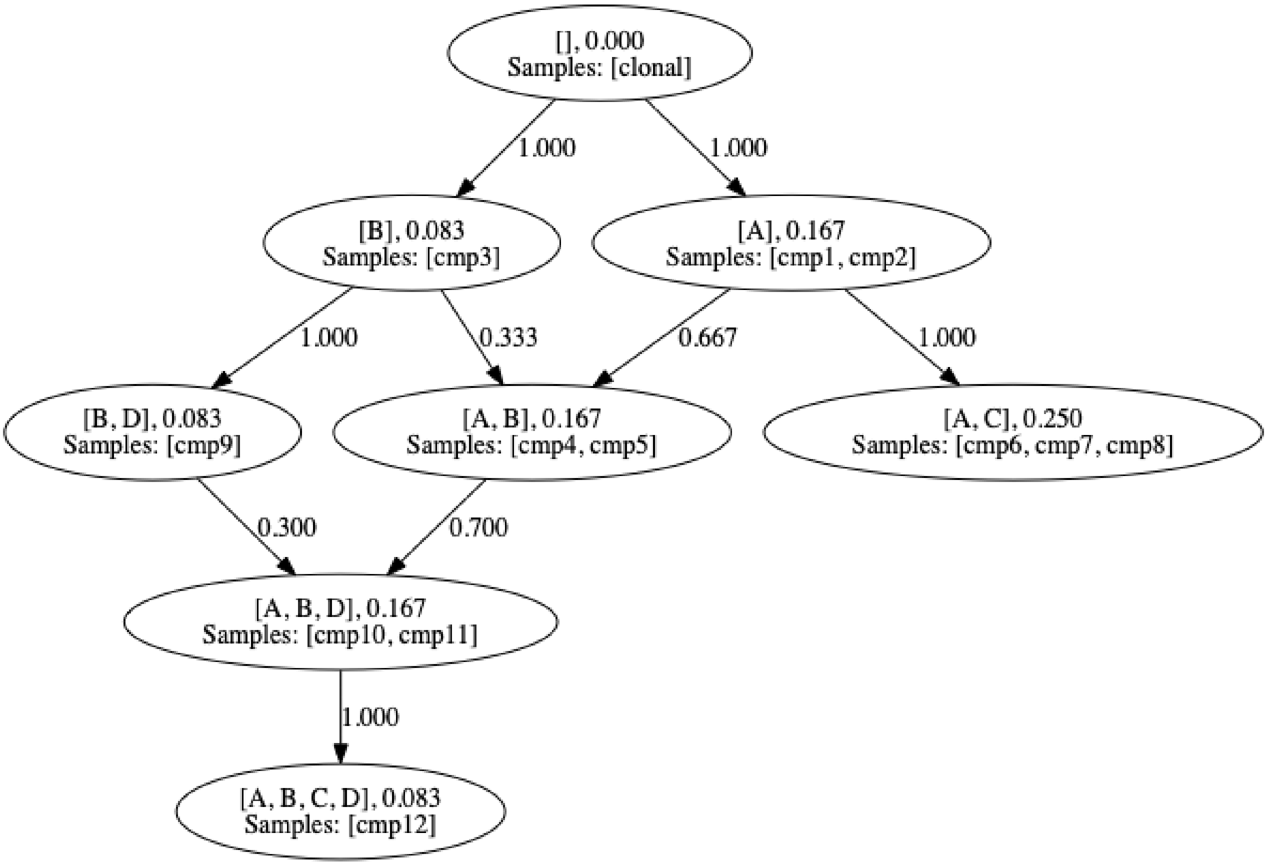

[], 0.000
Samples: [clonal]
1.000
1.000
[B], 0.083
Samples: [cmp3]
[A], 0.167
Samples: [cmp1, cmp2]
1.000
0.333
0.667
1.000
[B, D], 0.083
Samples: [cmp9]
[A, B], 0.167
Samples: [cmp4, cmp5]
[A, C], 0.250
Samples: [cmp6, cmp7, cmp8]
0.300
0.700
[A, B, D], 0.167
Samples: [cmp10, cmp11]
1.000
[A, B, C, D], 0.083
Samples: [cmp12]

Figure 3.2: Application of *UpWeights Normalization*

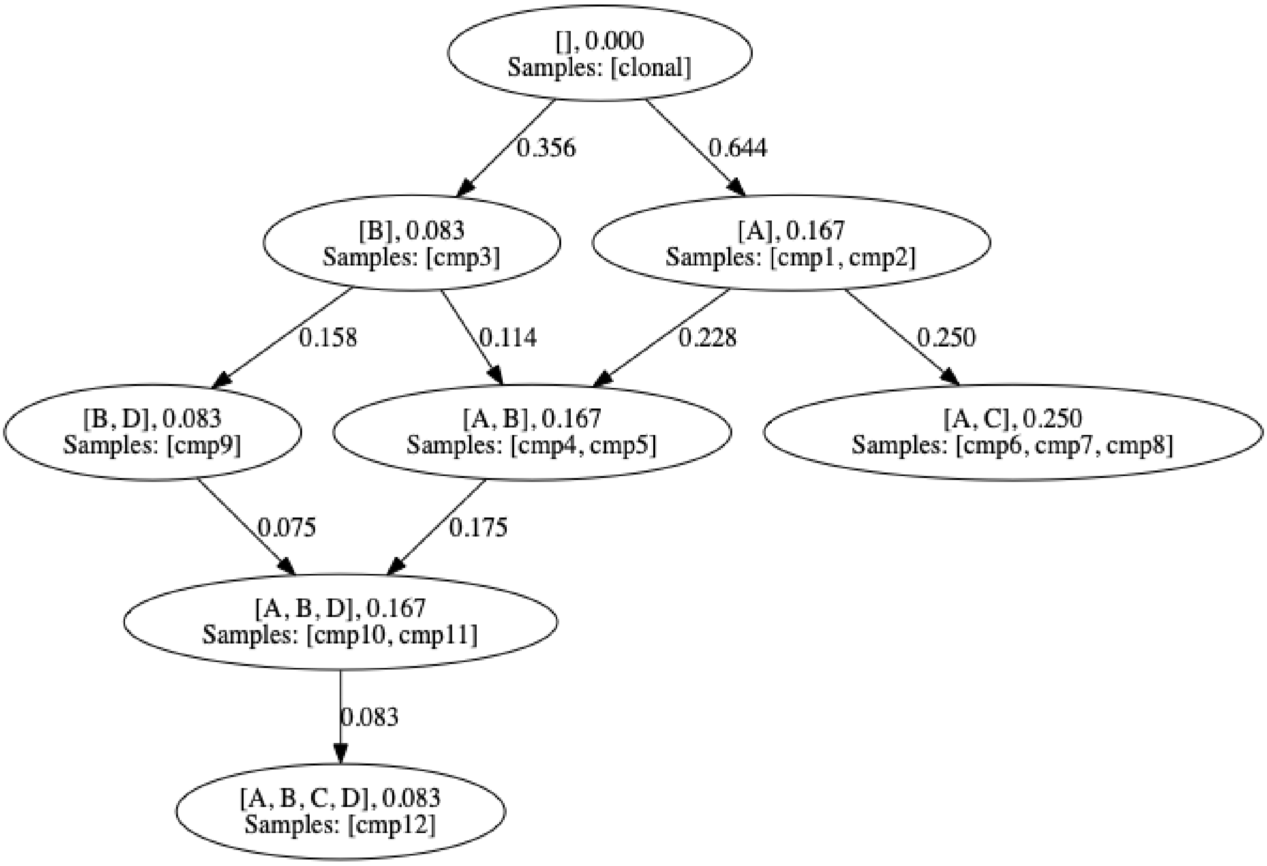

[], 0.000
Samples: [clonal]
0.356
0.644
[B], 0.083
Samples: [cmp3]
[A], 0.167
Samples: [cmp1, cmp2]
0.158
0.114
0.228
0.250
[B, D], 0.083
Samples: [cmp9]
[A, B], 0.167
Samples: [cmp4, cmp5]
[A, C], 0.250
Samples: [cmp6, cmp7, cmp8]
0.075
0.175
[A, B, D], 0.167
Samples: [cmp10, cmp11]
0.083
[A, B, C, D], 0.083
Samples: [cmp12]

Figure 3.3: Application of *DownWeights Extraction*

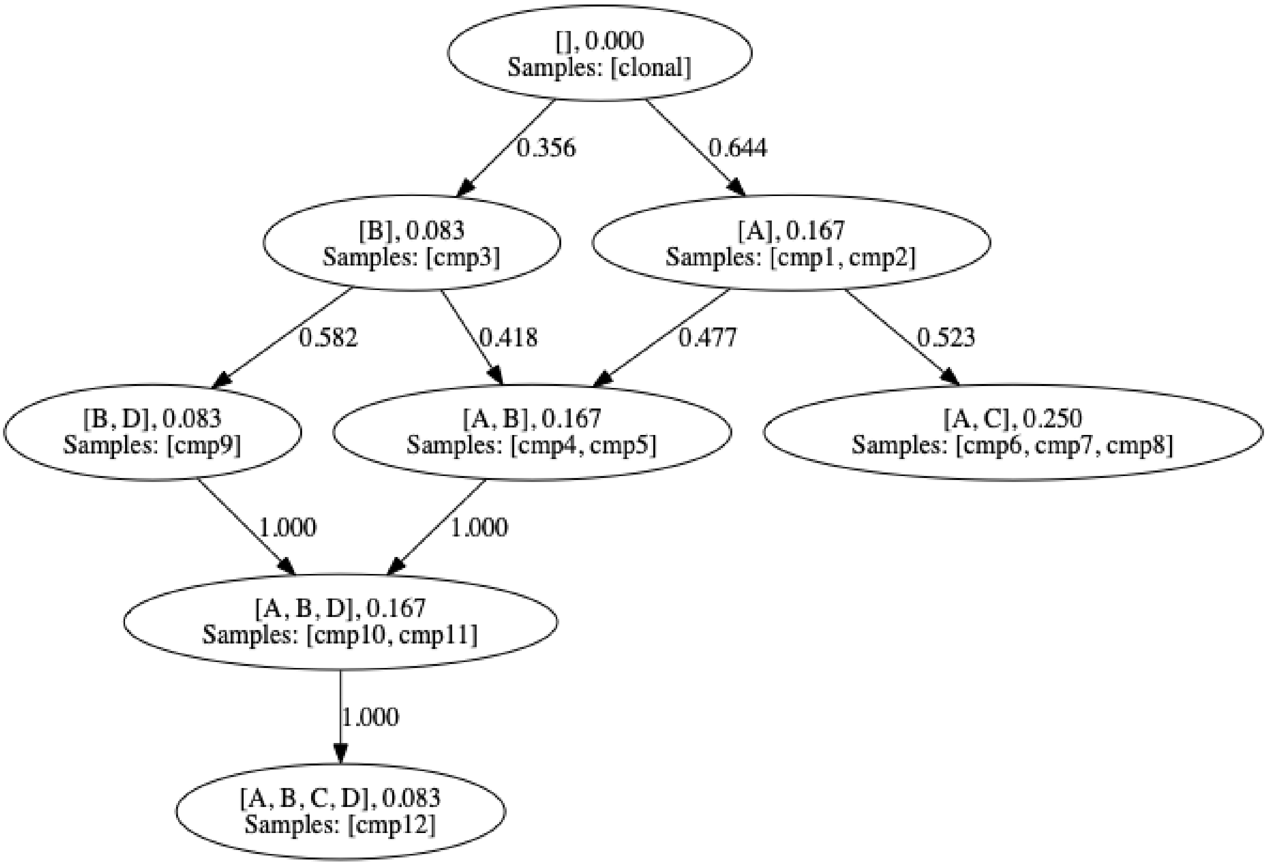

[], 0.000
Samples: [clonal]
0.356
0.644
[B], 0.083
Samples: [cmp3]
[A], 0.167
Samples: [cmp1, cmp2]
0.582
0.418
0.477
0.523
[B, D], 0.083
Samples: [cmp9]
[A, B], 0.167
Samples: [cmp4, cmp5]
[A, C], 0.250
Samples: [cmp6, cmp7, cmp8]
1.000
1.000
[A, B, D], 0.167
Samples: [cmp10, cmp11]
1.000
[A, B, C, D], 0.083
Samples: [cmp12]

Figure 3.4: Application of *DownWeights Normalization* & CPMC built by CIMICE

3.1.4 Assumptions

In order to build the CIMICE structure, some assumptions are made to keep the model simply:

UN mutations can only be acquired, either one at a time or in groups

HMC a mutational event in a cell occurs with a probability that only depends on its current genotype

AT the evolutionary history is always the one in which a minimal number of mutations is acquired at each time

Analysing the graph definition, we could notice that edges are created when its probability is greater than zero. For the **UN** hypothesis an edge $\langle u, v\rangle$ could be created if and only if genotypes in v is a subset of genotypes in u, more specifically acquiring either 0 or 1 mutation at each new generation. Thus, it's also obvious that the tree grows up toward the lower side of the genealogy tree. By **AT** the anti-transitivity property is applied to the graph: whenever edges $\langle u, v\rangle$ and $\langle v, w\rangle$ exist, the graph does not contain any edge $\langle u, w\rangle$.

3.1.5 Complexity

These assumptions keeps the complexity as lower as possible in order to reconstruct the model with low parameters and exclude the heuristic search, discarding any heavy theoretical complexity. Thus, **total estimated complexity is quadratic** over the number m of samples to create vertices, it's known that number of edges is quadratic over the vertices $|E| = \mathcal{O}(|V|^2)$, and finally to calculate the weights a linear bottom-up and linear top-down computation over the edges: $\mathcal{O}(m + m^2 + m^2) = \mathcal{O}(m^2)$.

3.1.6 Multiple Mutations

Given the definition of this model, in particular to the definition of edges, whenever a model contains all genotypes with at least two genes there are no links between the root and the other vertices, this reasoning could be applied also with genotypes of size n and any other genotypes of $n-2$ size which is a subset. So the genealogy tree could not be created accordingly and to solve this problem the definition of the edges are changed as following

$$E = \{\langle u, v\rangle : u, v \in V \wedge u[0] \subset v[0] \wedge \neg(\exists w \in V : w[0] \subset v[0] \wedge dist(w[0], v[0]) < dist(u[0], v[0]))\}$$
$$\text{with } dist(gnt_A, gnt_B) = ||gnt_A| - |gnt_B||$$

3.1.7 Tools Comparations

So far, we have described CIMICE as a model based on Markov Chains which computes in polynomial time with a DAG output. Of course, there are other methods and models based on different properties and below some tools are shown as state-of-art. For example, some tools applies an approach based on trees, instead of DAG, thus keeping the property that a node have only one parent. While others are based on clustering approaches. [4, 10]

These methods could be grouped in four classes:

- **Mutagenetic Tree & Oncogenetic Tree**
- **Bayesian Network**
- **Clustering & Evolutionary Fitting**
- **Other Approaches**

Mutagenetic Tree

The main difference between **Mutagenetic Tree** and our model CIMICE is based on mixtures of directed tree instead of DAG. More in details, a directed tree T is a directed graphs in which for each $u, v \in T$ the path is unique, and 'directed' means that a link (or edge) links them in a specific direction. The likelihood over edges is calculated over the set of all patterns of events using the time of occurrences of those events by EM Algorithm (Expectation-Maximization Algorithm) in which in the E-step software assigns data to tree components and estimates the missing data and in M-step it fits the trees on corresponding subsets. *Total computational time* is $\mathcal{O}(K \times n^3 + K \times n \times m)$ with K means the number of trees, n is the number of events (genes) and m is the number of observations (samples). Moreover, since usually number of simples m and number of genes n keeps this relation $m \gg n$, this formula can't be reduced with some assumption or prediction. For the calculation of the likelihood for a pattern the *BFS Algorithm* (Breadth-First Search) is used. The **Mutagenetic Tree** are an extension (or mixtures) of *Oncogenetic Tree* because these are a single tree model, having a first tree component with star topology and other trees a generic directed tree structure.

Comparing our base model we notice that a DAG is more flexible: for example let us have a simple case, a vertex labelled AB. This genotype could be reached by two different vertices both A and B. In a DAG, this is possible because the structure is a graph, instead the tree does not allow any form of multiple parents and this is a limitation.

Speaking about computation complexity, CIMICE has generally a computational time higher than this method for the quadratic time over the simples. [1, 8]

Oncogenetic Tree

Formally, an oncogenetic tree has a similar definition of CIMICE: $\mathcal{T}(V, E, r, W)$ in which genetic events are $V = \{0, \ldots, l\}$ with binary variables $X_1, \ldots, X_l$ containing the occurrence of them, E are the set of edges with weights $W : E \to [0, 1]$ such that $W(e = \langle u, v \rangle) = p(X_v = 1 \mid X_u = 1)$ describing the conditional probability of an event X_v when X_u already occurred and $r \in V$ is the tree root. However, in the graph not all combinations of genetic events are linked, in fact a sample $x \notin \mathcal{T}$ whenever $W(x \mid \mathcal{T}) = 0$. This derives that an approach like cross-validation for model validation is not possible since they're based on two subsets with elements not present in the tree. In the previous section, we described **Mutagenetic Tree** as a multiple composition of **Oncogenetic Tree** and the estimation in E-step refers to trees generated from observed data fixing the issue of those elements, letting cross-validation possible.

The **Oncogenetic Tree** has an additional property so **Mutagenetic Tree** has too: a time built using independent Poisson processes for the occurrence of the events. Performed by simulating the waiting process over the tree edges it brings an additional information which CIMICE does not provide. By the way, this model is simpler than the previous, by approximation via reduction computation complexity of CIMICE is still higher and structure has about the same characteristic, a little simplier. [1, 8]

Bayesian Network

The **Bayesian Networks** allows multiple parental nodes, thus the software output is a graph similar to CIMICE behaviour, dropping the assumption as described in *Mutagenetics Tree* and *Oncogenetics Tree*. However the basic property of Bayesian Network is to accumulation of events cause-effect, little similar to CIMICE: in our model the vertices are linked by the probability of mutating from a cell to another. It's little similar because every edge $\langle u, v\rangle$ has the specific characteristic that $u \subset v$, but for the Multiple Mutation (Section 3.1.6) u could contains a lot of genes more than v.

The **CT-CBNs** (Continuous Time - Conjunctive Bayesian Networks) are defined by a *poset* P (partial order set) of mutations and by the rate of fixation in the dataset with a transitive relation $\preceq$. The generic order relation $c \prec e$ defines that c is the cause of the effect e and this reflects, of course, the main property of Bayesian Network. Moreover, we refer also to $pa(i)$ as the set of mutations needed to be present before i appears. In the **H-CBN** version, parameters are estimated by EM Algorithm, the same used in *Mutagenetic Tree* with a timeline for the speed of carcinogenesis T_i which is an estimated and exponential random variable because it may vary from different components in the same population $T_i \sim e^{\lambda_i} + \max_{j \in pa(i)} T_j$. Looking at the formula, another property of **CT-CBN** is that an event i is analyzed after all preceding j parents are occurred making a *distributive lattice*. This formula is inherited from **D-CBN**, in fact this model makes the base for **CT-CBN** with timeline formula, another difference with *CIMICE*. [4, 2, 10]

Clustering

In **Clustering** methods, there are two approaches for learning data with a supervised and unsupervised analysis. In supervised analysis, there are initially sets with components having same characteristics, while in unsupervised every datum has the same role.

With the **first approach**, a Bayesian network envelopes the interrelation of mutations and its statistical inference based on parents in the network and frequency of each state to cluster patient samples into groups, taking care of interactions amid mutated genes. The target is to uncover common characteristics, therapeutic methods, and predictive biomarkers. In the Bayesian network, the edges represent the co-occurrence, mutual exclusivity or higher order correlations between sets of genes. Analysing those diagrams, it's found that there are highly connected genes across cancer types to deduce genetic correlations such as cellular processes and DNA damage repair.

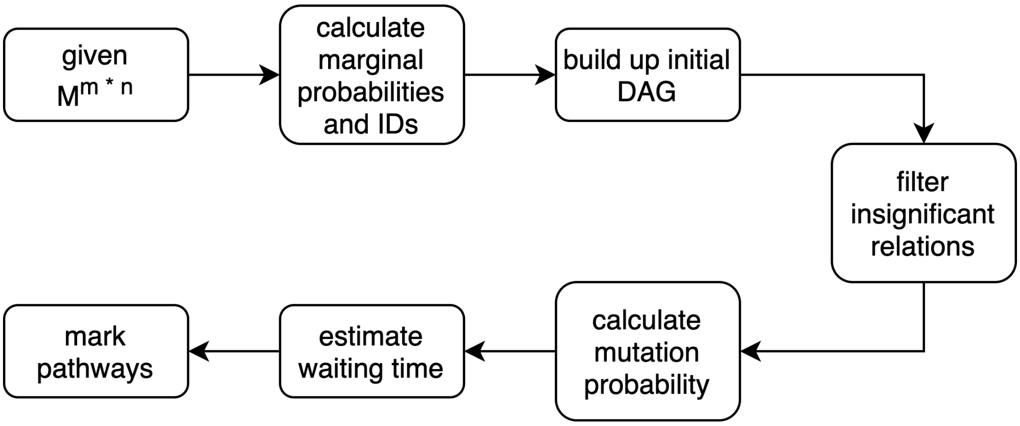
In the **unsupervised analysis**, the model is based on de novo clustering of the binary mutation data with a Bayesian network for each cluster (patient data). The comparison between these two models results that the unsupervised analysis is a better approach, also for the generation of new cluster for differentiate cancer types and patient samples. Every cluster is based on mutational data giving biological signals like survival prediction or overall survival with a percentile. Differently from the supervised method, the gene are not highly connected but the clusters are characterised by the combination of them giving also a patient stratification to improve a possible target treatment. In this work, computational complexity is not mentioned so it's not possible to compare with our tool *CIMICE*, but output structure is a DAG computed by MCMC (Monte Carlo Markov Chain) and MAP (Maximum a Posteriori). [5]

Evolutionary Fitting

The **PGM** (Probabilistic Graphical Model) infers cancer progression to calculate causal dependences and waiting times through a dependency theory for both pathways and individual genes to fit the heterogeneity and carcinogenesis, from different patients in the population. This method belongs to the third class, defining a probabilistic DAG given a matrix $M^{m \times n}$ with m rows and n genes (same of *CIMICE* input dataset structure) where an edge depends on

- marginal probability and conditional probability of gene
- ID (Intersection Degree) which measures the relationship between genes

Furthermore, ID applies two algorithms Standard Maximum Likelihood and BIC (Bayesian Information Criterion) in order to filter insignificant relations.
Briefly, the PMG workflow is detailed as following:


given M^m * n
calculate marginal probabilities and IDs
build up initial DAG
filter insignificant relations
calculate mutation probability
estimate waiting time
mark pathways

Output structure is similar to *CIMICE* but computational complexity is not mention, even if we guess it's polynomial time. [10]

Other Approaches

The **Progression Network** model is a special case of BNs, and they CBNs described previously are so similar in fact they are special cases of our monotone BNs with $\epsilon = 0$, both used to describe disease progression problems. This model, however, is reduced to a *MILP* (Mixed Integer Linear Programming) and even if it's NP-complete good heuristics exist. These models are represented by directed hypergraphs, remembering that in these data structures, an edge could connect more than 2 vertices. a *k-uniform hypergraph* (*k-bounded*) is a hypergraph in which all hyper-edges links exactly k-vertices. So graphs are 2-uniform hypergraphs.

The heuristic applied is solved by the *Maximum Likelihood* (ML) or *Bayesian Information Criterion* (BIC) score, both algorithms reduce the problem of learning a *Bayesian Network* with bounded number of parents making it acyclic as well and highest score. In each hyperedge there is an upper bound on the probability of the child vertex being occurred even if not all parent vertices have happened and moreover they are suited in the generating model where depends also the upper bound of the vertices parent in monotone and semi-monotone PNs. This characteristic is also suitable for modeling this problem, because general BNs does not.

The program **DiProg** is also based on probabilistic model and in the paper it was being compared with H-CBN algorithm, resulting that it can handle more variables than H-CBN. The output is a DAG as well, since they proved that it is acyclic and also it is directed. Thus the result is similar to *CIMICE* and computational complexity is not mentioned. [3]

3.2 Medicine Model

The **Medicine Models** represent a logic formula, its probability of application with a specific type. The model is a graph containing the informations of the therapy type and its probability of the cure, called also precision. Before the definition, the treatment of the DNA is split into three types:

1. **DNA Damage** with suffix D
2. **DNA Diverge** with suffix D
3. **DNA Repair** with suffix R

The *DNA Damage* method is more drastic as the name suggests. Differently from *DNA Repair*, the cell is brought to a state with a lot of mutation, by the effect of medicine which damages the cell itself. Thus, after a while the cell is dying for instability as the known chemotherapeutic treatments work. Similar approaches are the *homologous recombination* and *homology directed repair* which do not remove the damage but it synthesizes the sequence complementary to the damage area of the genoma. [9]

The *DNA Diverge* is a treatment which brings the cell to a state with other mutations, similar to *DNA Damage* with the difference that a cell in this state will be less risky than the other states. Subsequently, the treatment applied to the patient may be different to the previous changing from a weighty to a slightly medicine.

The structures for this method and *DNA Damage* are the same, the difference resides in the semantic of the F added vertex.

The *DNA Repair* method helps to refactor the DNA mutation of genes and restores the original state, so the organ function is about to be reset as previously. By the way, it has been shown that this method has some defects because the approach is huger and deeper than the *DNA Damage* and *DNA Diverge* but it is preferred to the others because it maintains the cells and preserves the structure of the body. The method is implemented by some ways such as mismatch repair, base excision repair, nucleotide excision repair, and the directed repair/Fanconi anemia. [9]

3.2.1 Model

Definition 9 (Medicine Model)**.** *The medicine model* $M_L = \langle V_L, E_L, W_L \rangle$ *is a graph with the following attributes:*

- $\varphi = \theta_1 \vee \cdots \vee \theta_n$*: logic formula (Definition 10) of the medicine* M_L
- p*: precision of the medicine* M_L
- *suffix* $L \in \{R, D\}$*: label of the treatment type*

in which:

$$V_L = \{\varphi, L\}$$
$$E_L = \{\langle\varphi,\varphi\rangle, \langle\varphi, L\rangle, \langle L, L\rangle\}$$
$$W_L = \{w(\langle\varphi,\varphi\rangle) = 1 - p, w(\langle\varphi, L\rangle) = p, w(\langle L, L\rangle) = 1\}$$

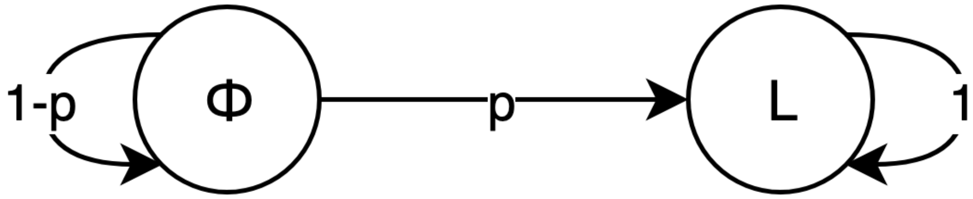


Figure 3.5: Medicine Model

Definition 10 (Logic Formula)**.** *The logic formula* φ *for medicine model is a predicate logic formula in Disjunctive Normal Form (DNF) in which:*

$$\varphi = \varphi \vee \theta \ \mid\ \theta$$
$$\theta = \theta \wedge g_i \ \mid\ \theta \wedge g_{i-NOT} \ \mid\ g_i \ \mid\ g_{i-NOT}$$

This is a standard definition also used for LOBICO [6] (see Section 3.2.2) in which θs express the properties of healing some genotypes, in fact a medicine can be used by different treatments and the DNF describes the total information about them.

3.2.2 LOBICO

LOBICO (http://lobico.nki.nl) is an example of medicine model based on DNF logic formulas of binary inputs. The predicate variables could be for example DNA mutations or CNAs, while there are two further parameters describing the formula: [6]

M number of φ (according to Definition 10) = “number of disjoint”

K max number of ϕ for each φ (according to Definition 10) = “max number of predicates in each disjoint”

Description Briefly, following two tables are showing some examples and structures for M and K.

$K = M = 1$ is a simple single-predictor

$K = 1 \wedge M > 1$ is a complex single-predictor

$K > 1 \wedge M = 1$ is a simple multi-predictor

$K > 1 \wedge M > 1$ is a complex multi-predictor

Examples of LOBICO These are some examples taken from LOBICO database applying the previous table definitions of M and K. Consider that & is logically linked to the connective logic $\wedge$ while | stands for $\vee$.

PARAMETERS	FORMULA	MODEL
$K = M = 1$	a7q36.2	a7q36.2
$K = 2 \wedge M = 1$	EWSR1-FLI1 | TLR-DOWN	EWSR1-FLI1
$K = 1 \wedge M = 3$	a(CCND1, CTTN) & d(FAT1) & ¬TNFa-UP	17-AAG
$K = 2 \wedge M = 2$	FLT3 & H2O2-DOWN | ¬NRAS & TP53	FLT3

3.2.3 Example

The following example represents a Medicine with type *Repair*. Its probability is $p = 0.8$ used to build also the *Therapy Model* applied to vertices that the formula $A \vee B$ activates.

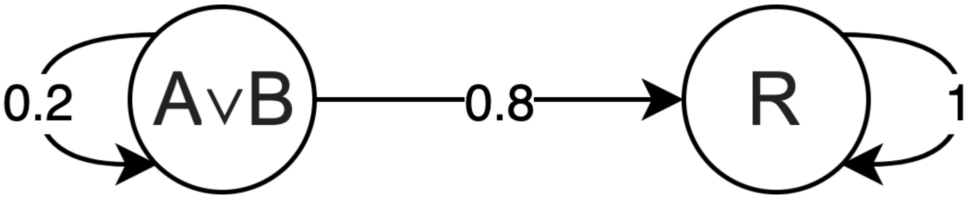


Figure 3.6: Example of Medicine Model

3.3 Therapy Model

We present three different approaches for the *medicine*, and the *therapy* are categorized in the same way. In fact, the graph built for *therapy* is given by the parallel composition of both *medicine* and *cancer* models and when edges are added the weights of all edges of activated vertices are recalculated to keep the invariant described by the definition 8 (third formula). For this detail, the Theorem 3 proves that the invariant is kept also in this graph.

Using *DNA Repair* the created edges are pointing some previous state: this behaviour means that a state with a specific genoma is reduced to a cell with less mutation. This previous state is also closer to "clone" cell.

Notice that the created edges are dropping the DAG property, verified by Theorem 2.

Instead for *DNA Damage* and *DNA Diverge* the created edges are pointing a specific state F. This state is a special state which describes the process of cell either suicide while adding a lot of mutations or staying in a mutated state for a different therapy treatment.

For this graph, the DAG structure is kept. The Theorem 1 verifies this.

The created edges destroy the invariant shown in CIMICE, specifically in the third formula of Definition 8. So, for the Theorem 3 we show that the balancing functions for the weight in the *DNA Therapy Models* (Definitions 12) are preserving the probability distribution over the edges.

3.3.1 Vertices Expansion

The vertices expansion is due to the application of the **Medicine Model** in *DNA Repair* type. Applying the formula which removes some genotypes to a particular gene does not mean that vertex belongs to the graph, sometimes it could not be present in the data set. In our software, a small program add these particular vertices with lowest frequency and rerun CIMICE software to obtain the graph with all vertices.

Consider the simplest example of having a graph with the genotype (so the vertex) named AB, thus applying the formula $\varphi = A$ it will be created a vertex B to handle this situation.

The **Vertices Expansion** is described by the following formula:

$$V_{exp} = \{u \setminus positiveLiteral(\varphi_u) \ \mid \ u \in V_A\} \setminus V_A$$

Max Expansion The maximum possible expansion is gained when all vertices are activated and the arrival pointer does not exists, remembering that all vertices could be activated but "clone" does not. This is verified by Theorem 5.

$$|V_{exp}| \leq |V_A| - 1$$

3.3.2 Model

The process to build this model is copying the model A built from CIMICE and adding some edges which describe the effects and probabilities of applying M_L, the medicine model. The formula is modulized to a *DNF* because more than one sub-formula could activate a specific vertex, therefore a transaction edge is created to link the activated vertex with the arrival vertex.

Parallel Composition The parallel composition creates some edges in the *Therapy Model*, thus it's defined φ_u as the activation of a generic vertex u using the medicine formula φ.

$$(\forall u \in V_A)(\varphi_u = \{\theta \mid \theta \in \varphi \wedge u \models \theta\})$$

Multiple Application In our model, multiple subformulas could activate a vertex in V_A, in fact those subformulas are a subset of the same vertex. Given the simplest example with a vertex AB and the formula $\varphi = A \vee B$, instinctively both subformulas A and B activate it.

To keep the model simply, we aggregate every subformula in order to merge every gene.

Multiple Application-Example Given the following example, we show the multiple application to the greatest vertex from the formula $\varphi = AB \vee CD$.

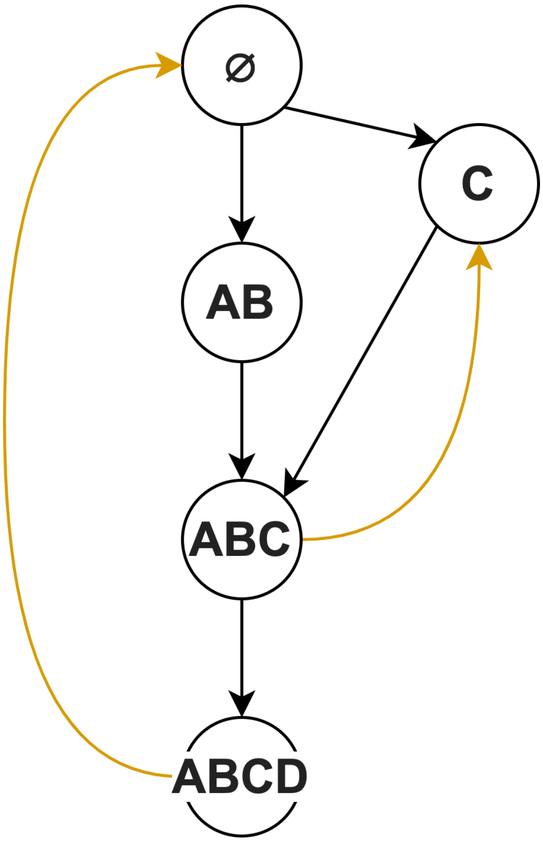


Figure 3.7: Example of Multiple Application

In this example, we explain the aggregation of both formulas ϕ which activate $ABCD$. In fact since both do it, for the **Multiple Application** we join all the genomes and consider an unique genotype in order to link to the proper arriving vertex.

Definition 11 (literal & positiveLiteral). *The literal and positiveLiteral are two formulas to transform a predicate formula into the corresponding set.*

$$\begin{aligned}\text{given } \varphi &= \theta_1 \vee \cdots \vee \theta_m \\ \text{given } \theta &= P_1 \wedge \cdots \wedge P_n \\ \text{literal}(\theta_i) &= \{g_i \mid \theta \equiv (P_1 \wedge \cdots \wedge P_n) \wedge (\exists j \in [n])(g_i \equiv P_j \vee g_i \equiv \neg P_j)\} \\ \text{positiveLiteral}(\theta_i) &= \{g_i \mid \theta \equiv (P_1 \wedge \cdots \wedge P_n) \wedge (\exists j \in [n])(g_i \equiv P_j)\}\end{aligned}$$

Definition 12 (Therapy Model). *The Therapy Model $C_L = \langle V_L, E_L, W_L \rangle$ is given by the parallel compositions of the following models*

- *cancer model $A = \langle V_A, E_A, W_A \rangle$*
- *medicine model $M = \langle V_M, E_M, W_M \rangle$, where $V_M = \{\varphi, L\}$*

having the following attribute

- *suffix $L \in \{R, D\}$: label of the treatment type (as Definition 9)*

$$V_L = \begin{cases} V_A \cup V_{exp} & \text{if } L = R \\ V_A \cup \{F\} & \text{otherwise} \end{cases}$$

$$E_L = E_A \cup E_{AM} \qquad E_{AM} = \begin{cases} \{e = \langle u, v \rangle \mid u, v \in V_L \wedge v = u \setminus \text{positiveLiteral}(\varphi_u)\} & \text{if } L = R \\ \{e = \langle u, F \rangle \mid u \in V_L \wedge literal(\varphi_u) \neq \emptyset\} & \text{otherwise} \end{cases}$$

$$W_L(e = \langle u, v \rangle) = \begin{cases} p & \text{if } e \in E_{AM} \\ W_A(e) * (1 - p) & \text{if } e \in E_A \wedge literal(\varphi_u) \neq \emptyset \\ W_A(e) & \text{if } e \in E_A \wedge literal(\varphi_u) = \emptyset \end{cases}$$

3.3.3 Self-Loops

The purpose of self-loops on vertices is to express a probability to stay in a certain state, thus expressing that on a certain time Δt there is the possibility to transitate in either another genotype or staying in the same. CIMICE model is describing the probability of transitions amid mutations and times are not mentioned to keep low complexity. Furthermore in the Premise, we said that this is difficult to manipulate as a state-of-art.

However, whenever a *Therapy Model* treats a CIMICE Model adding an edge from a leaf either to the F state in both *Damage* or *Diverge Type* or to a created vertex in *Repair Type* (see Section 3.3.1), self-loops are kept to maintain the structure of the Model itself and because we don't have any heuristic to manipulate this exception.

3.3.4 Example

The following example is the parallel composition of the *Cancer Model* described in Section 3.1.3, using the following *Medicine Model* with both types. Notices that blue edges are the added ones and D vertex is created even if in the initial dataset it does not exist.

$$\varphi = (A \land B) \lor (C) \qquad p = 0.8$$

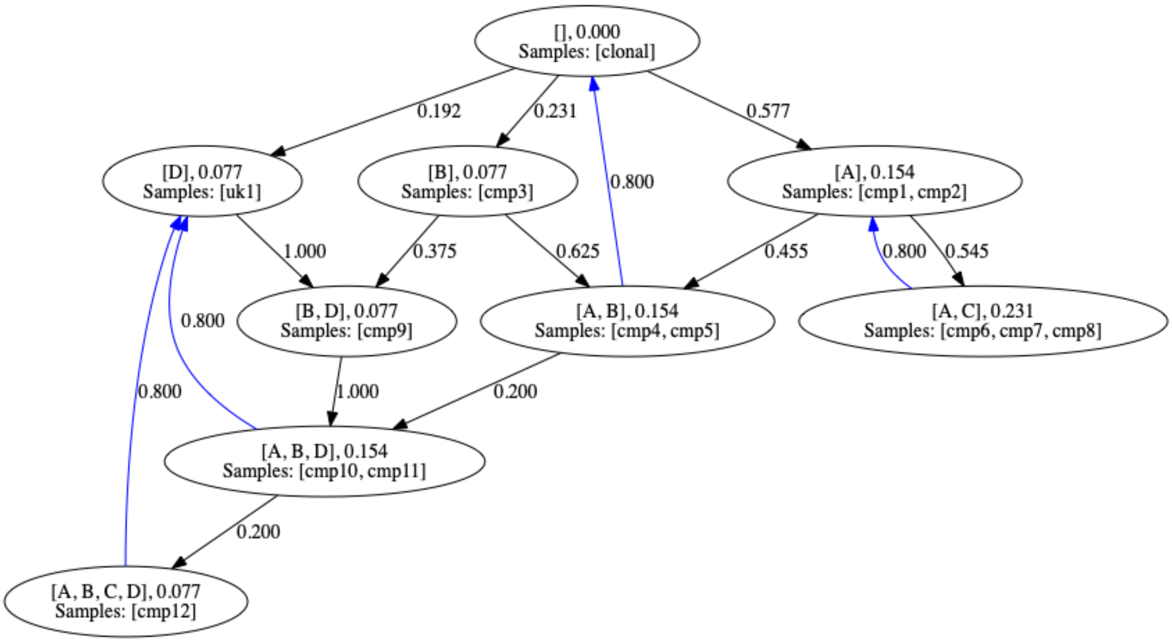

[], 0.000
Samples: [clonal]
0.192
0.231
0.577
0.800
[D], 0.077
Samples: [uk1]
[B], 0.077
Samples: [cmp3]
[A], 0.154
Samples: [cmp1, cmp2]
1.000
0.375
0.625
0.455
0.800
0.545
[B, D], 0.077
Samples: [cmp9]
[A, B], 0.154
Samples: [cmp4, cmp5]
[A, C], 0.231
Samples: [cmp6, cmp7, cmp8]
0.800
0.800
1.000
0.200
[A, B, D], 0.154
Samples: [cmp10, cmp11]
0.200
[A, B, C, D], 0.077
Samples: [cmp12]

Figure 3.8: Application of MOCATHE, with *Repair* type

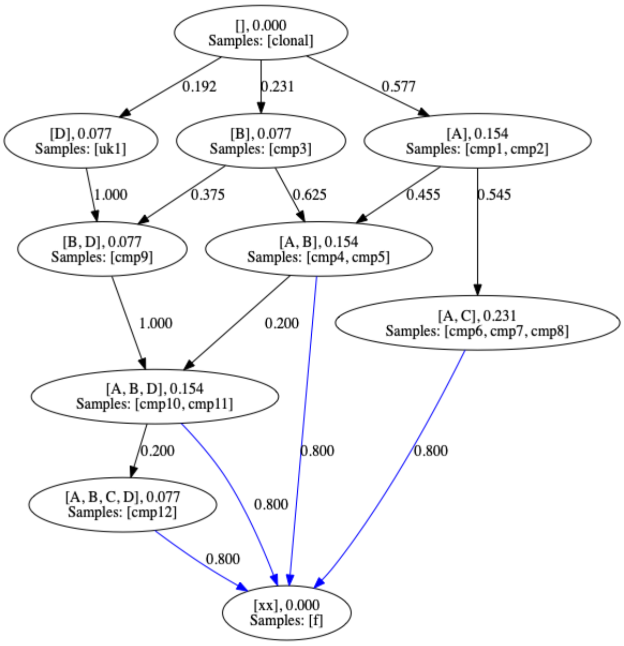

[], 0.000
Samples: [clonal]
0.192
0.231
0.577
[D], 0.077
Samples: [uk1]
[B], 0.077
Samples: [cmp3]
[A], 0.154
Samples: [cmp1, cmp2]
1.000
0.375
0.625
0.455
0.545
[B, D], 0.077
Samples: [cmp9]
[A, B], 0.154
Samples: [cmp4, cmp5]
[A, C], 0.231
Samples: [cmp6, cmp7, cmp8]
1.000
0.200
[A, B, D], 0.154
Samples: [cmp10, cmp11]
0.200
0.800
0.800
[A, B, C, D], 0.077
Samples: [cmp12]
0.800
0.800
[xx], 0.000
Samples: [f]

Figure 3.9: Application of MOCATHE, with *Damage or Diverge* type

3.3.5 Graph Dimension

The *Therapy Model* is the aggregation of the previous models, thus it's the biggest one. It's similar to the *CIMICE Model* but with the **Vertices Expansion** and the creation of the edges by **Parallel Composition** the dimension grows up. By the way, the whole dimension is made by all possible linear combinations of genes considering also the F vertex.

The dimension is growing particularly fast over the number of genes, since it's an exponential factor as shown in Theorem 7.

4

Model's Properties

4.1 Damage & Diverge Acyclicity Property

This theorem evaluates a property for both *DNA Damage* and *DNA Diverge* Therapy Models.

Theorem 1. *Let C_D be a Therapy Model (DNA Damage or DNA Diverge) and A a Cancer Model then C_D is a DAG*

Proof. C_D is the composition of A, *CIMICE Model* and the *Medicine Model*, for the theorem in Nicolò's Thesis the graph A is a DAG, which builds $C_D = \langle V_D, E_D, W_D \rangle$.
By structural induction, C_D has the same vertices of A in addition of F, the additional vertex.
Moreover, every added edge has F as target vertex which does not have "outcoming" edges (except self-loop). Therefore, the whole graph keeps DAG property. ☐

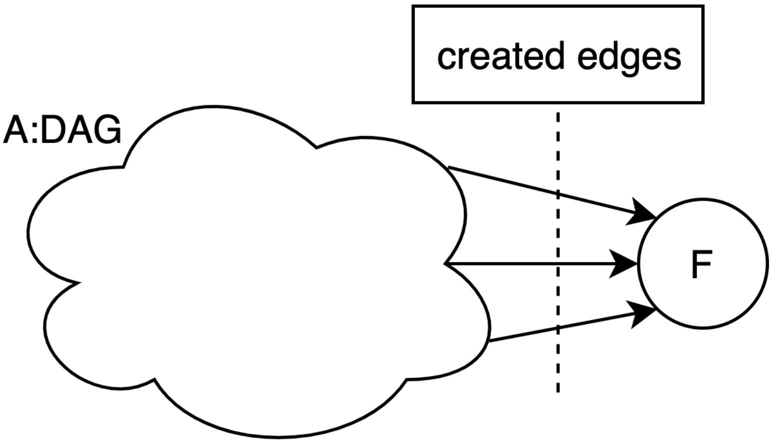


Figure 4.1: C_D graph

4.2 Repair Cyclicity Property

The following theorem verifies the dropping of DAG property in therapies with *DNA Repair* type.

Theorem 2. *Let $C_R = \langle V_R, E_R, W_R \rangle$ be a Repair Therapy Model, and be E_{AM} the edges added to C_R by Parallel Composition: $E_{AM} \neq \emptyset \implies C_R$ is not DAG*

Proof. Let $e = \langle u, v \rangle \in E_{AM}$. We show that e induces a cycle.

By *Parallel Composition* (Section 3.3.2) the Formula in DNF activates a vertex in C_R and by the formula in *Repair Therapy Model*, the target vertex u has the less genes of v. This due by the application of the φ_v formula and the reduction of genes. Since, for the **AT** assumption every superset is reached by subsets it means that there exists a path $u \rightsquigarrow v$ and when the generic edge $e = \langle v, u \rangle$ is added it means that a cycle is present in the graph. $\square$

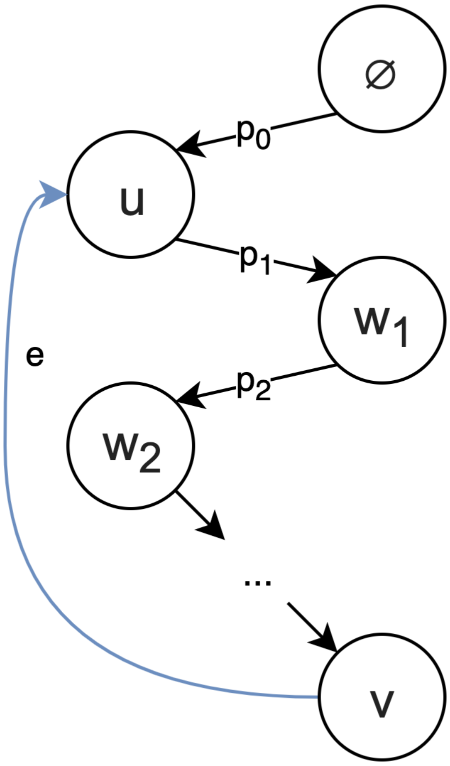


Figure 4.2: Generic activated vertex u

4.3 DTMC Properties

This theorem keeps the balancing function a probability distribution in order to maintain the Markov's Chain.

Theorem 3. *Let $C_L = \langle V_L, E_L, W_L \rangle$ be the balancing functions $W_L(e)$ keeps a probability distribution.*

Proof. The invariant of Definition 8 shows the probability distribution of CIMICE Model.

$$\left(\forall u \in V\right)\left(\sum_{e=\langle u,v\rangle} W(e) = 1\right)$$

By Parallel Composition some e_i are added. Vertices without any of these edges basically maintain the invariant, for the others we need to verify the balancing function. After e_i are applied the situation is the following.

$$\left(\forall u \in V_L \wedge \exists \theta \in \varphi : u \models \theta\right)\left(\sum_{e=\langle u,v\rangle} w(e) = 1 + p\right)$$

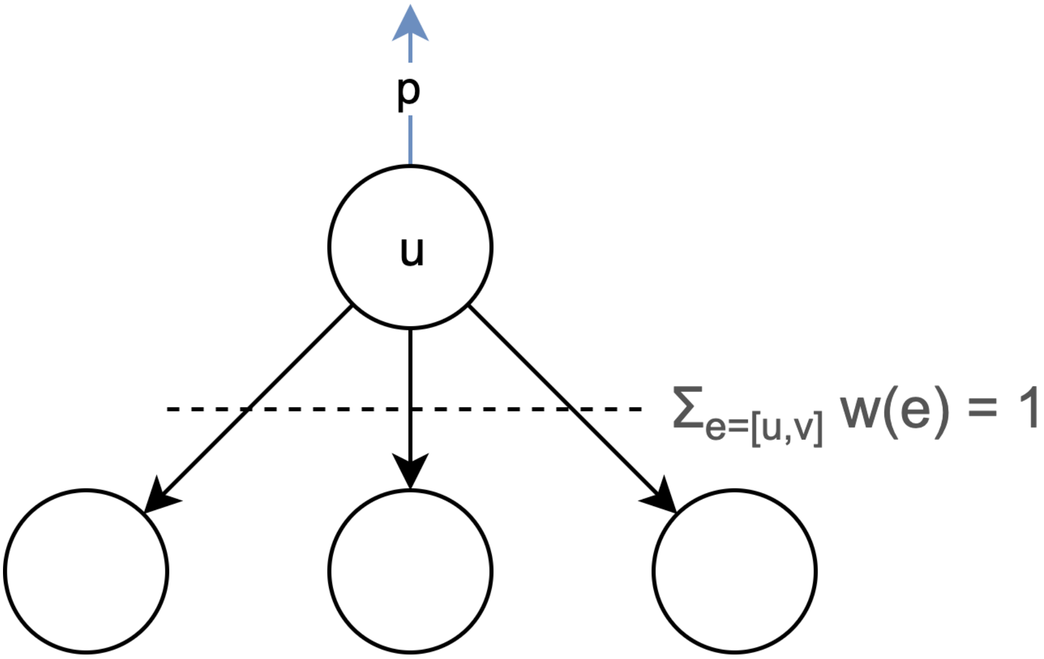


Figure 4.3: Generic activated vertex u

And applying the balancing function $W_L(e)$.

$$\Big(\forall u \in V_L \land \exists \theta \in \varphi : u \models \theta\Big)\Big(\sum_{e=\langle u,v\rangle} w(e) * (1-p) + p = 1 * (1-p) + p = 1\Big)$$

□

The following properties validates the basic **DTMC Properties** for *CIMICE* and *MOCATHE* having the Damage or Diverge Type, since the Repair Type is not a DAG as proved by Theorem 2.

Theorem 4. *Let G be a DAG with* Markov's Chain *property with n vertices and $m < n$ leafs*

- *$n - m$ are transient states*
- *m are recurrent and absorbing states*
- *n are communicating classes with cardinality equals to one*

Proof. The proof is made by each point:

- $n - m$ vertices are internal nodes, since they don't have any self loop and the G is a DAG thus every state is transient
- m vertices are leafs, since they have self loop thus every state is recurrent and absorbing
- there are n communicating classes in fact all internal nodes $n - m$ are transience states and for Theorem 1.5.5 [7] they are communicating classes, m leafs are absorbing thus communicating classes

□

4.4 Dimensional Bounds

These theorems show the dimensional bounds of MOCATHE and CIMICE.

Theorem 5. $|V_{exp}| \leq |V_A| - 1$

Proof. Let V_A be the set of vertices in $A = \langle V_A, E_A, W_A \rangle$, a generic **CIMICE Model**. Applying the V_{exp} formula to build a generic **DNA Repair Therapy Model**, for every vertex it could be created another vertex not present in the dataset. Therefore, in this case the upper bound to that equation equals to V_A and since the "clone" vertex φ_u could not be activated for the definition of Parallel Composition, we derive that equation is the upper bound for the **Expansion of Vertices**. □

Theorem 6. *let* $A = \langle V_A, E_A, W_A \rangle$ *be a CIMICE Model*
$d(A) \leq \max\{|genotypes|\}$ *where d represents the diameter of graph*

Proof. By **UN** assumption, either one mutation or in groups can be acquired. So, picking the greatest maximum size of genotypes, we derive that the diameter of the DAG could be less or equal. □

Theorem 7. *Let* $C_L = \langle V_L, E_L, W_L \rangle$ *be a generic Therapy Model and* $k = |G|$ *be the cardinality of gene set:* $|V_L| \in \mathcal{O}(2^k) \quad |E_L| \in \mathcal{O}(2^k)$

Proof. By all possible linear combination of k genes, the number of vertices is

$$|V_L| \leq 2^k + 1 \qquad +1 \text{ for the } F \text{ vertex}$$

And since the graph is oriented, the number of edges is

$$|E_L| \leq |V_L| * (|V_L| - 1) = (2^k + 1) * (2^k + 1 - 1) = 2^{2k} + 2^k$$

□

5

Synthetic Case Studies

The following examples make a real application of the definitions on different medicines applied to the following cancer cell model. The application of these models are made for the *DNA Repair* but not the other methods because the graph is the same, basically with the small difference that the created edges are pointing to the F state. By applying the definitions, we could see that it appears the exception of **Self-Loops** of Section 3.3.3.

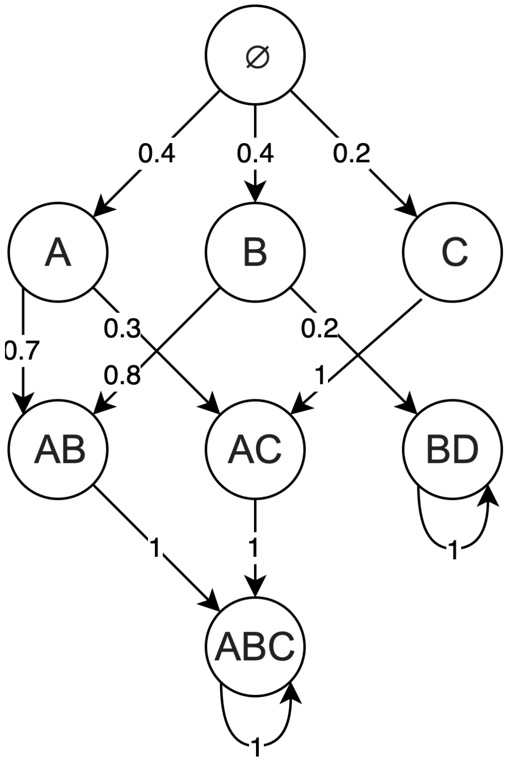


Figure 5.1: Example for cancer model

5.1 Example 1

Let the following formula be:

$$\varphi \equiv B \vee D$$

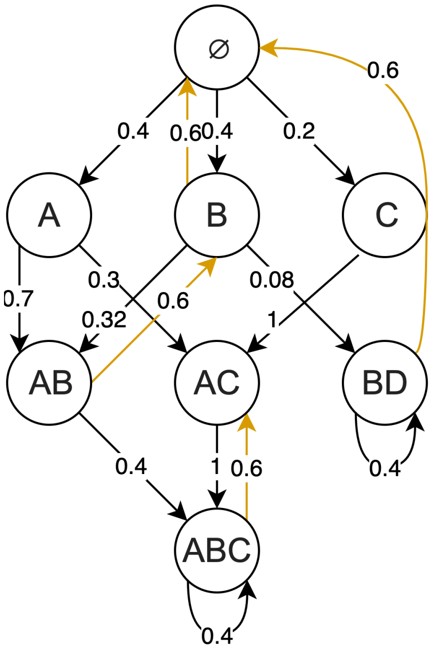

∅
A
B
C
AB
AC
BD
ABC
0.4
0.6
0.4
0.2
0.6
0.7
0.3
0.32
0.6
0.08
1
0.4
1
0.6
0.4
0.4

Figure 5.2: DNA Repair Therapy Model with $p = 0.6$

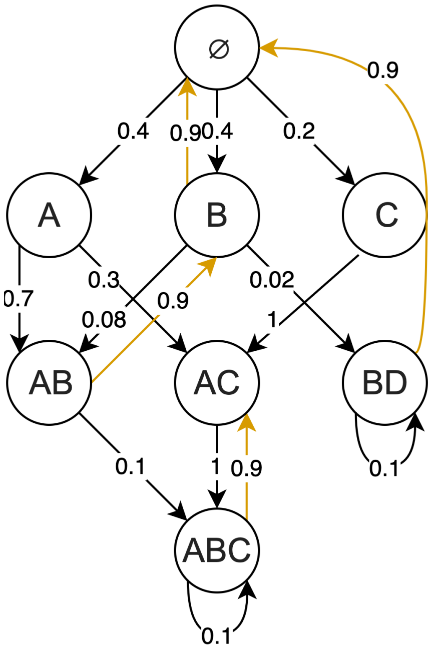

∅
A
B
C
AB
AC
BD
ABC
0.4
0.9
0.4
0.2
0.9
0.7
0.3
0.08
0.9
0.02
1
0.1
1
0.9
0.1
0.1

Figure 5.3: DNA Repair Therapy Model with $p = 0.9$

5.2 Example 2

Let the following formula be:

$$\varphi \equiv (A \land \neg C) \lor D$$

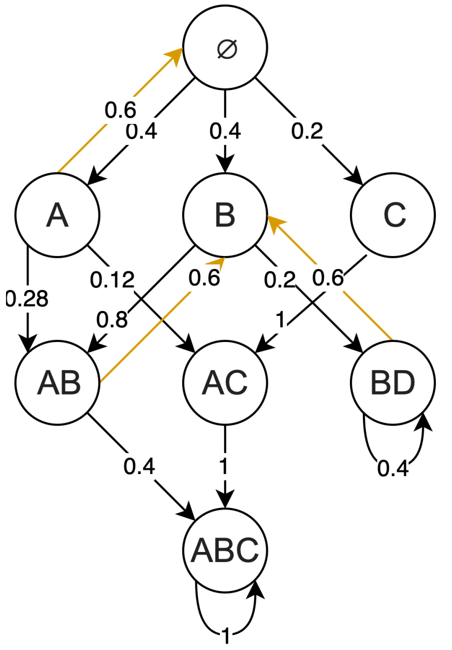

∅
0.6
0.4
0.4
0.2
A
B
C
0.12
0.6
0.2
0.6
0.28
0.8
1
AB
AC
BD
0.4
1
0.4
ABC
1

Figure 5.4: DNA Repair Therapy Model with $p = 0.6$

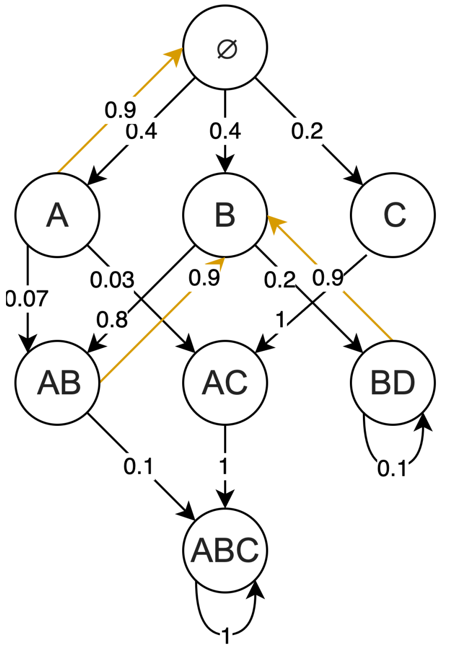

∅
0.9
0.4
0.4
0.2
A
B
C
0.03
0.9
0.2
0.9
0.07
0.8
1
AB
AC
BD
0.1
1
0.1
ABC
1

Figure 5.5: DNA Repair Therapy Model with $p = 0.9$

5.3 Example 3

Example 3 shows the **Vertices Expansion** and **Self Loops** described respectively in the Sections 3.3.1 and 3.3.3. Let the following formula be:

$$\varphi \equiv Z$$

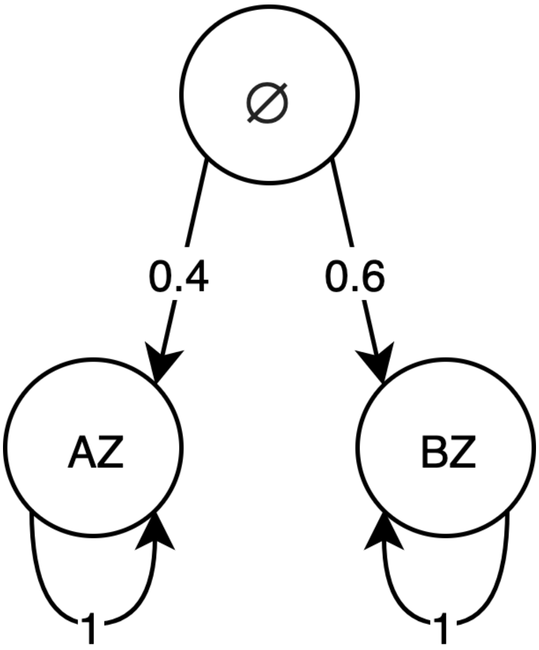

∅
0.4
0.6
AZ
BZ
1
1

Figure 5.6: Example for cancer model

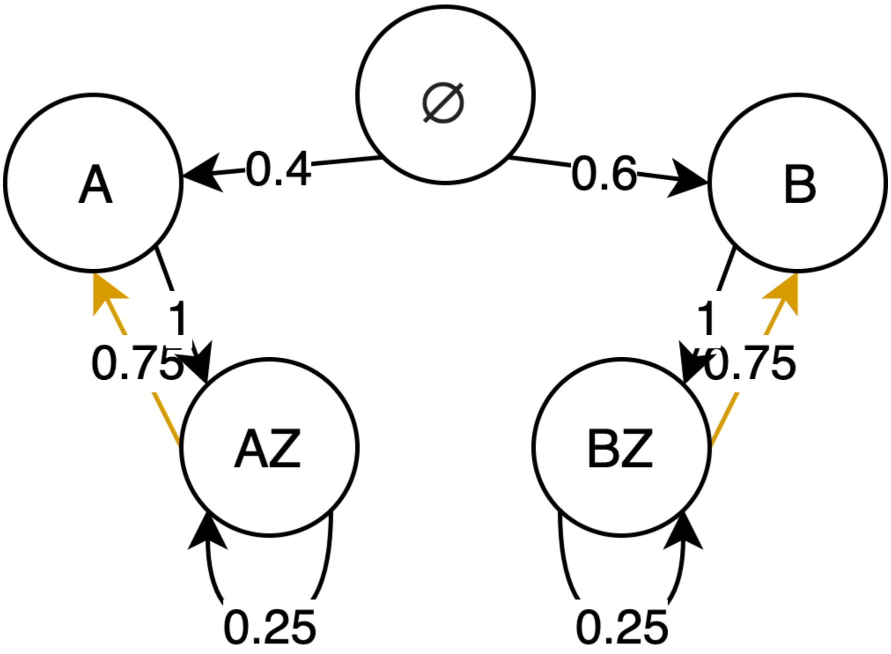

∅
A
0.4
0.6
B
1
0.75
1
0.75
AZ
BZ
0.25
0.25

Figure 5.7: DNA Repair Therapy Model with $p = 0.75$

6

Implementation

6.1 Development

The development of **MOCATHE** software was initially made in C and migrated to Java afterwards. The main reason is due to solve memory management while defining data structures, in fact very pointers were jumping over the memory and we can not resolve the issues linked to. Since Java, differently from C, manages the memory though the Java Virtual-Machine, we opted to translate from the software written in C. The code is available on GitHub.

The computation is made with these following steps:

1. analyzing the parameters
2. analyzing the DOT graph
3. setting up the graph settings
4. validating the graph

   if valid, it prints the DOT graph modified as shown in Chapter 3.2

   if not valid, it needs a re-run of CIMICE

6.1.1 Analyzing the parameters

The command line execution is made without any creation of JAR file:

```
java -cp bin main/Main -dotPathGraph -type -formula -pFormula
```

The first parameter describes the path to DOT file generated by **CIMICE**. The **-type** parameter is the type of the medicine and it could be either **-r** for Repair or **-d** for Damage and Diverge. The **formula** parameter describes the therapy formula in DNF mode with its probability of effectiveness is given by **pFormula**.

6.1.2 Analyzing the DOT graph

As explained before, the first parameters is DOT file path which is translated to the graph. Here we use some RegEx to capture the definitions of graph, vertices and edges.

6.1.3 Setting up the graph settings

In the **CIMICE** software, the DOT file parsed by this software does not contain any leaf on edges even if they exist and its value equals to 1. This decision is made assuming that all leafs contain that property, so it is useful to omit making the model simple.

In the Chapter 5 every example have at least one leaf with the self-loop with value different of 1 and thus we need to add all self-loops before computing the **MOCATHE** modification. The next step adds the edges accordingly the definition 12 and balance every edge where the source is an activated vertex.

6.1.4 Validating the graph

The previous step could spot that some target vertex is missing while adding edges: the graph is invalidated and the software asks the user to write the source file path. The same source file used by **CIMICE** to generate the DOT file now contains all genes to have target vertices while creating the edges. Therefore, after re-run **CIMICE** and **MOCATHE** the output file is completed.

We could notice here that whenever the type of Medicine is both Damage or Diverge, the graph is always valid because the target vertex F is always present (and added before).

6.2 Execution Example

We briefly discuss about the Example Thesis: a *Repair* Medicine with some missing vertices.
We recommend to install Graph Viz for a better view of graphs for the command line *dot*.

6.2.1 CIMICE

After compiling **CIMICE** into JAR file, we execute it with that example shown in GitHub obtaining the graph.

```
| => java -cp CIMICE.jar Main.CommandLineInterface -i study.dat -o study_cimice.dot -c;
| => dot -Tpng study_cimice.dot > study_cimice.png;
```

6.2.2 MOCATHE with missing vertices

Here, **MOCATHE** asks to write the .dat file path in order to add the missing vertices. In fact, by applying the formula $A \wedge B \vee C$ the missing vertex D is added to the source file.

```
| => java -cp bin main/Main study_cimice.dot -r "A & B | C" 0.8;

>  >> Need to add 1 vertices <<
> Please write the .dat file path in order to add them:
< study.dat
```

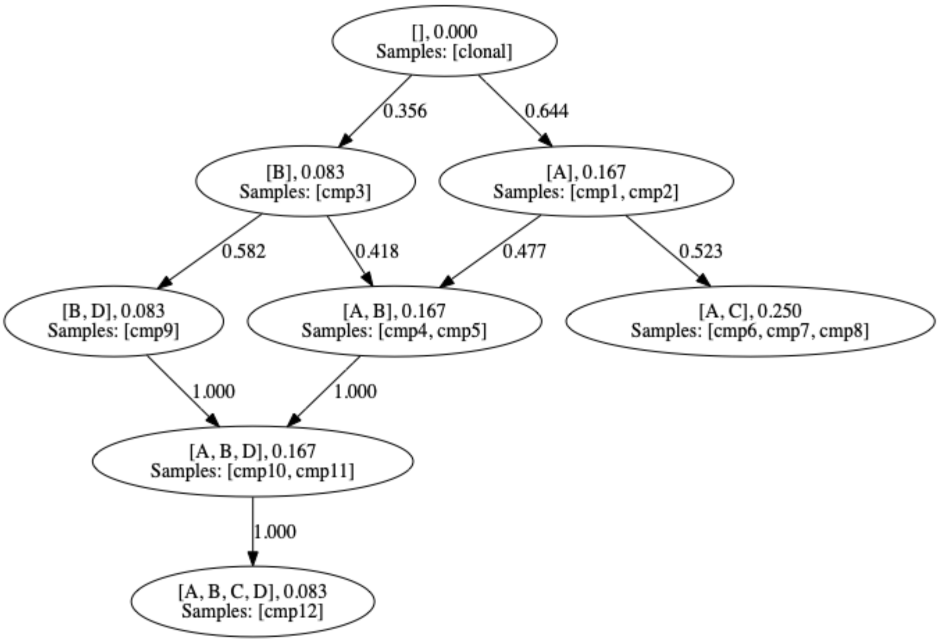

[], 0.000
Samples: [clonal]
0.356
0.644
[B], 0.083
Samples: [cmp3]
[A], 0.167
Samples: [cmp1, cmp2]
0.582
0.418
0.477
0.523
[B, D], 0.083
Samples: [cmp9]
[A, B], 0.167
Samples: [cmp4, cmp5]
[A, C], 0.250
Samples: [cmp6, cmp7, cmp8]
1.000
1.000
[A, B, D], 0.167
Samples: [cmp10, cmp11]
1.000
[A, B, C, D], 0.083
Samples: [cmp12]

Figure 6.1: CIMICE output

6.2.3 CIMICE re-run

By re-running **CIMICE** the output file will also contain the target vertices from **MOCATHE** execution.

```
| => java -cp CIMICE.jar Main.CommandLineInterface -i study.dat -o study_2_cimice.dot -c;
| => dot -Tpng study_2_cimice.dot > study_2_cimice.png;
```

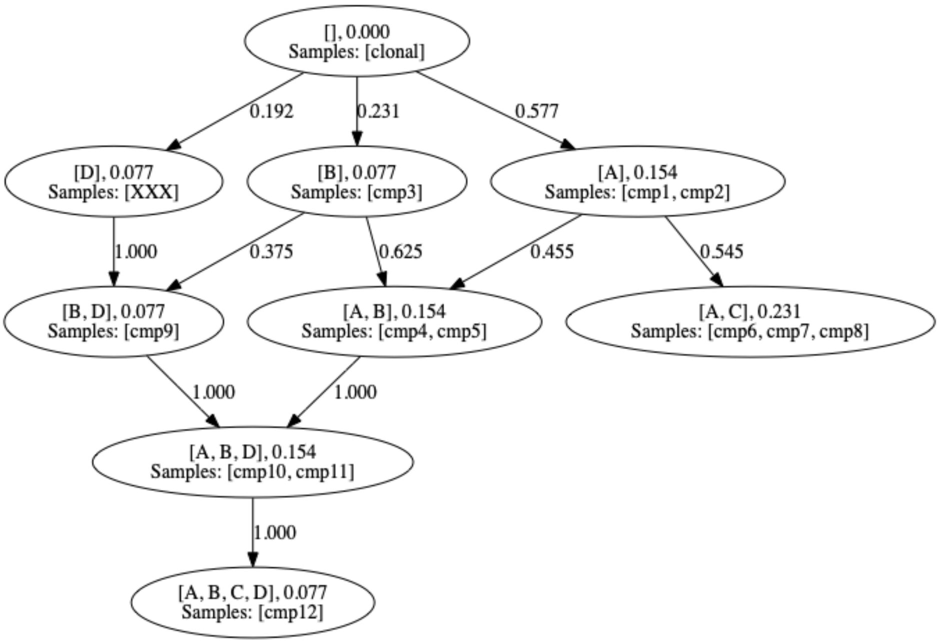

[], 0.000
Samples: [clonal]
0.192
0.231
0.577
[D], 0.077
Samples: [XXX]
[B], 0.077
Samples: [cmp3]
[A], 0.154
Samples: [cmp1, cmp2]
1.000
0.375
0.625
0.455
0.545
[B, D], 0.077
Samples: [cmp9]
[A, B], 0.154
Samples: [cmp4, cmp5]
[A, C], 0.231
Samples: [cmp6, cmp7, cmp8]
1.000
1.000
[A, B, D], 0.154
Samples: [cmp10, cmp11]
1.000
[A, B, C, D], 0.077
Samples: [cmp12]

Figure 6.2: CIMICE output with missing vertices

6.2.4 MOCATHE re-run

This is the final step when we re-run **MOCATHE** to have the correct definition of the graph with missing vertices.

```
| => java -cp bin main/Main study_2_cimice.dot -r "A & B | C" 0.8 > study_2_mocathe.dot;
| => dot -Tpng study_2_mocathe.dot > study_2_mocathe.png;
```

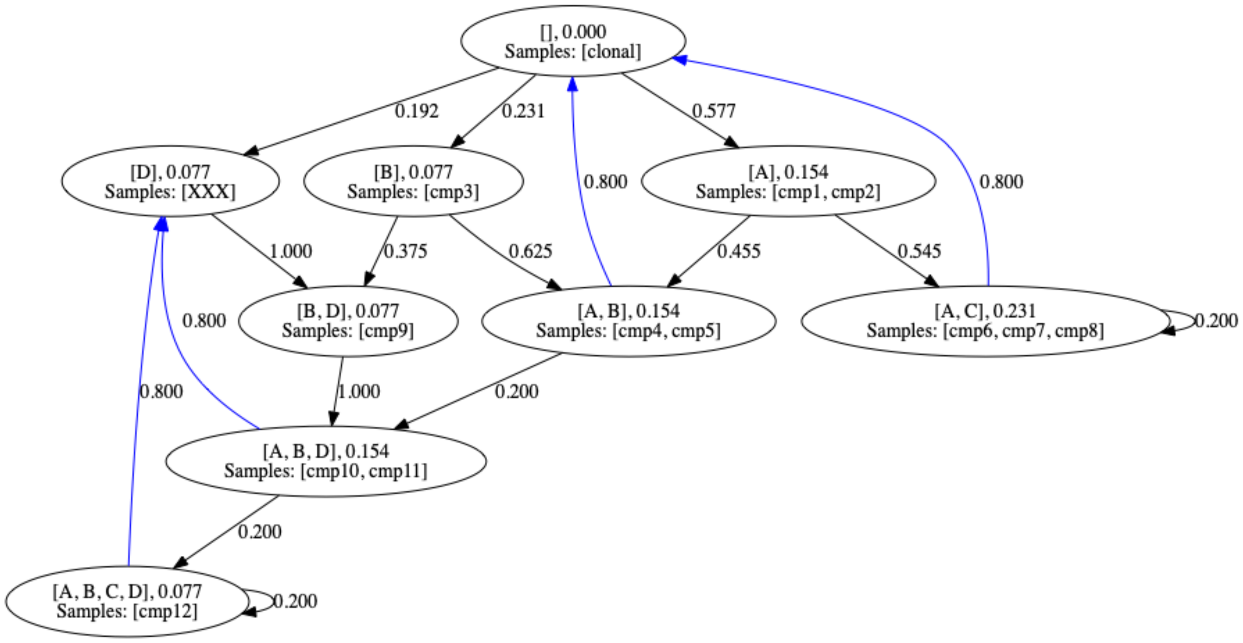

[], 0.000
Samples: [clonal]
0.192
0.231
0.577
[D], 0.077
Samples: [XXX]
[B], 0.077
Samples: [cmp3]
0.800
[A], 0.154
Samples: [cmp1, cmp2]
0.800
1.000
0.375
0.625
0.455
0.545
0.800
[B, D], 0.077
Samples: [cmp9]
[A, B], 0.154
Samples: [cmp4, cmp5]
[A, C], 0.231
Samples: [cmp6, cmp7, cmp8]
0.200
0.800
1.000
0.200
[A, B, D], 0.154
Samples: [cmp10, cmp11]
0.200
[A, B, C, D], 0.077
Samples: [cmp12]
0.200

Figure 6.3: MOCATHE output

7

Conclusion

In this thesis we presented **MOCATHE**, a model built on **CIMICE** to improve the functionality of knowing how the cancer progression is getting a change. Remembering that **CIMICE** builds an inference model of cancer phylogenetics, this implementation lets user know how this phylogenetic tree is modified by applying different types of treatments.

There are more functionalities that could enrich this implementation, in fact as explained the times of cancer could not be easily calculated even if some models are making it through proper heuristics. The implementation that enriches this software is the transformation from *DTMC* (Discrete-Time Markov Chain) to *CTMC* (Continuous-Time Markov Chain) with Jump Chain.

Sometimes doctors could apply different types of medicines within the same patient and therefore the parallel composition of multiple medicines is a good method to know how the software inferences the progression. We analyzed this as a simple computation of multiple times of this software: it is not a good procedure because the application is serial and not parallel.

In *Markov Chain* there are a method called *equilibrium* (invariant distribution) to understand how a state is reachable and its probability. An interesting application is to predict the inferences of returning at some previous state, for example the “clone” status. This means how much strong is the validity of the *Medicine*.

In the parameters passed to the software, we assume that the formula is expressed in DNF, but sometimes the user could add some Medicine formulas which is not in this form and thus a transformation is good to prevent any calculation by hand.

Bibliography

[1] Niko Beerenwinkel, Jörg Rahnenführer, Rolf Kaiser, Daniel Hoffmann, Joachim Selbig, and Thomas Lengauer. Mtreemix: a software package for learning and using mixture models of mutagenetic trees. *Bioinformatics*, 21(9):2106–2107, 2005.

[2] Niko Beerenwinkel and Seth Sullivant. Markov models for accumulating mutations. *Biometrika*, 96(3):645–661, 2009.

[3] Hossein Shahrabi Farahani and Jens Lagergren. Learning oncogenetic networks by reducing to mixed integer linear programming. *PLoS ONE*, 8(6), 2013.

[4] Moritz Gerstung, Michael Baudis, Holger Moch, and Niko Beerenwinkel. Quantifying cancer progression with conjunctive bayesian networks. *Bioinformatics*, 25(21):2809–2815, 2009.

[5] Jack Kuipers, Thomas Thurnherr, Giusi Moffa, Polina Suter, Jonas Behr, Ryan Goosen, Gerhard Christofori, and Niko Beerenwinkel. Mutational interactions define novel cancer subgroups. *Nature Communications*, 9, 2018.

[6] Netherlands Cancer Institute NKI. Logic optimization for binary input to continuous output.

[7] J. R. Norris. *Markov Chains*. Cambridge Series, Cambridge University Press, 1997.

[8] Jörg Rahnenführer, Niko Beerenwinkel, Wolfgang A. Schulz, Christian Hartmann, Andreas von Deimling, Bernd Wullich, and Thomas Lengauer. Estimating cancer survival and clinical outcome based on genetic tumor progression scores. *Bioinformatics*, 21(10):2438–2446, 2005.

[9] Navnath S.Gavande, Pamela S.VanderVere-Carozza, Hilary D.Hinshaw, Shadia I.Jalal, Catherine R.Sears, Katherine S.Pawelczak, and John J.Turchiacd. Dna repair targeted therapy: The past or future of cancer treatment? *Pharmacology & Therapeutics*, 160:65–83, 2016.

[10] Wei Zhang and Shu-Lin Wang. Inference of cancer progression with probabilistic graphical model from cross-sectional mutation data. *IEEE Access*, 6:22889–22898, 2018.