UNIVERSIDAD DE CONCEPCIÓN
Dirección de Postgrado
Facultad de Ingeniería – Programa de Doctorado en Ciencias de la Computación

Study of the block-sequential operator on Boolean networks. Application to discrete network analysis

Tesis para optar al grado de
Doctor en Ciencias de la Computación
Por

LUIS CABRERA CROT
Concepción, Chile

Profesores guía: **Julio Aracena Lucero**
Departamento de Ingeniería Matemática
Facultad de Ciencias Físicas y Matemáticas
Universidad de Concepción
Adrien Richard
CNRS Researcher
Laboratoire I3S - UMR CNRS 7271
Université Côte d'Azur, France.
Lilian Salinas Ayala
Depto. de Ing. Informática y Cs. de la Computación
Facultad de Ingeniería
Universidad de Concepción

©

Se autoriza la reproducción total o parcial, con fines académicos, por cualquier medio o procedimiento, incluyendo la cita bibliográfica del documento.

Acknowledgments

A Tomás y Yohana, por ser mi motor a seguir adelante.

A mis padres, mis abuelos y el resto de mis familiares, por su apoyo incondicional.

A mis profesores guías, por su infinita paciencia.

Al resto de profesores, colegas, amigos y compañeros, por esos "Vamos que se puede" que siempre son necesarios.

Abstract

A Boolean network is a system of n interacting Boolean variables, which evolve, in a discrete time, according to a regulation rule and to a predefined updating scheme. They have applications in many areas, including circuit theory, computer science, social networks and biological systems. The structure of such a network is often represented by a digraph, called interaction digraph, where vertices are network components, and where there is an arc from one component to another when the evolution of the latter depends on the evolution of the former.

The relationship between the structure of a regulatory network and its dynamical behavior is crucial to understand for instance how and why biological networks have evolved. Further, this relationship can be used to construct networks with desirable dynamical properties.

In the original scheme of a Boolean network all the nodes are synchronously updated at each time step (this scheme is also called parallel schedule). A more general scheme, introduced in [67], is to consider that the set of network nodes is partitioned into blocks and that the nodes in a block are updated simultaneously. Differences in the dynamical behaviors of Boolean networks with different update schedules has been studied mainly from an experimental and statistical point of view.

In this tesis, the variations of the interaction digraph of a Boolean network with respect to changes in the update schedule and its relation with some dynamical properties of the network are studied.

In order to achieve this goal, three main topics are discussed.

First, the variations in the parallel digraph of some structural characteristics (number of strongly connect components, transversal number, packing number) with respect to changes in the update schedule are analyzed.

Second, an algorithm is constructed to find the fixed points of a Boolean network taking advantage of knowledge about the upper bound of the fixed points of a network, in this case we use the positive transversal number.

Finally, a new, so far unexplored problem is defined, which states that given a Boolean network f, find a Boolean network h and an update schedule s that are dynamically equivalent to f. In this sense, several variations of the original problem are presented, many of which can be solved in polynomial time.

Contents

List of Figures

Chapter 1

Introduction

1.1 English version

A Boolean network is a system of n interacting Boolean variables, which evolve, in a discrete time, according to an evolution rule and to a predefined updating schedule. The structure of such network is often represented by a digraph, called interaction or regulation graph, where the vertices are network components, and there is an arc from one component to another when the evolution of the latter depends on the evolution of the former.

Boolean networks have many applications, including circuit theory, computer science and social systems. In particular, from the seminal works of Kauffman [45, 46] and Thomas [78, 79], they are extensively used as models of gene networks. Among the many dynamical properties that can be studied, the attractors of the network, such that limit cycles and fixed points (steady states) are of special interest because they are associated to distinct types of cells defined by patterns of gene activity. For example, the limit cycles are often associated with mitotic cycles in cells [14, 42, 45].

In the original scheme of a Boolean network all the nodes are updated at each time step, in parallel (this scheme is also called synchronous updating). A more general scheme, introduced in [67] and named block-sequential update schedule, is to consider that the set of network nodes is partitioned into blocks and that the nodes in a block are updated simultaneously, the blocks being considered in a given sequence. This generalizes the previous case because the parallel case corresponds to a single block. It also generalizes the sequential update schedules, where every node is updated in a defined sequence at every time step. Several theoretical studies about the dynamical behavior of Boolean networks with block-sequential update schedules have been made, especially in some particular families of networks, for example: neural networks [35, 38, 57], cellular automata [31, 32, 33] and sequential dynamical systems [18, 19, 56, 58, 62].

Differences in the dynamical behaviors of Boolean networks with different update schedules has been mainly studied from an experimental and statistical point of view [22, 26, 29]. Some recent theoretical studies in this regard are exhibited in [5, 13, 20, 31, 34, 35, 57].

The relationship of regulatory network structural properties to dynamical properties is crucial to understand for instance how and why biological networks have evolved. Further, this relationship can be used to construct networks with desirable dynamical properties. It is well known that the architecture of the interaction digraph influences the dynamical behavior of the network. For example, a Boolean network with an acyclic interaction digraph has only one fixed point [67], and more generally, the number of fixed points in a Boolean network is bounded by 2^τ, where τ is the minimum cardinal of a feedback vertex set of its interaction digraph [7]. More recently, in [16] there are new upper and lower bounds for the number of fixed points in monotone Boolean

networks based in the packing number of the interaction digraph.

On the other hand, there are studies where it can be found a relationship between architecture and dynamical properties for some particular kind of Boolean networks. In this line we can mention that in a disjunctive Boolean network, the existence and the length of the limit cycles is controlled by the index of cyclicity of its interaction digraph, that is the greatest common divisor of the length of the cycles of the digraph. In particular, in [21, 24, 43] it was proved that in conjunctive Boolean networks, if the index of cyclicity is one then the dynamical behavior has only the two trivial fixed points and no limit cycles, they also proved that if the index of cyclicity is greater than one, the length of every limit cycle is a divisor of this index.

1.1.1 Problematic

The general objective of this thesis is to study the variations of the interaction digraph of a Boolean network with respect to changes in the update schedule and its relation with some dynamic properties of the network. In this way, we hope to better understand the possible dynamic behaviors of a Boolean network f with different update schedules. For this purpose, we focus on studying three main aspects.

Transversal number and packing number

As seen throughout the literature, the dynamics of each strongly connected component plays an important role when studying the attractors of the dynamics (fixed points and limit cycles) [7, 43]. In addition, there are parameters that present us with interesting information when analyzing the interaction graph. For example, the transversal number of a digraph G is related to the maximum number of fixed points that a Boolean network f whose interaction graph is G can have [7, 66]. Besides, the packing number is related to the maximum number of fixed points when the associated network is monotonic [16]. Therefore, as the parallel digraph of a labeled digraph can be considered as the interaction graph of a new Boolean network, we are interested in studying how strongly connected components work when obtaining the parallel digraph.

Algorithm to find the fixed points in Boolean networks

In the modeling of biological systems by Boolean networks, a key problem is finding the set of fixed points of a given network. Some algorithms are based on transforming the fixed point problem, such as [80, 82, 84], which uses the reduction method, or [41, 86], which represents Boolean networks as polynomial functions, or [70, 3, 27, 28, 53, 76, 77], which uses methods based on SAT solvers, or [1], which uses methods based on integer programming. Other types of algorithms consider certain structural properties of the regulatory graph such as those proposed by [2, 85], which consider a feedback vertex set of the graph. However, these methods do not take into account the type of interaction (activation and inhibition) between its components. For this reason, this thesis addresses a new strategy by combining upper bound results for fixed points in Boolean graphs [7, 66] with results seen in [67], such as that fixed points are invariant to the update schedule applied to a network and that networks without cycles have a single fixed point.

Dynamic equivalence between Boolean networks

Parallel digraphs, which are preliminary presented by F. Robert [67, 68], calling them Gauss-Seidel operator, are a widely used tool over the years. Thanks to [36, 12], equivalence classes have been

defined between different update schedules based on their update digraph, so that elements in the same class have the same dynamic behavior.

In this sense, in [21, 10, 8, 34, 37] several investigations have been carried out on the properties of these parallel digraphs with respect to the networks that generate them.

However, to our knowledge, the following questions have been little explored: What other networks have the same dynamics as that of a given network? What dynamics are only yielded by a parallel schedule?

1.1.2 Organization of the document

This thesis is organized as follows:

In first place, in Chapter 2 the definitions and notations that are used throughout this thesis are presented. If any particular notation is needed in any chapter, it is presented at the beginning of the chapter.

Then, in Chapter 3 we focus on some preliminary results on the structure of parallel digraphs that are helpful for the rest of this thesis. First, we can conclude that the number of strongly connected components of a digraph G is conserved since for each of these components, an associated component is formed in the parallel digraph that has the same or fewer vertices than the original component but in no case disappear. Secondly, the transversal number (τ) of the interaction graph is analyzed, because it is known that it is related to the upper bound for the number of fixed points. Therefore, we study how τ change when obtaining the parallel digraph. The result is that, when obtaining the parallel digraph, the τ of the original digraph is maintained or increased. Furthermore, to prevent τ from increasing uncontrollably when calculating the parallel digraph, an update schedule has been defined that maintains the transversal number in the parallel digraph. Finally, the study focuses on the packing number (ν), a value that gives us a bound for the fixed points in monotonous networks. As in the case of the transversal number, the packing number is also maintained or increased. If the ν of a graph is equal to its τ, using the previous labeling function we can control the ν. And even more, for the case of the complete graph K_n whose ν is $\frac{n}{2}$, a different labeling function is presented that maintains that ν.

Subsequently, in Chapter 4, we propose a new algorithm for finding the set of fixed points of a Boolean network, based on a positive feedback vertex set P of its regulatory graph and which works, by applying a sequential update schedule, in time $O(2^{|P|} \cdot n^{2+k})$, where n is the number of components and the regulatory functions of the network can be evaluated in time $O(n^k), k \geq 0$. The theoretical foundation of this algorithm is due a nice characterization, that we give, of the dynamical behavior of the Boolean networks without positive cycles and with a fixed point. In addition, a polynomial algorithm is presented to find a PFVS (not necessarily minimal) and a minimal FVS containing it. The results of this chapter have been published in [9].

In Chapter 5 we define the problem of finding a Boolean network that is dynamically equivalent to another network, i.e., given a Boolean network f find another Boolean network h and an update schedule s (not equivalent to the parallel schedule), such that $h^s = f$. For the general case, we have two important results. First, we prove that if there exists a Boolean network h and an update schedule s (of block-sequential type with k blocks), one can always construct a solution with an update schedule s' with two blocks and a Boolean network h' resulting from applying modifications to the network h. Also, it was successfully shown that the problem itself is NP-Hard. Subsequently, in order to find a way to a better solution, we restrict the problem to disjunctive Boolean networks, and with this change we found that, if there is two or more vertices sharing the input neighborhood, it is easy to find a solution to the problem. If this does not occur, however, a polynomial algorithm

is developed that manages to give a solution to the problem (when it has one) or indicate that there is no solution when there is none. The results of this chapter have been submitted to a journal for publication.

Following the idea of the previous chapter, in Chapter 6 we analyze particular versions of the dynamic equivalence problem and how they can be solved. For example, if the update schedule is fixed, the dynamic equivalence problem in disjunctive networks can be solved in polynomial time. Moreover, all networks satisfying dynamic equivalence for a network f and an update schedule s can be found with polynomial delay. Besides, if the Boolean network h is fixed, finding an update schedule s such that h^s is dynamically equivalent to f can be found in polynomial time. Moreover, a particular case is generated when $h = f$ and a new algorithm is generated which, in polynomial time, is able to find a two-block update schedule s such that $h^s = f$. The results of this chapter will be submitted to a future publication.

Finally, in Chapter 7 we summarize the results of this research.

1.2 Versión en español

Una red Booleana es un sistema de n variables Booleanas que interactúan entre sí, que evolucionan en un tiempo discreto, de acuerdo a una regla de regulación y un esquema de actualización predefinido. La estructura de una red de este tipo suele representarse mediante un digrafo, denominado grafo de interacción o de regulación, en el que los vértices son componentes de la red, y existe un arco de un componente a otro cuando la evolución de este último depende de la evolución del primero.

Las redes Booleanas tienen muchas aplicaciones, entre ellas la teoría de circuitos, la informática y los sistemas sociales. En particular, desde los trabajos seminales de Kauffman [45, 46] y Thomas [78, 79], se utilizan ampliamente como modelos de redes genéticas. Entre las muchas propiedades dinámicas que pueden estudiarse, los atractores de la red, como los ciclos límite y los puntos fijos (estados estacionarios) son de especial interés porque están asociados a distintos tipos de células definidas por patrones de actividad génica. Por ejemplo, los ciclos límite suelen estar asociados a los ciclos mitóticos de las células [14, 42, 45].

En el esquema original de una red Booleana todos los nodos se actualizan en cada paso de tiempo, en paralelo (este esquema también se denomina actualización sincrónica). Un esquema más general, introducido en [67] y denominado esquema de actualización bloque-secuencial, consiste en considerar que el conjunto de nodos del grafo se particiona en bloques y que los nodos de un bloque se actualizan simultáneamente, considerándose los bloques en una secuencia determinada. Esto generaliza el caso anterior, ya que el caso paralelo corresponde a un único bloque. También generaliza los esquemas de actualización secuencial, en los que cada nodo se actualiza en una secuencia definida en cada paso temporal. Se han realizado varios estudios teóricos sobre el comportamiento dinámico de las redes Booleanas con esquemas de actualización secuencial por bloques, especialmente en algunas familias particulares de redes, por ejemplo: redes neuronales [35, 38, 57], autómatas celulares [31, 32, 33] y sistemas dinámicos secuenciales [18, 19, 56, 58, 62].

Las diferencias en los comportamientos dinámicos de redes Booleanas con diferentes esquemas de actualización se han estudiado principalmente desde un punto de vista experimental y estadístico [22, 26, 29]. Algunos estudios teóricos recientes a este respecto se exponen en [5, 13, 20, 31, 34, 35, 57].

La relación entre las propiedades estructurales de las redes reguladoras y sus propiedades dinámicas es crucial para entender, por ejemplo, cómo y por qué han evolucionado las redes

biológicas. Además, esta relación puede utilizarse para construir redes con propiedades dinámicas deseables. Es bien sabido que la arquitectura del digrafo de interacción influye en el comportamiento dinámico de la red. Por ejemplo, una red Booleana con un digrafo de interacción acíclico tiene sólo un punto fijo [67], y más en general, el número de puntos fijos en una red Booleana está limitado por 2^τ, donde τ es el cardinal mínimo de un conjunto recubridor de ciclos de su digrafo de interacción [7]. Más recientemente, en [16] hay nuevos límites superior e inferior para el número de puntos fijos en redes Booleanas monótonas basadas en el número de empaquetamiento del digrafo de interacción.

Por otro lado, existen estudios donde se puede encontrar una relación entre arquitectura y propiedades dinámicas para algún tipo particular de redes Booleanas. En esta línea podemos mencionar que en un grafo Booleano disyuntivo, la existencia y la longitud de los ciclos límite está controlada por el índice de ciclicidad de su digrafo de interacción, que es el máximo común divisor de la longitud de los ciclos del digrafo. En particular, en [21, 24, 43] se demostró que en las redes Booleanas conjuntivos, si el índice de ciclicidad es uno entonces el comportamiento dinámico tiene sólo los dos puntos fijos triviales y ningún ciclo límite, también se probó que si el índice de ciclicidad es mayor que uno, la longitud de cada ciclo límite es un divisor de este índice.

1.2.1 Problemática

El objetivo general de esta tesis es estudiar las variaciones del grafo de interacción de una red Booleana con respecto a cambios en el esquema de actualización y su relación con algunas propiedades dinámicas de la red. De esta forma, esperamos comprender mejor los posibles comportamientos dinámicos de una red Booleana f con diferentes esquemas de actualización. Para ello, nos centramos en el estudio de tres aspectos principales.

Número transversal y número de empaquetamiento

Como se ha visto a lo largo de la literatura, la dinámica de cada componente fuertemente conectada juega un papel importante a la hora de estudiar los atractores de la dinámica (puntos fijos y ciclos límite) [7, 43]. Además, existen parámetros que nos aportan información interesante a la hora de analizar el grafo de interacción. Por ejemplo, el número transversal de un digrafo G está relacionado con el número máximo de puntos fijos que puede tener una red Booleana f cuyo grafo de interacción sea G [7, 66]. Además, el número de empaquetamiento está relacionado con el número máximo de puntos fijos cuando la red asociada es monótona [16]. Por tanto, como el digrafo paralelo de un digrafo etiquetado puede considerarse el grafo de interacción de una nueva red Booleana, nos interesa estudiar cómo funcionan las componentes fuertemente conectadas al obtener el digrafo paralelo.

Algoritmo para encontrar los puntos fijos en redes Booleanas

En el modelado de sistemas biológicos mediante redes Booleanas, un problema clave es encontrar el conjunto de puntos fijos de una red dada. Algunos algoritmos se basan en la transformación del problema de los puntos fijos, como [80, 82, 84], que utiliza el método de reducción, o [41, 86], que representa las redes Booleanas como funciones polinomiales, o [70, 3, 27, 28, 53, 76, 77], que utiliza métodos basados en solucionadores de SAT, o [1], que utiliza métodos basados en programación entera. Otros tipos de algoritmos consideran ciertas propiedades estructurales del grafo regulatorio, como los propuestos por [2, 85], que consideran un conjunto recubridor de ciclos del grafo. Sin

embargo, estos métodos no tienen en cuenta el tipo de interacción (activación e inhibición) entre sus componentes. Por ello, esta tesis aborda una nueva estrategia combinando resultados de cotas superiores de puntos fijos en redes Booleanas [7, 66] con resultados vistos en [67], tales como que los puntos fijos son invariantes respecto al esquema de actualización aplicado y que una red sin ciclos tiene un único punto fijo.

Equivalencia dinámica entre grafos Booleanos

Los digrafos paralelos, presentados de forma preliminar por F. Robert [67, 68], denominándolos operador de Gauss-Seidel, son una herramienta ampliamente utilizada a lo largo de los años. Gracias a [36, 12], se han definido clases de equivalencia entre distintos esquemas de actualización en función de su digrafo de actualización, de forma que los elementos de una misma clase tienen el mismo comportamiento dinámico.

En este sentido, en [21, 10, 8, 34, 37] se han realizado diversas investigaciones sobre las propiedades de estos digrafos paralelos respecto a las redes que los generan.

Sin embargo, hasta donde sabemos, las siguientes preguntas han sido poco exploradas: ¿Qué otras redes tienen la misma dinámica que la de una red dada? ¿Qué dinámicas sólo se obtienen con un esquema paralelo?

1.2.2 Organización del documento

Esta tesis está organizada de la siguiente manera:

En primer lugar, en el Capítulo 2 se presentan las definiciones y notaciones que se utilizan a lo largo de esta tesis. Si alguna notación en particular es necesaria en algún capítulo, se presenta al principio del mismo.

Entonces, en el Capítulo 3 nos centramos en algunos resultados preliminares sobre la estructura de los digrafos paralelos que son útiles para el resto de esta tesis. En primer lugar, podemos concluir que el número de componentes fuertemente conexas de un digrafo G se conserva ya que para cada una de estas componentes se forma una componente asociada en el digrafo paralelo que tiene igual o menos vértices que la componente original pero en ningún caso desaparece. En segundo lugar, se analiza el número transversal (τ) del grafo de interacción, ya que se sabe que está relacionado con la cota superior para el número de puntos fijos. Por tanto, se estudia cómo cambia τ al obtener el digrafo paralelo. El resultado es que, al obtener el digrafo paralelo, el τ del digrafo original se mantiene o aumenta. Además, para evitar que τ aumente descontroladamente al calcular el digrafo paralelo, se ha definido un esquema de actualización que mantiene el número transversal en el digrafo paralelo. Por último, el estudio se centra en el número de empaquetamiento (ν), valor que nos da una cota para los puntos fijos en grafos monótonos. Como en el caso del número transversal, el número de empaquetamiento también se mantiene o aumenta. Si el ν de un grafo es igual a su τ, utilizando la función de etiquetado anterior podemos controlar el ν. Y aún más, para el caso del grafo completo K_n cuyo ν es $\frac{n}{2}$, se presenta una función de etiquetado diferente que mantiene ese ν.

Posteriormente, en el Capítulo 4, proponemos un nuevo algoritmo para encontrar el conjunto de puntos fijos de una red Booleana, basado en un FVS positivo de su grafo regulatorio y que funciona, aplicando un esquema de actualización secuencial, en tiempo $O(2^{|P|} \cdot n^{2+k})$, donde n es el número de componentes y las funciones regulatorias del grafo pueden evaluarse en tiempo $O(n^k), k \geq 0$. El fundamento teórico de este algoritmo se debe a una buena caracterización, que damos, del comportamiento dinámico de las redes Booleanas sin ciclos positivos y con un punto

fijo. Además, se presenta un algoritmo polinomial para encontrar un PFVS (no necesariamente minimal) y un FVS minimal que lo contenga. Los resultados de este capítulo se han publicado en [9].

En el Capítulo 5 definimos el problema de encontrar una red Booleana que sea dinámicamente equivalente a otro red, es decir, dado una red Booleana f encontrar otra red Booleana h y un esquema de actualización s (no equivalente al esquema paralelo), tal que $h^s = f$. Para el caso general, tenemos dos resultados importantes. En primer lugar, demostramos que si existe una red Booleana h y un esquema de actualización s (de tipo bloque-secuencial con k bloques), siempre se puede construir una solución con un esquema de actualización s' con dos bloques y una red Booleana h' resultante de aplicar modificaciones a la red h. Además, se demostró con éxito que el problema en sí es NP-Hard. Posteriormente, para encontrar un camino hacia una mejor solución, restringimos el problema a red Booleanas disyuntivas, y con este cambio encontramos que, si hay dos o más vértices con la misma vecindad de de entrada, es fácil encontrar una solución al problema. Si esto no ocurre, sin embargo, se desarrolla un algoritmo polinomial que consigue dar una solución al problema (cuando la tiene) o indicar que no hay solución cuando no la hay. Los resultados de este capítulo se han enviado a una revista para su publicación.

Siguiendo la idea del capítulo anterior, en el Capítulo 6 analizamos versiones particulares del problema de equivalencia dinámica y cómo pueden resolverse. Por ejemplo, si el esquema de actualización es fijado, el problema de equivalencia dinámica en grafos disyuntivos puede resolverse en tiempo polinomial. Además, todos las redes que satisfacen la equivalencia dinámica para un red f y un esquema de actualización s pueden encontrarse con delay polinomial. Además, si la red Booleana h es fijada, se puede encontrar en tiempo polinomial un esquema de actualización s tal que h^s sea dinámicamente equivalente a f. Además, se genera un caso particular cuando $h = f$ y se genera un nuevo algoritmo que, en tiempo polinomial, es capaz de encontrar un esquema de actualización de dos bloques s tal que $h^s = f$. Los resultados de este capítulo se presentarán en una futura publicación.

Finalmente, en el Capítulo 7 se consolidan los resultados de esta investigación.

Chapter 2

Definitions and Notations

A *Boolean network* (BN) with n components is a discrete dynamical system usually defined by a *global transition function*:

$$f : \mathbb{B}^n \to \mathbb{B}^n,\ x \mapsto f(x) = (f_1(x), \ldots, f_n(x)),$$

where $\mathbb{B} = \{0, 1\}$ and each function $f_u : \mathbb{B}^n \to \mathbb{B}$ associated to the component u is called *local activation function* (also known as regulatory function).

Any vector $x = (x_1, \ldots, x_n) \in \mathbb{B}^n$ is called a *state* of the network f with *local state* x_u on each component u.

In the sequel, we denote $[n] = \{1, \ldots, n\}$, for any integer n.

An *update schedule* is defined by a function $s : [n] \to [n]$ such that $s([n]) = [m]$ for some $m \leq n$, where $s(u)$ indicates the updating order of the component u in a time step. A *block* of an update schedule s is a set $B_i = \{u \in [n] : s(u) = i\}$, with $i \in [m]$. An update schedule s is also denoted by $s = B_1, B_2, \ldots, B_m$. If $m = 1$, the update schedule is called *synchronous or parallel* (e.g., $\{1, 2, 3, 4, 5\}$). If $m = n$, the update schedule is called *sequential* (e.g., $\{1\}\,\{2\}\,\{3\}\,\{4\}\,\{5\}$). Other kinds of update schedules can be named as *block-sequential updates* (e.g., $\{1, 2\}\,\{3\}\,\{4, 5\}$). In the sequel, we denote s_p to the parallel update schedule.

2.1 Interaction digraph

We define the *interaction graph* of a Boolean network $f : \mathbb{B}^n \to \mathbb{B}^n$, denoted by $G(f) = ([n], A(f))$, as:

$$\begin{aligned} [n] &= \{1, \ldots, n\}, \\ A(f) &= \{(u, v) \in [n] \times [n] : \exists x \in \mathbb{B}^n, f_v(x) \neq f_v(x^{\neg u})\} \end{aligned}$$

where $\forall i \in [n],\ x_i^{\neg u} = x_i \iff u \neq i$.

Also, for each $u \in [n]$ we define the in-neighborhood and the out-neighborhood of u as:

$$\begin{aligned} N_f^-(u) &= \{v \in [n] : (v, u) \in A(f)\} \\ N_f^+(u) &= \{v \in [n] : (u, v) \in A(f)\} \end{aligned}$$

And the in-degree and the out-degree of u as:

$$d^-(u) = |N_f^-(u)| \qquad\qquad d^+(u) = |N_f^+(u)|$$

If $d^-(u) = 0$, we say that u is a *source vertex*. The degree of u is $d(u) = d^-(u) + d^+(u)$.

Definition 2.1. We say that a vertex $u \in V$ is a *pendant node* if $d^+(u) = 0$.

Example 2.1. An example of a Boolean network f and its interaction graph $G(f)$ is shown in Figure 2.1.

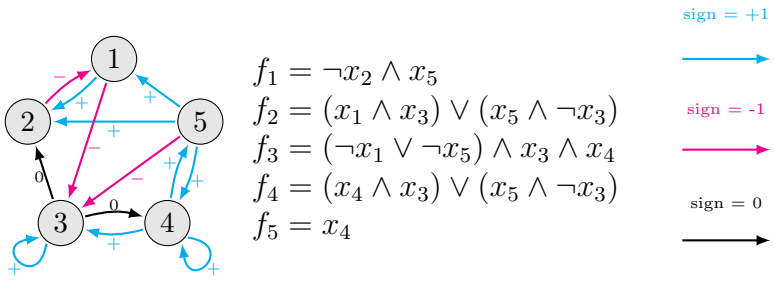


Figure 2.1: Example of a Boolean network and its interaction graph.

If for some arc $(u, v) \in A(f)$, there exists $x \in \mathbb{B}^n$, such that $x_u = 0$ and $f_v(x) < f_v(x + e_u)$, where $e_u \in \mathbb{B}^n$ denotes the binary vector with all entries equal to 0, except for entry u, which equals 1; then, we say that f_v *is increasing monotone on input* u and the sign of (u, v) is $+1$. Otherwise, if for some arc $(u, v) \in A(f)$, there exists $x \in \mathbb{B}^n$, such that $x_u = 0$ and $f_v(x) > f_v(x + e_u)$ we say that f_v *is decreasing monotone on input* u and the sign of (u, v) is -1.

For easy notation, the vertex set of $G(f)$ is referred to as V and its arc set as A. If $G(f)$ does not have multiple signs from one vertex to another, then we say that f is a regulatory Boolean network (RBN).

A *walk* from a vertex v_0 to a vertex v_l in the interaction graph $G(f)$ is a sequence of vertices and arcs $W = v_0, a_0, v_1, \ldots, a_{l-1}, v_l$ of $G(f)$ such that $\forall i \in \{0, \ldots, l-1\}$, $a_i = (v_i, v_{i+1}) \in A$. A *path* is a walk without repetition of vertices (except eventually the extreme ones). A *circuit* is a walk without repetition of arcs and closed (i.e. its extreme vertices are equal). A *cycle* is a closed path.

The sign of a walk (path, circuit or cycle) is the product of the signs of its arcs. We say that a cycle is a positive cycle, if the sign of the cycle is $+1$, otherwise, we say that is a negative cycle. For example, in Figure 2.1 the sign of the cycle $\{4, 5, 4\}$ is $+1$ because $+1 \cdot +1 = +1$ and the sign of the cycle $\{1, 2, 1\}$ is -1 because $+1 \cdot -1 = -1$.

In the sequel, we will refer to the signed cycles of a BN f as the cycles of $G(f)$.

We refer [17] for other basic definitions in digraphs.

2.2 Labeled digraph

Definition 2.2. Let $G = (V, A)$ be a digraph and s an update schedule. We define the label function $\text{lab}_s : A \to \{\ominus, \oplus\}$ in the following way:

$$\forall (j, i) \in A \quad \text{lab}_s(j, i) = \begin{cases} \oplus & \text{if } s(j) \geq s(i) \\ \ominus & \text{if } s(j) < s(i) \end{cases}$$

An arc $a \in A$ such that $\text{lab}_s(a) = \oplus$ is called an arc with positive labeling and an arc $a \in A$ such that $\text{lab}_s(a) = \ominus$ is called an arc with negative labeling. Labeling every arc a of A by $\text{lab}_s(a)$, we obtain a labeled digraph (G, lab_s) (Figure 2.2).

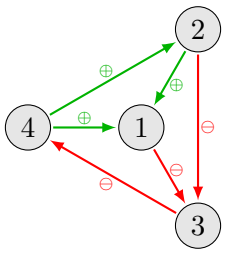


Figure 2.2: A digraph $G = (V, A)$ labeled by the function lab_s, where $\forall i \in V = \{1, \ldots, 4\}, s(i) = i$.

Definition 2.3. A labeled digraph (G, lab) is said to be an *update digraph* (UD) if there exists an update schedule s such that $\text{lab} = \text{lab}_s$, that is $\forall a \in A(G), \text{lab}(a) = \text{lab}_s(a)$ (see the example in Figure 2.3)

A cycle with full negative labeling is a cycle in which all its arcs are labeled negative (Figure 2.3(b)). Notice that an update digraph does not have full negative cycles.

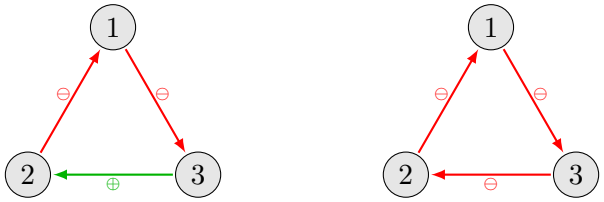


Figure 2.3: (a) A labeled digraph (G, lab) which is an update digraph. (b) A labeled digraph (G, lab') which is not an update digraph.

Finally, we define the following equivalence relation between two update schedules s and s':

$$s \sim_{G(f)} s' \iff G(f, s) = G(f, s'). \tag{2.1}$$

We denote $[s]_{G(f)}$ the equivalence class of s induced by $\sim_{G(f)}$.

2.3 Parallel digraph

A useful tool to study some interesting properties for this thesis is the *potential dependencies digraph of the equivalent parallel network* (in short, *parallel digraph*).

Definition 2.4. Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network and s an update schedule, the *parallel digraph*, denoted as $G_P(f, s) = ([n], A)$, where:

$$\forall v \in B_1, (u, v) \in A \iff (u, v) \in A(f) \tag{2.2}$$

$$\forall v \notin B_1, (u, v) \in A \iff (\exists w \in N_f^-(v), s(w) < s(v) \wedge (u, w) \in A) \vee (s(u) \geq s(v) \wedge (u, v) \in A(f)) \tag{2.3}$$

which is equivalent to:

$$(u, v) \in A \iff \left[(\exists w \in N_f^-(v), s(w) < s(v) \wedge (u, w) \in A) \vee ((u, v) \in A(f) \wedge s(u) \geq s(v))\right] \tag{2.4}$$

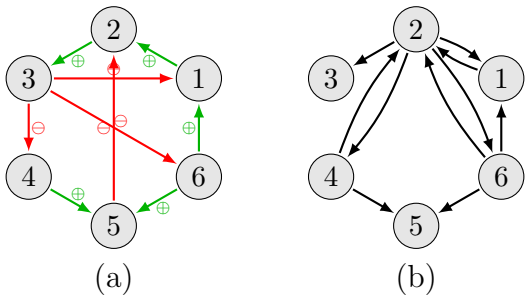


Figure 2.4: (a) A labeled digraph (G, lab) (b) The parallel digraph $G_P(G, \text{lab})$.

Example 2.2. An example of parallel digraph is shown in Figure 2.4

Note that in Equations (2.3) and (2.4) the construction of the arc (u, v) depends on the arc (u, w) which was previously constructed (since w is updated before v). Therefore, the set A is well defined.

By calling it a digraph of potential dependencies we mean that transitively these variables may depend on each other. For example, in Figure 5.1(a), f_3 depends on x_5, and also f_5 depends on x_2, therefore, if x_5 is updated before x_3 (as in the case of update schedule s), it is likely that f_3 depends on x_2. But this is not always the case, since by the nature of the different Boolean functions, some may cancel with others, as is the case of f_3^s in Figure 5.1(b) and Figure 5.1(c). Therefore, it is easy to notice that $G(f^s) \subseteq G_P(f, s)$.

In the case of disjunctive networks, from the research conducted in [34], there is a direct relationship between the parallel digraph and the effective network when updated in parallel.

Remark 2.1. *Given $f, h : \mathbb{B}^n \to \mathbb{B}^n$ two disjunctive Boolean networks and an update schedule s, $h^s = f$ is equivalent to $G_P(h, s) = G(f)$.*

Indeed, if we consider the transitivity of dependencies of the parallel digraph, it produces a composition of functions when defining the effective dependencies of a variable. And considering that the OR function is closed under compositions, no potential dependency is canceled. Therefore, in the case of disjunctive networks, all potential dependencies are effective dependencies and therefore $h^s = f$ is equivalent to $G_P(h, s) = G(f)$.

Chapter 3

Preliminary results

It is widely recognized that several critical characteristics of the interaction digraph exhibit strong correlations with the dynamic behavior manifested within the associated Boolean network. Among these characteristics, the transversal number [7], positive transversal number [7], and packing number [16] of the interaction digraph have been identified as offering upper bounds for the cardinality of fixed points inherent to the corresponding Boolean network. As fixed points of the Boolean network remain unaltered regardless of the update schedule applied, an exploration into the behavior of these three numerical attributes under non-parallel update schedules becomes an interesting topic to study.

In this context, our study delves into the intricate dynamics that emerge when employing update schedules beyond the parallel paradigm. Given the inherent complexity of determining the precise dependencies of a Boolean network when subjected to alternative update schedules, we navigate this challenging terrain by focusing our attention on the parallel digraph. This decision is driven by the expedient fact that the parallel digraph can be deciphered in polynomial time, offering a more tractable foundation from which to unravel the behavior of the aforementioned numerical properties. By investigating how these pivotal metrics evolve within the realm of the parallel digraph, we gain insights into the potential implications of diverging from the standard parallel update scheme in Boolean networks. This endeavor not only expands our comprehension of Boolean network dynamics but also contributes to the broader understanding of network theories and their real-world applications.

3.1 Definition and notation

A vertex set $F \subseteq V$ is a *feedback vertex set* (FVS) of G if $G - F$ is acyclic. F is said to be a *minimal feedback vertex set* of G if F is a FVS of G and there is no other FVS F' such that $F' \subsetneq F$. F is said to be a *minimum feedback vertex set* of G if F is a FVS of G and there is no other FVS set F' such that $|F'| < |F|$.

We denote $\tau(G) = \min\{|F| : F \text{ is a FVS of } G\}$ the *transversal number* of G.

$P \subseteq V$ is a *positive feedback vertex set* (PFVS) of the signed regulatory graph G if $G - P$ is a digraph without positive cycles. The minimum cardinality of a PFVS is denoted by $\tau^+(G)$ and called the *positive transversal number* of G.

$NN \subseteq V$ is a *non-negative feedback vertex set* (NNFVS) of the signed regulatory graph G if $G - NN$ is a digraph with only negative cycles. The minimum cardinality of a NNFVS is denoted by $\tau^{+0}(G)$ and called the *non-negative transversal number* of G.

When there is no confusion, for simplicity we denote $\tau(G)$ by τ, $\tau^+(G)$ by τ^+ and $\tau^{+0}(G)$ by τ^{+0}.

Remark 3.1. *Since a FVS is a particular PFVS,* $\tau^+ \leq \tau$.

Two cycles are said to be vertex-disjoint if they have no common vertex. The maximum number of vertex-disjoint cycles in a graph G is named *packing number* and is denoted by $\nu(G)$. It is known that $\nu(G) \leq \tau(G)$.

Given a labeled digraph (G, lab), we define:

$$V_{\oplus}(G) = \{u \in [n] : \exists v \in [n], (u,v) \in A(G) \wedge \text{lab}(u,v) = \oplus\},$$
$$V_{\ominus}(G) = V(G) \setminus V_{\oplus}(G).$$

3.2 Some relations between interaction graph and parallel digraph

The next lemma shows how the connections between vertices in the interaction digraph are related to the connections in the parallel digraph. This lemma is fundamental for the results in this chapter.

Lemma 3.1. *Let* (G, lab) *be an update digraph. For all* $u, v \in V(G)$*, there exists a walk from* u *to* v *in* G *with a positive label in the first arc if and only if there exists a path from* u *to* v *in* $G_P(G, \text{lab})$.

Proof. Given $W = u_0, u_1, \ldots, u_k$ a walk from u_0 to u_k in G such that $\text{lab}(u_0, u_1) = \oplus$.

We prove by induction that for all $l \in \{1, \ldots, k\}$, there exists P_l a path from u_0 to u_l in $G_P(G, \text{lab})$.

Basis. Since $\text{lab}(u_0, u_1) = \oplus$, $(u_0, u_1) \in A(G_P(G, \text{lab}))$. Therefore, there exists P_1 a path from u_0 to u_1 in $G_P(G, \text{lab})$.

Induction Hypothesis. for all $l \in \{1, \ldots, j\}$, there exists P_l a path from u_0 to u_l in $G_P(G, \text{lab})$.

Case $j+1$**.** There are two cases:

1. $\text{lab}(u_j, u_{j+1}) = \oplus$

   Since, $(u_j, u_{j+1}) \in A(G_P(G, \text{lab}))$ and there exists P_j a path from u_0 to u_j in $G_P(G, \text{lab})$. (by induction hypothesis). Then, there exists a walk from u_0 to u_{j+1} in $G_P(G, \text{lab})$, therefore, there exists P_{j+1} a path from u_0 to u_{j+1} in $G_P(G, \text{lab})$.

2. $\text{lab}(u_j, u_{j+1}) = \ominus$

   Let $j^* = \max\{i \in \{0, \ldots, j-1\} : \text{lab}(u_i, u_{i+1}) = \oplus\}$. We observe that eventually $j^* = 0$.

   Since $(u_{j^*}, u_{j+1}) \in A(G_P(G, \text{lab}))$ and there exists P_{j^*} a path from u_0 to u_{j^*} in $G_P(G, \text{lab})$ (by induction hypothesis). Then, there exists a walk from u_0 to u_{j+1} in $G_P(G, \text{lab})$. Therefore, there exists a path from u_0 to u_{j+1} in $G_P(G, \text{lab})$.

In this way for all $l \in \{1, \ldots, k\}$, there exists P_l a path from u_0 to u_l in $G_P(G, \text{lab})$.

Conversely, given $P = u_0, u_1, \ldots, u_k$ a path in $G_P(G, \text{lab})$. Then, by definition of $G_P(G, \text{lab})$, $\forall i \in \{0, \ldots, k-1\}, (u_i, u_{i+1}) \in A(P)$, there exists a walk from u_i to u_{i+1} such that the first arc

has positive labeling. Finally, since for every $i \in \{0, \ldots, k-1\}$, there exists a walk from u_i to u_{i+1} in G such that the first arc has positive labeling. Then, the concatenation of these walks form a walk from u_0 to u_k whose first arc has positive labeling. Therefore, there exists a walk from u_0 to u_k in G such that the first arc has positive labeling. □

Example 3.1. Let (G, lab) be an update digraph as in Figure 2.4(a) and its parallel digraph in Figure 2.4(b). Since there is a walk W from 2 to 2 in (G, lab) such that W starts with an arc with positive labeling (2,3,4,5,2; 2,3,1,2 or 2,3,6,1,2), then there exists a path P from 2 to 2 in $G_P(G, \text{lab})$ (2,4,2; 2,1,2 or 2,6,1,2). Let (G, lab) be a labeled digraph as in Figure 3.1(a) and its parallel digraph in Figure 3.1(b). Since there is a walk W from 2 to 1 in (G, lab) such that W starts with an arc with positive labeling (2,3,4,5,6,1), then there exists a path P from 2 to 1 in $G_P(G, \text{lab})$ (2,4,5,1).

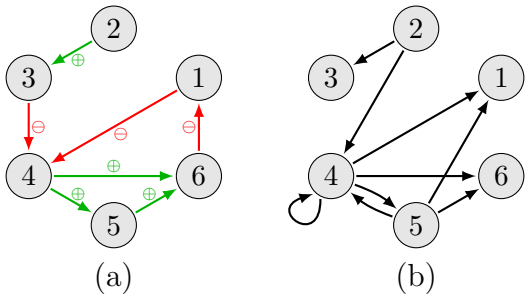


Figure 3.1: (a) A labeled digraph (G, lab) (b) The parallel digraph $G_P(G, \text{lab})$.

In the proposition below, we prove that the structure of non-trivial strongly connected components of a digraph is preserved applying an update schedule.

Proposition 3.2. *Let (G, lab) be an update digraph with $G_1, \ldots, G_m$ its non-trivial strongly connected components. Then, $G_P(G, \text{lab})$ has $\tilde{G}_1, \ldots, \tilde{G}_m$ its non-trivial strongly connected components such that:*

$$\forall i \in \{1, \ldots, m\}, V(\tilde{G}_i) = V_{\oplus}(G_i)$$

Proof. Let G_i be any non-trivial strongly connected component of G.

Note that G_i has at least one arc with positive labeling, otherwise there would be cycles with fully negative labeling, and that is not an update digraph.

Given $u \in V_{\oplus}(G_i)$, for all $v \in V_{\oplus}(G_i)$, there exists a walk W from u to v in G_i such that the first arc of W has positive labeling. (Particularly, there exists a walk from u to u in G_i) Then, by Lemma 3.1, there exists a path from u to v and from v to u in $G_P(G, \text{lab})$. Therefore, $\exists j \in \{1, \ldots, m\}, V_{\oplus}(G_i) \subseteq V(\tilde{G}_j)$.

Conversely, let $\tilde{u}, \tilde{v} \in V(\tilde{G}_j)$ be any two vertices. Since $\tilde{G}_j$ is a strongly connected component of $G_P(G, \text{lab})$, there exists a path from $\tilde{u}$ to $\tilde{v}$ in $\tilde{G}_j$ and, by Lemma 3.1, there exists a walk from $\tilde{u}$ to $\tilde{v}$ in G such that the first arc has positive labeling.

Since there exists a walk from $\tilde{u}$ to $\tilde{v}$ in G with the first arc with positive labeling and there exists a walk from $\tilde{v}$ to $\tilde{u}$ in G with the first arc with positive labeling, $\{\tilde{u}, \tilde{v}\} \subseteq V(G_i)$ for some $i \in \{1, \ldots, m\}$ and $\{\tilde{u}, \tilde{v}\} \in V_{\oplus}(G_i)$. Therefore, $V(\tilde{G}_j) \subseteq V_{\oplus}(G_i)$. □

Example 3.2. Let (G, lab) be a labeled digraph as in Figure 3.2(a) and its parallel digraph in Figure 3.2(b). Then, (G, lab) has three non-trivial strongly connected components: G_1 with vertices 1 and 2, G_2 with vertex 3 and G_3 with vertices 5, 6 and 7. At the same time, $G_P(G, \text{lab})$ has three non-trivial strongly connected components: $\tilde{G}_1$ with vertex 1, $\tilde{G}_2$ with vertex 3 and $\tilde{G}_3$ with vertex 5.

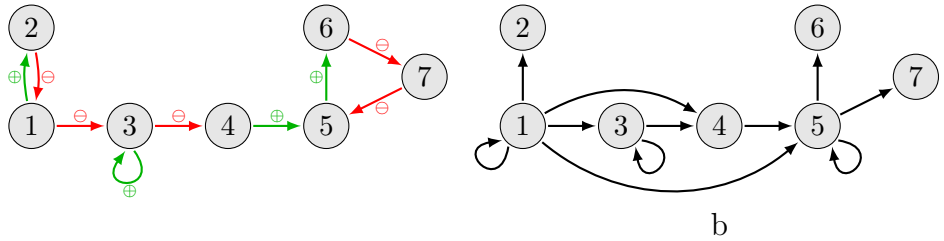


Figure 3.2: (a) A labeled digraph (G, lab) (b) The parallel digraph $G_P(G, \text{lab})$.

The following proposition shows that the pendant nodes of the parallel digraph (Definition 2.1) can be easily identified in the labeled digraph that generates the parallel digraph.

Proposition 3.3. *Let (G, lab) be an update digraph and G_i a strongly connected component of G. Given $u \in V(G_i)$, if $u \in V_{\ominus}(G_i, \text{lab})$, then u is a pendant node of $G_P(G_i, \text{lab})$.*

Proof. We prove by contradiction. Let us suppose that $d^+_{G_P(G_i,\text{lab})}(u) \neq 0$, i.e. $\exists (u, v) \in A(G_P(G_i, \text{lab}))$ then, by definition of $G_P((G, \text{lab}))$, there exists $w \in V(G_i)$, such that $\text{lab}(u, w) = \oplus$, which is a contradiction. □

Combining the results of Proposition 3.2 and Proposition 3.3, the following corollaries are obtained.

Corollary 3.4. *Let (G, lab) be an update digraph. $G_P(G, \text{lab})$ is a strongly connected digraph if and only if G has a unique strongly connected component and $V_{\oplus}(G, \text{lab}) = V(G)$.*

In the following proposition, we present a relationship between the transversal number of G and the size of the non-trivial connected source component of its parallel graph (independent of the update schedule with which it is generated).

Proposition 3.5. *Let (G, lab) be an update digraph strongly connected. Then, the size of the non-trivial strongly connected component of $G_P(G, \text{lab})$ is greater or equal than $\tau(G)$.*

Proof. By contradiction, let us suppose that there exists a labeling function lab such that $G_P(G, \text{lab})$ has a strongly connected component of size lesser than $\tau(G)$.

Let $S \subseteq V(G)$ be the vertex set in the strongly connected component of $G_P(G, \text{lab})$, such that $|S| < \tau(G)$.

Then, there exists a cycle $C = v_0, \ldots, v_l$ with $v_0 = v_l$ in G, such that $\forall i \in \{0, \ldots, l\}$, $v_i \notin S$.

Then, by Corollary 3.4, $\forall i \in \{0, \ldots, l-1\}$, $\text{lab}(v_i, v_{i+1}) = \ominus$, resulting in a cycle in G with all its arcs negative, which is a contradiction. □

Now, with these results, the following objective is proposed: to define an update schedule that controls the growth of the transversal number of G when calculating its parallel digraph. For this purpose, we define the following candidate update schedule.

Definition 3.1. Let $F \subseteq V(G)$ be a feedback vertex set of G. We define the labeling function $\text{lab}_F : A(G) \to \{\oplus, \ominus\}$ as:

$$\forall (u,v) \in A(G), \quad \text{lab}_F(u,v) = \begin{cases} \oplus & \text{if } u \in F \\ \ominus & \text{otherwise.} \end{cases}$$

First, we show that the labeled digraph is a valid update schedule.

Proposition 3.6. *Let G be a strongly connected digraph and $F \subseteq V(G)$ a feedback vertex set of G, then (G, lab_F) is an update digraph.*

Proof. Let F be an FVS it is easy to notice that the set $A^{\oplus}$ defined as follows:

$$A^{\oplus} = \{(u,v) \in A(G) : u \in F\}$$

is a feedback arc set. Therefore, $G - A^{\oplus}$ is an acyclic directed graph. The topological order of this DAG, shows us different "layers" that we can consider as the blocks of our candidate update schedules. If we finally place the set F as the last block, the resulting update schedule respects each and every labeling of lab_F, so since there is an update schedule, (G, lab_F) is an update digraph. □

For example, if for the digraph presented in Figure 3.3(a), we choose the $\{1, 6\}$ set as FVS, the following sets of positive arcs (Figure 3.3(b)) and negative arcs (Figure 3.3(c)) are generated. If we follow the topological order of the negative arcs, the update schedule $s = \{3, 4\}\,\{5\}\,\{2\}\,\{1, 6\}$ is generated, which respects both positive and negative labels, therefore the label generated by lab_F is update.

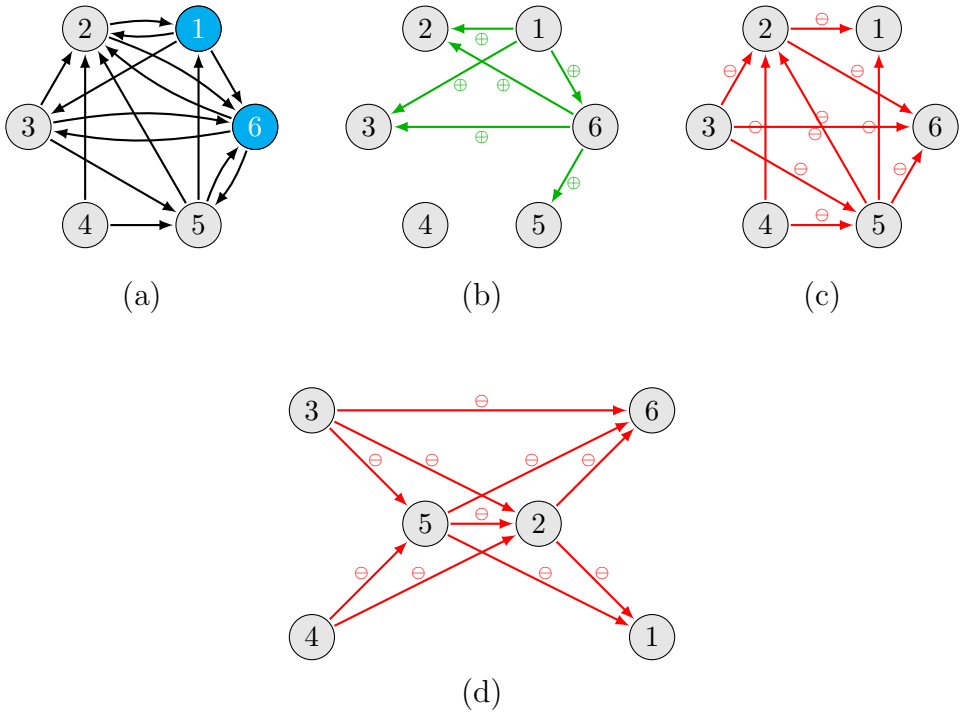


Figure 3.3: (a) $\{1, 6\}$ is a FVS of G (b) $A^{\oplus}$ (c) $G - A^{\oplus}$ (d) Topological order of $G - A^{\oplus}$

Finally, it is shown that the size of the nontrivial strongly connected component resulting from the parallel digraph is equal to $\tau(G)$ and no larger.

Proposition 3.7. *Let G be a strongly connected digraph. Then, there exists a labeling function* lab *such that (G, lab) is an update digraph and $G_P(G, \text{lab})$ has one and only one non-trivial strongly connected component of size $\tau(G)$ and the vertices that are not in the strongly connected component are "pendant nodes".*

Proof. Let $F \subseteq V(G)$ be a feedback vertex set of G such that $|F| = \tau(G)$. Then, by Corollary 3.4, (G, lab_F) has one strongly component of size $\tau(G)$.

By contradiction, let us suppose that exists a vertex u that is not in the strongly connected component of G and is not a "pendant node". Then, by definition of "pendant node", there exists a vertex v such that $(v, u) \in A(G)$ and v is not in the strongly connected component of G. Therefore, the vertex v is not in the strongly connected component and $d^+_{G_P(G,\text{lab})}(v) > 0$, which contradicts Corollary 3.4. □

In addition, we want to study the packing number of the parallel digraph. For this, it is necessary to study how the parallel digraph cycles are formed.

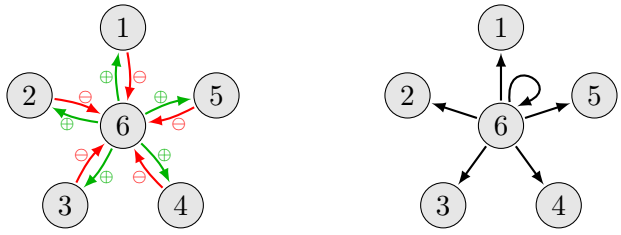
Lemma 3.8. *Let (G, lab) be an update digraph. For every cycle $C_i \in G$, there exists a cycle $\tilde{C}_i \in G_P(G, \text{lab})$ such that $V(\tilde{C}_i) \subseteq V(C_i)$*

Proof. Given $C_i : u_0, u_1, \ldots, u_k = u_0$ a cycle in G. For all $u \in V_\oplus(C_i)$, by Lemma 3.1, there exists a path from u to u in $G_P(G, \text{lab})$. Then, $V_\oplus(C_i) \subseteq V(\tilde{C}_i)$.

Conversely, given $\tilde{C}_i : v_0, v_1, \ldots, v_k = v_0$ a cycle in $G_P(G, \text{lab})$. For all $v \in V(\tilde{C}_i)$, by Lemma 3.1, there exists a walk from v to v in G such that the first arc has a positive labeling. Then $V(\tilde{C}_i) \subseteq V_\oplus(C_i)$. □

Lemma 3.8 shows that each cycle of G has a representative cycle in the parallel digraph, but Remark 3.2 shows that this representative cycle is not exclusive to a single cycle of G, so the number of cycles in the parallel digraph may decrease.

Remark 3.2. *Figure 3.4 shows that given (G, lab), the number of cycles in G is not necessarily equal to the number of cycles in $G_P(G, \text{lab})$.*



(a) (b)

Figure 3.4: (a) A labeled digraph (G, lab) with 5 cycles (b) The parallel digraph $G_P(G, \text{lab})$ with 1 cycle.

With these results, as in the previous case, we can define a relationship between the packing number of G and the packing number of any parallel digraph of G.

Proposition 3.9. *Let (G, lab) be an update digraph strongly connected. Then, the packing number of $G_P(G, \text{lab})$ is greater or equal than the packing number of G.*

Proof. By contradiction, let us suppose that there exists a labeling function lab such that $\nu(G_P(G, \text{lab}))$ is lesser than $\nu(G)$.

Then, there exists at least one cycle in G that does not have an associated cycle in $G_P(G, \text{lab})$, which contradicts Lemma 3.8. □

Since lab_F was designed to preserve the transversal number of G, it is interesting to see if the same is true for the packing number of G.

Remark 3.3. *Figure 3.5 shows that given G and a labeling function lab_F, not necessarily $\nu(G) = \nu(G_P(G, \text{lab}_F))$.*

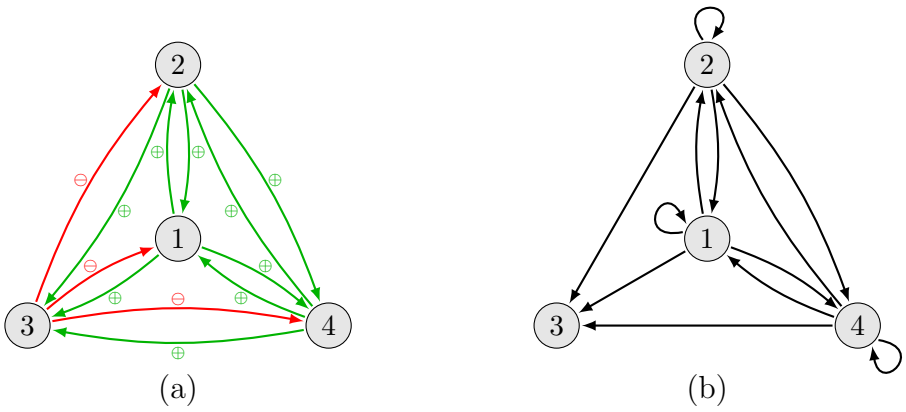


Figure 3.5: (a) A labeled digraph (G, lab_F) with $\tau(G) = 3$ and $\nu(G) = 2$ (b) The parallel digraph $G_P(G, \text{lab}_F)$ with $\tau(G_P(G, \text{lab}_F)) = 3$ and $\nu(G_P(G, \text{lab}_F)) = 3$.

It is possible to prove that, if there exists a digraph G with $\nu(G) = \tau(G)$ (a very particular case), in that case, lab_F generates a parallel digraph such that the packing number of G is conserved.

Proposition 3.10. *Let G be a strongly connected digraph such that $\nu(G) = \tau(G)$. Then, there exists a labeling function* lab *such that (G, lab) is an update digraph and $\nu(G_P(G, \text{lab})) = \nu(G)$.*

Proof. Let $F \subseteq V(G)$ be a feedback vertex set of G such that $|F| = \tau(G)$. Let SC be the set of disjoint cycles of G with $|SC| = \nu(G)$. As $\nu(G) = \tau(G)$, then each cycle of SC has one and only one vertex in F. Then, $\forall u \in F$, there is a walk from u to u with the first arc with positive labeling in (G, lab_F), therefore, there is a loop over u in $G_P(G, \text{lab}_F)$. Each of these loops are disjoint sets, so $\nu(G_P(G, \text{lab}_F)) = \nu(G)$. □

In addition, for the particular case of the graph K_n (complete digraph of n vertices) we define the following update schedule.

Definition 3.2. Let $F \subseteq V(G)$ a feedback vertex set of G. We define a labeling function $\text{lab}_{\overline{F}} : A(G) \to \{\oplus, \ominus\}$ as:

$$\forall (u, v) \in A(G) \quad \text{lab}_{\overline{F}}(u, v) = \begin{cases} \oplus & \text{if } v \in F \\ \ominus & \text{otherwise.} \end{cases}$$

First, we show that $\text{lab}_{\overline{F}}(u, v)$ induces a valid update schedule.

Proposition 3.11. *Let G be a strongly connected digraph and $F \subseteq V(G)$ a feedback vertex set of G, then $(G, \text{lab}_{\overline{F}})$ is an update digraph.*

Proof. Let F be an FVS it is easy to notice that the set $A^{\oplus}$ defined as follows:

$$A^{\oplus} = \{(u, v) \in A(G) : v \in F\}$$

is a feedback arc set. Therefore, $G - A^{\oplus}$ is an acyclic directed graph. The topological order of this DAG, shows us different "layers" that we can consider as the blocks of our candidate update schedules. If we place the set F as the first block, the resulting update schedule respects each and every labeling of $\text{lab}_{\overline{F}}$, so since there is an update schedule, $(G, \text{lab}_{\overline{F}})$ is an update digraph. □

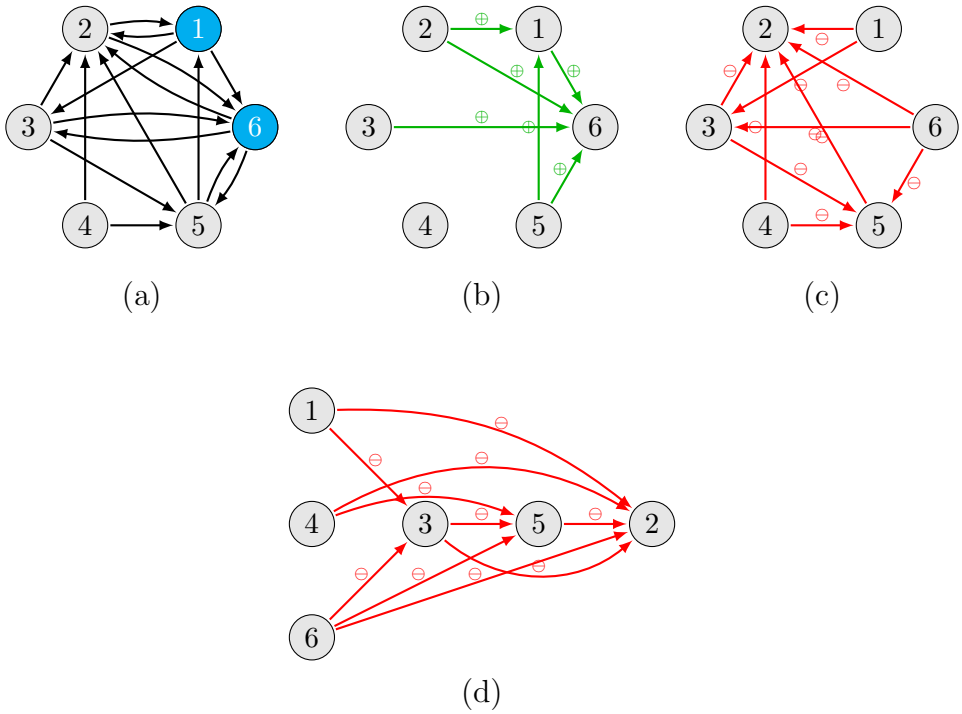


Figure 3.6: (a) $\{1, 6\}$ is a FVS of G (b) $A^{\oplus}$ (c) $G - A^{\oplus}$ (d) Topological order of $G - A^{\oplus}$

For example, if for the digraph presented in Figure 3.6(a), we choose the $\{1, 6\}$ set as FVS, the following sets of positive arcs (Figure 3.6(b)) and negative arcs (Figure 3.6(c)) are generated. If we follow the topological order of the negative arcs, the update schedule $s = \{1, 4, 6\}\,\{3\}\,\{5\}\,\{2\}$ is generated, which respects both positive and negative labels, therefore the label generated by lab_F is update.

Proposition 3.12. *Given $n \in \mathbb{N}$ even, there exists a labeling function* lab *such that (K_n, lab) is an update digraph and $\nu(G_P(K_n, \text{lab})) = \nu(K_n) = \frac{n}{2}$.*

Proof. Let $F \subseteq V(K_n)$ be a feedback vertex set of K_n such that $|F| = n - 1$. Then, let u be the node that is not in F (whose input arcs have negative labeling in $(K_n, \text{lab}_{\overline{F}})$) has the following behavior in $G_P(K_n, \text{lab}_{\overline{F}})$: there is a loop in u, since there is a walk from u to u with the first arc with positive labeling (going through any vertex in F), for every arc with negative labeling towards u, that arc is in $G_P(K_n, \text{lab}_{\overline{F}})$, since there is a walk with the first arc with positive labeling from any vertex in F to u (passing through any other vertex in F). Finally, $G_P(K_n, \text{lab}_{\overline{F}})$ is K_n plus a loop in u, then the vertices in F form a subgraph with form K_{n-1} which has $\frac{n-2}{2}$ disjoint cycles. also considering the loop in u, $G_P(K_n, \text{lab}_{\overline{F}})$ has $\frac{n}{2}$ disjoint cycles. □

Proposition 3.13. *Given $n \in \mathbb{N}$ odd. Then, there is no labeling function* lab *such that* (K_n, lab) *is an update digraph and* $\nu(G_P(K_n, \text{lab})) = \nu(K_n) = \frac{n-1}{2}$.

Proof. If we divide $V(K_n)$ in two sets A and B, such that $\forall u \in A, s(u) = 1$ and $\forall v \in B, s(v) = 2$ (with $|B| \geq 1$), $G_P(K_n, \text{lab})$ would have the following characteristics: every vertex in B has a loop and the vertices in A form a subgraph of form $K_{|A|}$ which has:

$$\nu(K_{|A|}) = \begin{cases} \frac{|A|}{2} \text{ disjoint cycles} & \text{if } |A| \text{ is even} \\ \frac{|A|-1}{2} \text{ disjoint cycles} & \text{otherwise} \end{cases}$$

So $G_P(K_n, \text{lab})$ has in total:

$$\nu(G_P(K_n, \text{lab})) = \begin{cases} \frac{|A|}{2} + (n - |A|) \text{ disjoint cycles} & \text{if } |A| \text{ is even} \\ \frac{|A|-1}{2} + (n - |A|) \text{ disjoint cycles} & \text{otherwise} \end{cases}$$

Finally, so that $\nu(G_P(K_n, \text{lab})) = \frac{n-1}{2}$ then:

$$|A| = \begin{cases} n+1 & \text{if } |A| \text{ is even} \\ n & \text{otherwise} \end{cases}$$

Which leads to a contradiction to the fact that $|B| = n - |A|$ must be greater than 1. □

3.3 FVS invariant

From Proposition 3.5 to 3.7, we study the relation of the transversal number (size of the smallest feedback vertex set) of G and the transversal number of its parallel digraph. Moreover, we define a special update schedule that allows us to have the same transversal number in both G and its parallel digraph. But, how are these feedback vertex sets conformed?, do they have the same vertices? We answer these questions below.

Proposition 3.14. *Let* (G, lab) *be an update digraph. Given* $F \subseteq V(G)$*, if* $F \in \text{FVS}(G_P(G, \text{lab}))$*, then* $F \in \text{FVS}(G, \text{lab})$*. Thus,* $\tau(G_P(G, \text{lab})) \geq \tau(G\,\text{lab})$.

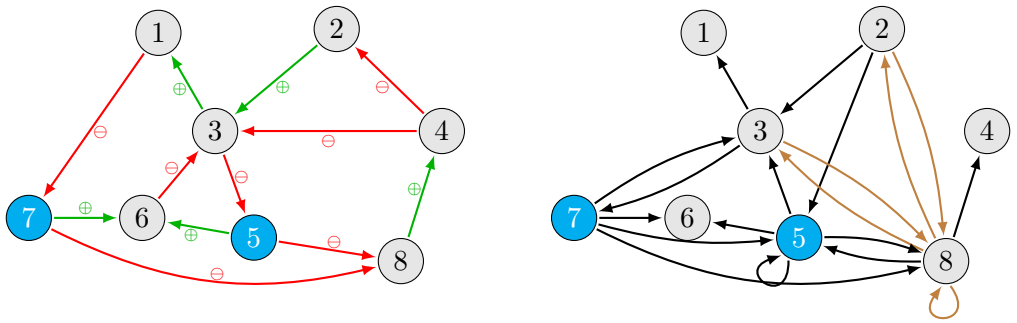
Proof. By contradiction, let us suppose that there exists $F \subseteq V(G)$ such that $F \in \text{FVS}(G_P(G, \text{lab}))$ and $F \notin \text{FVS}(G, \text{lab})$.

Then, there exists a cycle $C = v_0, \ldots, v_l$ with $v_0 = v_l$ in (G, lab), such that $\forall i \in \{0, \ldots, l\}$, $v_i \notin F$:

- If there is only one arc with positive labeling (v_i, v_{i+1}), then there exists a walk from v_i to v_i with the first arc with positive labeling. Then, there exists a loop in v_i in $G_P(G, \text{lab})$, in this way $F \notin \text{FVS}(G_P(G, \text{lab}))$, which is a contradiction.
- If there is more than one arc with positive labeling, the vertices in $V_{\oplus}(C)$ form a walk $W = w_0, \ldots, w_m$ with $w_m = w_0$ and $m \leq l$ such that $\forall j \in \{0, \ldots, m-1\}$, there exists a walk from w_j to w_{j+1} with the first arc with positive labeling. Finally, the vertices in W form a cycle $C' \in G_P(G, \text{lab})$, in this way $F \notin \text{FVS}(G_P(G, \text{lab}))$, which is a contradiction.

□

Remark 3.4. *Figure 3.7 shows that given $F \subseteq V(G)$, if $F \in \mathrm{FVS}(G, \mathrm{lab})$, not necessarily $F \in \mathrm{FVS}(G_P(G, \mathrm{lab}))$.*



(a) (b)

Figure 3.7: (a) $F = \{5, 7\}$ is a FVS for (G, lab) (b) Brown cycles are not covered by F in $G_P(G, \mathrm{lab})$.

3.4 Interaction Graph with weighted arcs

So far we have studied the behavior of the transversal number and the packing number considering arcs without weights, i.e., it is not taken into account what type of Boolean function generates such arc in the interaction graph. But, what changes would occur if we consider such weights? Would new definitions be necessary to study such cases?

Definition 3.3. A Boolean function $f : \mathbb{B}^m \to \mathbb{B}$ is *increasing monotone* on input u if for all $x \in \mathbb{B}^m$, $x_u = 0$ and $f(x) \leq f(x + e_u)$, and *decreasing monotone* on input u if for all $x \in \mathbb{B}^m$, $x_u = 0$ and $f(x) \geq f(x + e_u)$, where $e_u \in \mathbb{B}^m$ denotes the binary vector with all components equal to 0, except for component u, which equals 1.

Definition 3.4. A *signed interaction graph* is obtained by associating a weight function ω to the arcs of an interaction graph G. The weight function ω is defined for every arc $(u, v) \in G$ as follows:

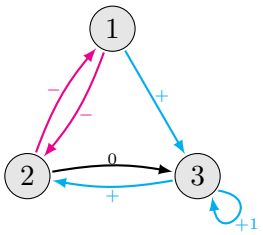
$$\omega(u, v) = \begin{cases} +1 & \text{if } f_v \text{ is increasing monotone on input } u \\ -1 & \text{if } f_v \text{ is decreasing monotone on input } u \\ 0 & \text{otherwise} \end{cases}$$

Definition 3.5. Similarly, the weight function ω can be extended to walks (cycles, circuits or paths) as follows, the weight of a walk is the product of the weight of its arcs.

Example 3.3. Figure 3.8 shows an example of *weight* of cycles

Definition 3.6. A *signed parallel digraph* is obtained by associating a weight function ω_p to the arcs of a parallel digraph $G_P(G, \mathrm{lab})$. The weight function ω_p is defined for all arc $(u, v) \in G_P(G, \mathrm{lab})$ as follows:

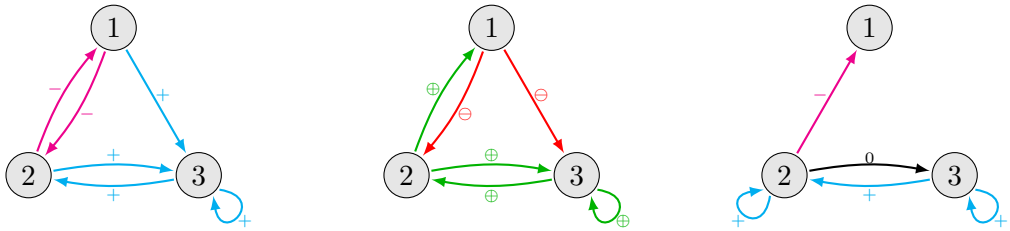
$$\omega_p(u, v) = \begin{cases} +1 & \text{if } \forall W(u, v) \in G,\ \omega(W(u, v)) = +1 \\ -1 & \text{if } \forall W(u, v) \in G,\ \omega(W(u, v)) = -1 \\ 0 & \text{otherwise} \end{cases}$$



$$f_1 = \neg x_2$$
$$f_2 = \neg x_1 \wedge x_3$$
$$f_3 = (x_1 \wedge x_3 \wedge x_2) \vee \neg x_2$$
$$C_1 = 1,3,2,1 \rightarrow \omega(C_1) = (+1) \cdot (+1) \cdot (-1) = -1$$
$$C_2 = 1,2,1 \rightarrow \omega(C_2) = (-1) \cdot (-1) = +1$$
$$C_3 = 2,3,2 \rightarrow \omega(C_3) = (0) \cdot (+1) = 0$$

Figure 3.8: Example of weight of cycles.

Example 3.4. An example of *signed parallel digraph* is shown in Figure 3.9. Note that the arc from 2 to 3 in the signed parallel digraph has weight 0 because the walk $W_1 = 2,3$ has weight $\omega(W_1) = +1$ and the walk $W_2 = 2,1,3$ has weight $\omega(W_2) = -1$.



(a) (b) (c)

Figure 3.9: (a) A signed interaction graph G (b) A labeled digraph (G, lab) (c) The signed parallel digraph $G_P(G, \text{lab})$.

With these new definitions, it is possible to adapt Lemma 3.8, in the following proposition.

Proposition 3.15. *Let $G_P(G, \text{lab})$ a signed parallel digraph. Then, for every cycle $C_i \in G_P(G, \text{lab})$ with positive weight, there exists a circuit in G with positive weight associated with C_i, and for every cycle $C_j \in G_P(G, \text{lab})$ with negative weight, there exists circuit in G with negative weight associated with C_j.*

Proof. By Definition 3.5, a cycle with positive weight in $G_P(G, \text{lab})$ is formed by an even number of arcs with negative weight in $G_P(G, \text{lab})$, each arc with negative weight (by Lemma 3.1 and Definition 3.6), arises from a walk with an odd number of arcs with negative weight in G. Then, the concatenation of these walks forms a circuit which has an even number of arcs with negative weight, so the circuit has positive weight.

Similarly, a cycle with negative weight in $G_P(G, \text{lab})$ is formed by an odd number of arcs with negative weight in $G_P(G, \text{lab})$, each arc with negative weight (by Lemma 3.1 and Definition 3.6), arises from a walk with an odd number of arcs with negative weight in G. Then, the concatenation of these walks forms a circuit which has an odd number of arcs with negative weight, so the circuit has negative weight. □

Example 3.5. Let $C_1 = 2,3,2$ be a cycle in $G_P(G, \text{lab})$ (Figure 3.10(c)), which is associated with the walk $W_1 = 2,1,2,3,4,1,2$ in G (Figure 3.10(a)):

$$\omega_p(C_1) = \omega_p(2,3) \cdot \omega_p(3,2) = (-1) \cdot (-1) = +1$$

$$\omega(W_1) = \omega(2,1) \cdot \omega(1,2) \cdot \omega(2,3) \cdot \omega(3,4) \cdot \omega(4,1) \cdot \omega(1,2) =$$
$$(+1) \cdot (-1) \cdot (+1) \cdot (-1) \cdot (-1) \cdot (-1) = +1$$

Let $C_2 = 2, 2$ be a cycle in $G_P(G, \text{lab})$, which is associated with the walk $W_2 = 2, 1, 2$ in G:

$$\omega_p(C_2) = \omega_p(2,2) = (-1) = -1$$

$$\omega(W_2) = \omega(2,1) \cdot \omega(1,2) = (+1) \cdot (-1) = -1$$

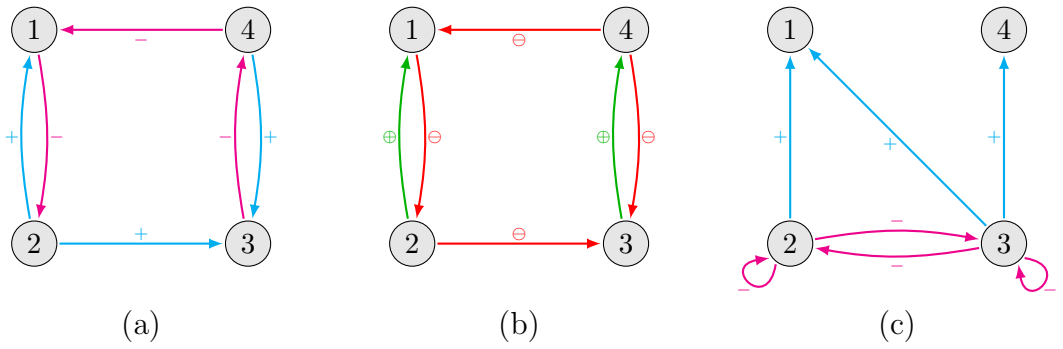


Figure 3.10: (a) A signed interaction graph G (b) A labeled digraph (G, lab) (c) The signed parallel digraph $G_P(G, \text{lab})$.

Remark 3.5. *The Figure 3.10 shows that a signed interaction graph without cycles with positive weight (a), can generate a signed parallel graph with cycles with positive weight (c). Figure 3.10 also shows that a* PFVS *in* $G_P(G, \text{lab})$ *is not necessarily a* PFVS *in* G*, since* $\{2\}$ *is a* PFVS *for* $G_P(G, \text{lab})$ *and it is not a* PFVS *for* G*, the same happens with* $\{3\}$*.*

Remark 3.6. *Let* $u \in V(G)$ *be a vertex that cover a cycle with positive weight in* G*, but* $u \in V(G_P(G, \text{lab}))$ *does not necessarily cover a cycle with positive weight in* $G_P(G, \text{lab})$*. Figure 3.11 shows an example about this.*

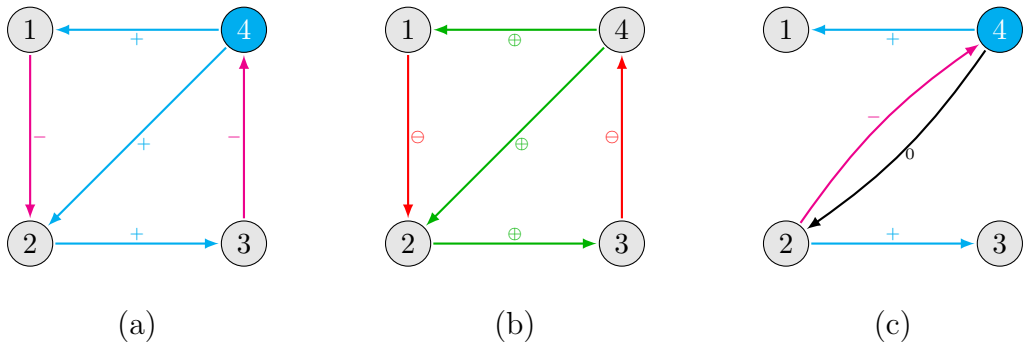
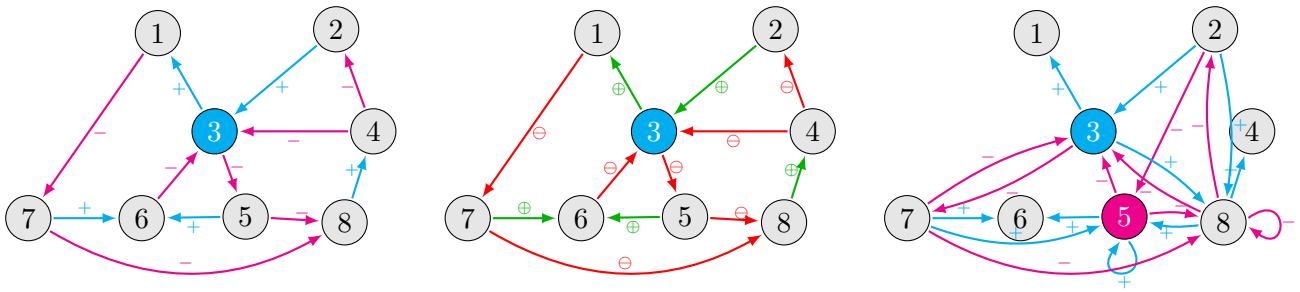


Figure 3.11: (a) $\{4\}$ cover the cycle $\{4, 1, 2, 3, 4\}$ with positive weight (c) $\{4\}$ do not cover any cycle with positive weight.

Remark 3.7. *Figure 3.12 shows that given* $F \subseteq V(G)$*, if* $F \in \text{PFVS}(G)$*,* F *is not necessarily a* PFVS *in* $G_P(G, \text{lab})$*.*

By refining the above proposition with the following remarks it is possible to obtain the following lemma



(a) (b) (c)

Figure 3.12: (a) $F = \{3\}$ is a PFVS for G (c) Loop with positive weight in 5 is not covered by F in $G_P(G, \text{lab})$.

Lemma 3.16. *Let (G, lab) be a regulatory update digraph. For every cycle $C_i \in G$ with positive weight, there exists a cycle $\tilde{C}_i \in G_P(G, \text{lab})$ such that $\omega_p(\tilde{C}_i) \in \{0, +1\}$ and $V(\tilde{C}_i) = V_\oplus(C_i)$ and for every cycle $C_j \in G$ with negative weight, there exists a cycle $\tilde{C}_j \in G_P(G, \text{lab})$ such that $\omega_p(\tilde{C}_j) \in \{-1, 0\}$ and $V(\tilde{C}_j) = V_\oplus(C_j)$.*

Proof. By Lemma 3.8, we know that for every cycle $C_i \in G$, exists a cycle $\tilde{C}_i \in G_P(G, \text{lab})$ such that $V(\tilde{C}_i) = V_\oplus(C_i)$. We must show that if $\omega(C_i) = +1$ then $\omega_p(\tilde{C}_i) \in \{0, +1\}$ and if $\omega(C_j) = -1$ then $\omega_p(\tilde{C}_j) \in \{-1, 0\}$.

Whether the weight of C_i is in $\{-1, +1\}$, the arcs that form C_i have weight in $\{-1, +1\}$, so the arcs of $\tilde{C}_i$ also have weight in $\{-1, +1\}$. If some arc of $\tilde{C}_i$ with positive (or negative) weight arises from another walk (with arcs out of C_i) with negative (or positive) weight, that arc has weight 0 in $G_P(G, \text{lab})$. If some arc of $\tilde{C}_i$ has weight 0, then $\tilde{C}_i$ has weight 0.

If no arc of $\tilde{C}_i$ has weight 0 and C_i has positive weight, then the number of arcs with negative weight in $\tilde{C}_i$ is even (otherwise, the number of arcs with negative weight in C_i would be odd), so the weight of $\tilde{C}_i$ is $+1$.

If no arc of $\tilde{C}_i$ has weight 0 and C_i has negative weight, then the number of arcs with negative weight in $\tilde{C}_i$ is odd (otherwise, the number of arcs with negative weight in C_i would be even), so the weight of $\tilde{C}_i$ is -1. □

Remark 3.8. *Let $F \subseteq V(G)$ be a* PFVS *of G, F is not necessarily a* NNFVS *of $G_P(G, \text{lab})$. Figure 3.13 shows an example about this.*

Remark 3.9. *The Figure 3.13 also shows that if $F \in \text{PFVS}(G)$ and $H \in \text{NNFVS}(G_P(G, \text{lab}))$, F is not necessarily a subset of H. In this case, $F = \{4\}$ and $H = \{2\}$.*

Finally, with this lemma, it is possible to conclude the following proposition.

Proposition 3.17. *Let (G, lab) be a regulatory update digraph. Then, $\tau^+(G) \leq \tau^{+0}(G_P(G, \text{lab}))$.*

Proof. As for every cycle with positive weight in G, its associated cycle in $G_P(G, \text{lab})$ has weight in $\{0, +1\}$, and since it is also possible that new cycles with positive weight appear in $G_P(G, \text{lab})$ (see Figure 3.10), then $\tau^+(G) \leq \tau^{+0}(G_P(G, \text{lab}))$. □

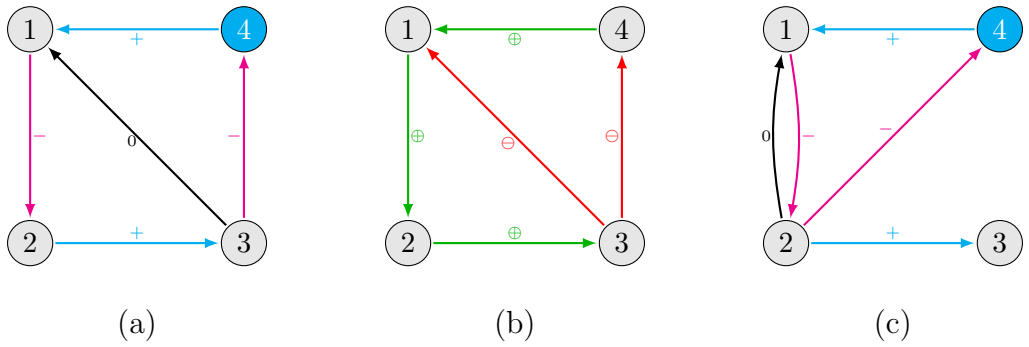


Figure 3.13: (a) $\{4\}$ cover the cycle $\{4, 1, 2, 3, 4\}$ with positive weight (c) $\{4\}$ do not cover the cycle $\{1, 2, 1\}$ with weight 0

3.5 Conclusions

The findings presented in this chapter not only provide us with a crucial threshold regarding the inherent structure of parallel digraphs, as elucidated through their transversal number and packing number, but they also furnish us with invaluable insights. Beyond merely delineating the characteristics of these digraphs, the results empower us to discern and comprehend the intricate structure of an update schedule intricately linked to its corresponding labeled digraph. This newfound understanding proves to be instrumental in paving the way for the subsequent chapters, where the implications and applications of this structural knowledge are further explored and harnessed for more comprehensive insights and practical implications. Thus, the revelations in this chapter not only contribute to our theoretical understanding of parallel digraphs but also offer a tangible and practical framework for addressing real-world problems in subsequent discussions.

Chapter 4

Fixed Point Algorithm

The results and algorithms in this chapter correspond to the research published in [9]. As Bioinformatics is a biological journal, some concepts are referred to by other names (e.g., the interaction graph is referred to as the regulatory graph).

4.1 Introduction

The fixed points of a network are, for example, associated to distinct types of cells defined by patterns of gene activity [14, 42, 45]. In this chapter, we focus on finding the steady states of a Boolean network using the fact that these steady states are invariant with respect to the update mode.

The problem of finding the steady states or fixed points of a Boolean network is a difficult task. In [6] it is shown that deciding if a network has a fixed point is NP-complete, and in [30] it is proved that counting how many fixed points a Boolean network has is #P-complete. For this reason, in the literature, several strategies have been proposed to find fixed points in Boolean networks. Some of them are:

- **Reduction methods**: This strategy, used in [80, 82, 84], focuses on reducing the number of variables of the network, and finding the fixed points in the reduced network.
- **Representation as polynomial functions**: This technique used in [41, 86] consists in representing each Boolean function of the network as a polynomial function in the variables $x_1, \ldots, x_n$, such that the problem of finding fixed points in a Boolean network is reduced to solving the system of equations generated by the polynomial functions.
- **SAT-based methods**: Because there are several algorithms that solve SAT problem with a complexity better than 2^n (For example, Rolf in [70] gets solutions for 3-SAT in $O(1.323^n)$), in [3, 27, 28, 53, 76, 77] algorithms are developed based on reducing the problem of finding the fixed points of a Boolean network to define a function $\phi(x)$ associated with the Boolean network and finding which values x satisfy it. Of these investigations, [27, 28] are oriented to obtain all the fixed points of the network, while others are oriented to obtain a single fixed point in Boolean networks with canalyzing functions [3], or in AND-OR networks [53, 76, 77].
- **Methods based on Integer Programming**: As in the case of SAT, [1] develop an algorithm that transforms a Boolean network into a set of linear inequalities whose maximizes a linear function of the form $c^T x$ is associated with the fixed points of the network.

- **Strategic Sampling**: In [85], Zhang et al. develop a recursive algorithm that seeks to identify all the fixed points of a Boolean network. In the case of networks with a maximum in-degree of 2, the average time of this algorithm is $O(1.19^n)$ (where n is the number of nodes).

- **Methods based on Minimal Feedback Vertex Sets**: A solution that uses the network interaction digraph. In [2], Akutsu et al. present an algorithm that seeks to find the fixed points of a set of vertices S (typically, of smaller size than n) and these fixed points are propagated on the rest of the vertices of the network.

For a more detailed description of these and other methods see [39, 59, 81] and references therein.

So far, there is no known strategy that take advantage of that the set of fixed points of a Boolean network is invariant to the update schedule. For this reason, a solution is proposed that uses the information generated from the structural characteristics of the interaction digraph, to choose an asynchronous update schedule that allows to find the fixed points of the Boolean network in a faster way.

In this chapter we propose a new strategy to find the fixed points of a Boolean network which consists in fixing the local states of a positive feedback vertex set P of the regulatory graph and updating the other local states according to a sequential update schedule constructed from P. In this way we reduce the problem of finding the fixed points of a Boolean network of n components to check $2^{|P|}$ state configurations, where P can be smaller than a feedback vertex set of the network. Thus, the proposed algorithm is polynomial in the size of the network and exponential in the size of P. Consequently, this method can be very efficient, with respect to other methods, in Boolean networks with a small positive feedback vertex set, as for example ones with few positive cycles. A particular case of this latter are the strong inhibition Boolean networks [39].

For the construction of the proposed algorithm we proved first that the dynamical behavior of a Boolean network without positive cycles and with a fixed point is similar to an acyclic Boolean network, i.e. it quickly converges to the fixed point from any initial configuration.

4.2 Definitions and Notations

To explain the algorithms of this chapter, we introduce some notation. Given $f : \mathbb{B}^n \to \mathbb{B}^n$ a BN and $u \in [n]$, the *partial evaluation* of f in the local function f_u is the function $f^u : \mathbb{B}^n \to \mathbb{B}^n$ such that: $\forall x \in \mathbb{B}^n$, $f^u(x) = (x_1, \ldots, x_{u-1}, f_u(x), x_{u+1}, \ldots, x_n)$

In this way, a *sequential schedule* $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ is a permutation of $[n]$. The dynamics of a BN $f : \mathbb{B}^n \to \mathbb{B}^n$ updated according to π is defined by: $\forall x \in \mathbb{B}^n$, $f^\pi(x) = f^{\pi_n} \circ f^{\pi_{n-1}} \circ \cdots \circ f^{\pi_1}(x)$, i.e. if x is the state of the network in a time step, $f^\pi(x)$ is the state of the network in the next time step.

To avoid confusion, we use a special notation for self-compositions of f: we denote by $f^{\langle 0 \rangle}$ the identity on $\mathbb{B}^n$ and for $k \geq 1$ we set $f^{\langle k \rangle} = f \circ f^{\langle k-1 \rangle}$. The concatenation of k times the sequential schedule π is denoted π^k. In this way $f^{\pi^k} = (f^\pi)^{\langle k \rangle}$.

4.3 Boolean networks without positive cycles

The relationship between the fixed points and the positive and negative cycles in Boolean networks has been greatly studied [7, 15, 16, 63, 64, 65]. In particular, in [7, 63] was proved that every

Boolean network without positive cycles has at most one fixed point and, in addition, if the network has a source non-trivial strongly connected component, then it has no fixed points.

On the other hand, F. Robert studied the Boolean networks without cycles and proved in [67, 68] that these networks have a simple dynamical behavior with a unique attractor, which is a fixed point, and where all initial states quickly converge to it. More precisely, he proved the following theorem.

Theorem 4.1 (F. Robert)**.** *Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network such that $G(f)$ is acyclic. Then,*

1. *f has a unique fixed point $y \in \mathbb{B}^n$.*
2. *$\forall x \in \mathbb{B}^n,\ f^{\langle n \rangle}(x) = y$.*
3. *$\exists \pi$ a sequential schedule such that $\forall x \in \mathbb{B}^n,\ f^{\pi}(x) = y$.*

In order to understand the following results, the following definitions are necessary:

Definition 4.1. Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network, we denote the following sets (possibly empty):

$$I_1(f) = \{v \in [n] : \exists c \in \mathbb{B}, \forall x \in \mathbb{B}^n, f_v(x) = c\}.$$

In other words, $v \in I_1(f)$ if f_v is constant. For all $v \in I_1(f)$, we denote c_v to the constant value of $f_v(x)$.

Recursively, we define $\forall k \in \mathbb{N}, k \geq 2, I_k(f)$ in the following way:

$$I_k(f) = \{v \in [n] : \exists c \in \mathbb{B}, \forall x \in \mathbb{B}^n, f_v(c_u : u \in I_{k-1}(f); x_u : u \notin I_{k-1}(f)) = c\}.$$

Similarly, $v \in I_k(f)$ if the regulatory function associated to the component v with fixed value c_u on input $u \in I_{k-1}(f)$ is a constant function. $\forall v \in I_k(f)$, we denote c_v to the constant value of $f_v(c_u : u \in I_{k-1}(f); x_u : u \notin I_{k-1}(f))$.

From here, if $v \in I_k(f)$, then:

$$\forall t \geq k, \forall x \in \mathbb{B}^n, f_v^{\langle t \rangle}(x) = c_v.$$

In this case, we say that the component v is *fixed at iteration k of f with value c_v.*

Proposition 4.2. *The sets $I_k(f)$ have the following properties:*

1. *$\forall k \geq 1, I_k(f) \subseteq I_{k+1}(f)$.*
2. *If $\exists k \in \mathbb{N}, I_k(f) = I_{k+1}(f)$, then $\forall l \in \mathbb{N}, I_k(f) = I_{k+l}(f)$.*
3. *If $I_1(f) = \emptyset$, then $\forall k \in \mathbb{N}, I_k(f) = \emptyset$.*

Proof.

1. By induction on k. First, we prove that $I_1(f) \subseteq I_2(f)$. By definition, $v \in I_1(f)$ if f_v is a constant function. On the other hand, $v \in I_2(f)$ if f_v with fixed value c_u on each input $u \in I_1(f)$ is a constant function. Therefore, $I_1(f) \subseteq I_2(f)$.

   Let us suppose that $I_k(f) \subseteq I_{k+1}(f)$. Now, we prove that $I_{k+1}(f) \subseteq I_{k+2}(f)$. Analogously, $v \in I_{k+1}(f)$ if f_v with fixed value c_u on each input $u \in I_k(f)$ is a constant function. On the other hand, $v \in I_{k+2}(f)$ if f_v with fixed value c_u on each input $u \in I_{k+1}(f)$ is a constant function. Since $I_k(f) \subseteq I_{k+1}(f)$, the inclusion $I_{k+1}(f) \subseteq I_{k+2}(f)$ is direct.

2. To prove this, we prove that if $I_k(f) = I_{k+1}(f)$ then $I_{k+1}(f) = I_{k+2}(f)$.

   Let us suppose that $I_k(f) = I_{k+1}(f)$. If $I_{k+2}(f) = \emptyset$, by Property 1, $I_{k+1}(f) = \emptyset$. So, let v be a component in $I_{k+2}(f)$, by definition:

   $$\exists c \in \mathbb{B}, \forall x \in \mathbb{B}^n, f_v(c_u : u \in I_{k+1}(f); x_u : u \notin I_{k+1}(f)) = c.$$

   Since $I_k(f) = I_{k+1}(f)$,

   $$\forall x \in \mathbb{B}^n, f_v(c_u : u \in I_k(f); x_u : u \notin I_k(f)) = c.$$

   Therefore, $v \in I_{k+1}(f)$. For this reason, $I_{k+2}(f) \subseteq I_{k+1}(f)$ and since $I_{k+1}(f) \subseteq I_{k+2}(f)$, then $I_{k+1}(f) = I_{k+2}(f)$.

3. First, we prove that if $I_1(f) = \emptyset$, then $I_2(f) = \emptyset$.

   By contradiction, let us suppose that $I_1(f) = \emptyset$ and $I_2(f) \neq \emptyset$. Let v be a component in $I_2(f)$, then:

   $$\exists c \in \mathbb{B}, \forall x \in \mathbb{B}^n, f_v(c_u : u \in I_1(f); x_u : u \notin I_1(f)) = f_v(x) = c.$$

   Therefore, by definition, $v \in I_1(f)$ which is a contradiction.

   Since $I_1(f) = I_2(f) = \emptyset$, by Property 2, $\forall k \in \mathbb{N}, I_k(f) = \emptyset$.

□

Definition 4.2. Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network. We say that f is an *irreducible network after k iterations* or *k-irreducible network* if $I_k(f) = I_{k+1}(f)$. If $I_1(f) = \emptyset$, we say that f is an *irreducible network*.

Definition 4.3. Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network and $k \in \mathbb{N}$. We define $f^{I_k} : \mathbb{B}^n \to \mathbb{B}^n$ the *Boolean network associated to f and $I_k(f)$* by:

$$\forall x \in \mathbb{B}^n, f^{I_k}(x) = f(c_u : u \in I_k(f); x_u : u \notin I_k(f)).$$

Remark 4.1. *Note that if f is an irreducible network, then $f^{I_k}(x) = f(x)$.*

Example 4.1. Figure 4.1 shows an example of a Boolean network f and the network f^{I_3}, according to Definition 4.3. Since f_4 and f_5 are the only constant functions, $I_1(f) = \{4, 5\}$ (with $c_4 = 1$ and $c_5 = 0$). On the other hand, $f_1(x_1, x_2, x_3, 1, 0, x_6, x_7) = \overline{x_1} \vee 1 \vee 0 = 1$, hence $1 \in I_2(f)$. Moreover, since there is no other component that satisfies the same condition, $I_2(f) = \{1, 4, 5\}$.

Analogously, $I_3(f) = \{1, 4, 5, 6, 7\}$. In a new iteration, $I_4(f) = I_3(f)$, so f is 3-irreducible.

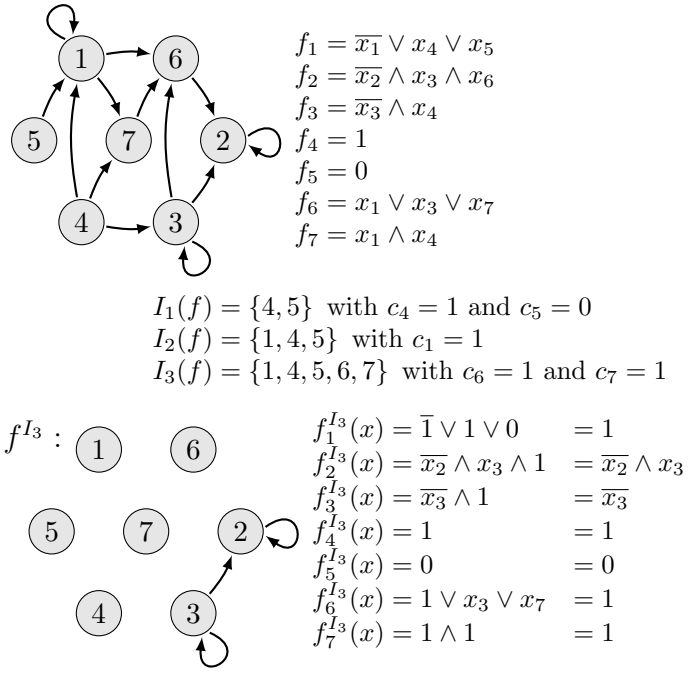


Figure 4.1: Example of Boolean networks f and f^{I_3}.

Remark 4.2. *It is important to remark that* $\forall k \in \mathbb{N}$ *and for all* $f : \mathbb{B}^n \to \mathbb{B}^n$*:*

1. $V(G(f^{I_k})) = V(G(f))$.
2. $A(G(f^{I_k})) \subseteq A(G(f))$.
3. *If* $(u, v) \in A(G(f^{I_k}))$*, then the sign of* (u, v) *is the same in* $G(f)$ *and* $G(f^{I_k})$.
4. y *is a fixed point of* f *if and only if* y *is a fixed point of* f^{I_k} *and* $\forall v \in I_k(f), c_v = y_v$.

Remark 4.3. *Let* $f : \mathbb{B}^n \to \mathbb{B}^n$ *be a Boolean network such that* $I_k(f) = I_{k+1}(f) \neq \emptyset$*. Then,* $G(f^{I_k})$ *has the following properties:*

1. $\forall u \in I_k(f),\ d^+_{G(f^{I_k})}(u) = 0$.
2. *If there exists a non-trivial strongly connected component (SCC), then there is a source non-trivial strongly connected component (i.e. a SCC where the arcs incident to the vertices of the component have their origin in vertices of the same component).*

Next, we show that whether a Boolean network without positive cycles has a fixed point, then its dynamical behavior is like a Boolean network without cycles. More precisely, we prove the following theorem.

Theorem 4.3. *Let* $f : \mathbb{B}^n \to \mathbb{B}^n$ *be a Boolean network without positive cycles. Then, the following propositions are equivalent:*

1. f *has a unique fixed point* $y \in \mathbb{B}^n$.
2. $\forall x \in \mathbb{B}^n,\ f^{\langle n \rangle}(x) = y$.
3. $\exists \pi$ *a sequential schedule such that* $\forall x \in \mathbb{B}^n,\ f^{\pi}(x) = y$.

Proof. The proof of (2) $\Rightarrow$ (1) and (3) $\Rightarrow$ (1) are straightforward, so we prove that (1) $\Rightarrow$ (2) and (1) $\Rightarrow$ (3).

Both proofs are very similar, so we give the proof for (1) $\Rightarrow$ (2).

First, notice that $\{I_1, I_2 \setminus I_1, \ldots, I_k \setminus I_{k-1}\}$ is a partition of $[n]$, we define an order of the vertices $\pi = (\pi_1, \ldots, \pi_n)$ such that:

$$\pi_i \in (I_j \setminus I_{j-1}) \ \wedge \ \pi_{i'} \in (I_{j'} \setminus I_{j'-1}) \ \wedge \ j < j' \Longrightarrow i < i'.$$

Now we prove by induction that $\forall i \in \{1 \ldots, k\}, \forall t \geq k, \ f_{\pi_i}^{\langle t \rangle}(x) = y_{\pi_i}$.

If $k = 1$, $\pi_1 \in I_1(f)$, then f_{π_i} is a constant function and therefore $\forall t \geq 1$, $f_{\pi_1}^{\langle t \rangle}(x) = y_{\pi_1}$.

Let us suppose that $\forall i \in \{1 \ldots, k\}, \forall t \geq k, \ f_{\pi_i}^{\langle t \rangle} t(x) = y_{\pi_i}$.

Now we need to prove that $\forall t \geq k+1, \ f_{\pi_{k+1}}^{\langle t \rangle}(x) = y_{\pi_{k+1}}$.

$$f_{\pi_{k+1}}^{\langle t \rangle}(x) = f_{\pi_{k+1}}(f^{\langle t-1 \rangle}(x)).$$

Since $t \geq k+1$, $t-1 \geq k$, then by hypothesis of induction:

$$f_{\pi_{k+1}}^{\langle t \rangle}(x) = f_{\pi_{k+1}}(y_{\pi_i} : i \leq k; \tilde{x}_{\pi_i} : i > k).$$

By definition of π, if $\pi_{k+1} \in I_j \setminus I_{j-1}$ then $I_{j-1} \subseteq \{\pi_1, \ldots, \pi_k\}$, therefore, by definition of I_j:

$$f_{\pi_{k+1}}^{\langle t \rangle}(x) = y_{\pi_{k+1}}.$$

The proof for (1) $\Rightarrow$ (3) works the same way to prove that $\forall i \in \{1, \ldots, k\}$, $f_{\pi_i}^{(\pi_1, \ldots, \pi_k)}(x) = f_{\pi_i}((f^{\pi_{k-1}} \circ \cdots \circ f^{\pi_1})(x)) = y_{\pi_i}$. □

A difference between BNs with acyclic regulatory graphs and BNs without positive cycles is that, in the first case, the sequential schedule referred in F. Robert's theorem depends only on the regulatory graph and not on the function. This latter is not true in the case of BNs without positive cycles as shown in Example 4.2.

Example 4.2. Let $f : \mathbb{B}^3 \to \mathbb{B}^3$ be a BN with interaction graph shown in Figure 4.2 and $\pi = (1, 2, 3)$ a sequential schedule. If we choose the functions:

$$f_1(x) = 1, \qquad f_2(x) = \neg x_1 \wedge x_3, \qquad f_3(x) = x_1 \wedge \neg x_2,$$

then $y = (1, 0, 1)$ is a fixed point of f and $\forall x \in \mathbb{B}^3, f^{\pi}(x) = y$.

On the contrary, if we choose the functions:

$$f_1(x) = 1, \qquad f_2(x) = \neg x_1 \vee x_3, \qquad f_3(x) = x_1 \vee \neg x_2,$$

$y' = (1, 1, 1)$ is a fixed point of f, but $f^{\pi}(1, 1, 0) = (1, 0, 1) \neq y'$. In this case, if we choose $\pi' = (1, 3, 2)$, then $\forall x \in \mathbb{B}^3, f^{\pi'}(x) = y'$.

On the other hand, as consequence of Theorem 4.3 the following corollary shows us that the problem of determining the existence of a fixed point in a Boolean network without positive cycles can be solved in polynomial time if the regulatory functions can be evaluated in polynomial time, i.e. in $O(n^k)$ with $k \geq 0$ a constant. Examples of regulatory functions whose evaluation can be done in polynomial time are: threshold functions, hierarchically canalizing functions, strong-inhibition functions, with a bounded in-degree, etc.

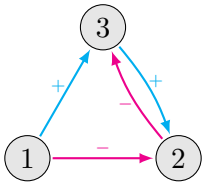


Figure 4.2: Regulatory graph of a network without positive cycles.

Corollary 4.4. *Determining whether a Boolean network $f : \mathbb{B}^n \to \mathbb{B}^n$ without positive cycles has a fixed point can be done through $n+1$ applications of f on any state of the network.*

In [7, 63] was proved the following theorem, known as the Thomas's first rule:

Theorem 4.5 (Thomas's first rule)**.** *Let $f : \mathbb{B}^n \to \mathbb{B}^n$ a BN without positive cycles. Then, f has at most one fixed point.*

A direct corollary from previous theorem is that if f is a BN without positive cycles such that has an initial non-trivial strong component, then f has no fixed points [7].

Proof of Corollary 4.4. Thus, if y is a fixed point of f, then $\forall x \in \mathbb{B}^n, f^{\langle n \rangle}(x) = y$. Hence, it suffices to check for some state x (for example, $x = \vec{0} := (0, 0, \ldots, 0)$), if $f^{\langle n \rangle}(x)$ is a fixed point of the network. In other words, if $f^{\langle n \rangle}(x) = f^{\langle n+1 \rangle}(x)$, then $f^{\langle n \rangle}(x)$ is a fixed point of f. Otherwise, f has no fixed points. □

From Theorem 4.5 we can state the following result.

Lemma 4.6. *Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a RBN without positive cycles. f has a unique fixed point if and only if there exists $k \in \{1, \ldots, n\}$ such that $\emptyset \neq I_1(f) \subset I_2(f) \subset \cdots \subset I_k(f) = [n]$.*

Proof. ($\Rightarrow$) Let us suppose that f has a unique fixed point $y \in \mathbb{B}^n$, we prove that $\forall k \in \{1, \ldots, n\}, |I_k| \geq k$. Since f does not have any positive cycle, by theorem Thomas's first rule all initial strong components of $G(f)$ are trivial, hence there exists a vertex $v_1 \in [n]$ whose regulatory function is constant. Then, $v_1 \in I_1(f)$ and $|I_1(f)| \geq 1$.

By contradiction, let us suppose that $\exists l \in \{2, \ldots, n\}$ such that $|I_l| < l$. So, by Property 1 of Proposition 4.2:

$$\exists k \leq (n-1), I_k(f) = I_{k+1}(f) \neq [n].$$

In this case, $\forall v \notin I_k(f)$, $\exists u \notin I_k(f)$, such that f_v depends on x_u, i.e., $\forall v \notin I_k(f)$, $\exists u \notin I_k(f)$, $(u, v) \in A(f^{I_k})$.

Then, the in-degree of all vertices in $[n] \setminus I_k(f)$ is greater than 0, hence there exists a non-trivial strong component of $G(f^{I_k})$ induced by the vertices in $[n] \setminus I_k(f)$. Since $\forall u \in I_k(f)$, $d^+_{G(f^{I_k})}(u) = 0$, there exists an initial non-trivial strong component in $G(f^{I_k})$ and thus f^{I_k} has no fixed points. Hence, by Remark 4.2.4) f has no fixed points, which is a contradiction.

Therefore, if $k \leq (n-1)$, then $I_k(f) \subsetneq I_{k+1}(f)$ or $I_k(f) = [n]$.

($\Leftarrow$) If $I_n(f) = [n]$, $\forall v \in [n], \forall t \geq n, \forall x \in \mathbb{B}^n, f_v^{\langle t \rangle}(x) = c_v$. Then, f has a fixed point and, because f does not have any positive cycles, this fixed point is unique. □

Theorem 4.3 is the main result of this article, this theorem allows the construction of the proposed algorithm. This result shows that the behavior of Boolean network without positive

cycles and that has a fixed point as attractor, is very similar to the behavior of an acyclic Boolean network. In fact, we can partition the set vertices according to the time step in which they reach their stable state. Since a partition of a set of n elements has at most n sets, this steady state is reached in at most n iterations of the network.

4.4 Methods

4.4.1 Algorithm to detect the fixed points in Boolean networks from a PFVS

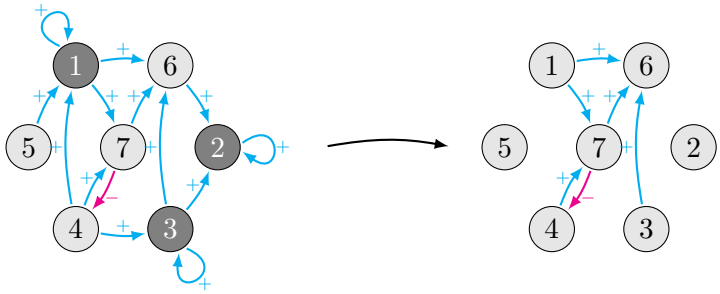
In [7] was proved that for every RBN f there is an injection between its fixed points and the local states of a PFVS of its regulatory graph $G(f)$, allowing to conclude that the maximum number of fixed points of a RBN f is $2^{\tau^+(G(f))}$. A simple exercise shows that this result is also valid in general Boolean networks [65]. In this way, we develop an algorithm that considers all the possible states of a given PFVS and efficiently verifies, by application of f, which of them produces a fixed point for the network. This is achieved thanks to the dynamical behavior of a BN without positive cycles described in Theorem 4.3. First, we give some definitions and results.

Definition 4.4. Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a BN, P a PFVS of $G(f)$ and $a : P \to \mathbb{B}$ a function, which can be represented by a vector $a \in \mathbb{B}^{|P|}$. We define the Boolean network $fa : \mathbb{B}^n \to \mathbb{B}^n$ as follows:

$$\forall v \in [n], \forall x \in \mathbb{B}^n,\ fa_v(x) = \begin{cases} a(v) & \text{if } v \in P, \\ f_v(x) & \text{if } v \notin P. \end{cases}$$

In other words, fa is the Boolean network obtained from f setting the value of each element $v \in P$ to $a(v)$.

Example 4.3. Figure 4.3 shows an example of a BN f and a BN fa, where $a(1) = a(3) = 0$ and $a(2) = 1$, according to Definition 4.4. Dark gray vertices represent P.



$f_1(x) = x_1 \vee x_4 \vee x_5$
$f_2(x) = x_2 \wedge x_3 \wedge x_6$
$f_3(x) = x_3 \wedge x_4$
$f_4(x) = \neg x_7$
$f_5(x) = 0$
$f_6(x) = x_1 \vee x_3 \vee x_7$
$f_7(x) = x_1 \wedge x_4$

$fa_1(x) = 0$
$fa_2(x) = 1$
$fa_3(x) = 0$
$fa_4(x) = \neg x_7$
$fa_5(x) = 0$
$fa_6(x) = x_1 \vee x_3 \vee x_7$
$fa_7(x) = x_1 \wedge x_4$

Figure 4.3: A Boolean network f and a *network without positive cycles* fa.

Note that $G(fa) = G(f) - \{(u,v) \in A : \ v \in P\}$. Hence, fa is a BN without positive cycles. As a direct result, we have the following proposition.

Proposition 4.7. *Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a BN, $P \neq \emptyset$ a PFVS of $G(f)$ and $a : P \to \mathbb{B}$ a function. $x \in \mathbb{B}^n$ is a fixed point of f such that $\forall v \in P, x_v = a(v)$ if and only if:*

1. *x is a fixed point of fa,*
2. *$\forall v \in P, f_v(x) = a(v)$.*

Proof. ($\Rightarrow$) Let $x \in \mathbb{B}^n$ be a fixed point of f such that $\forall v \in P, x_v = a(v)$, then, as x is a fixed point of f, then $\forall v, f_v(x) = x_v$. Hence, by Definition 4.4, if $v \notin P, fa_v(x) = f_v(x) = x_v$, and if $v \in P, fa_v(x) = a(v) = x_v$. Therefore, $\forall v, fa_v(x) = x_v$, thus, x is a fixed point of fa and, by hypothesis, condition (2) also holds.

($\Leftarrow$) If $x \in \mathbb{B}^n$ is a fixed point of fa, then $\forall v, fa_v(x) = x_v$. Hence, by Definition 4.4, if $v \notin P, x_v = fa_v(x) = f_v(x)$.

Moreover, If $\forall v \in P, x_v = fa_v(x) = a(v) = f_v(x)$.

Therefore, $\forall v, x_v = f_v(x)$, then, x is a fixed point of f and $\forall v \in P, x_v = a(v)$. □

The following proposition gives a bound on the number of iterations to reach the steady state (when this exists) of a Boolean network without positive cycles with a sequential update schedule compatible with the PFVS. The idea is show that in each sequential iteration of the network at least one vertex of the PFVS fixes its value. In this way, after at most $|P| + 1$ iterations we can check if the network has a fixed point.

In this way, we use Proposition 4.7 to define Algorithm 4.1, which finds the fixed points of a given BN.

Proposition 4.8. *Given $f : \mathbb{B}^n \to \mathbb{B}^n$ be a BN, whose regulatory functions can be evaluated in time $O(n^k), k \geq 0$, and P a PFVS of $G(f)$ Algorithm 4.1 finds the set of fixed points of f in time $O(2^{|P|}n^{2+k})$.*

Proof. The correctness of the algorithm is direct from Proposition 4.7 and Corollary 4.4. In particular, if $P \neq \emptyset$, the instruction $x \leftarrow fa^{\langle n \rangle}(\vec{0})$ obtains a fixed point candidate (after n executions of fa, for some $a \in \mathbb{B}^{|P|}$) that is checked in the next line according to Proposition 4.7. Otherwise, i.e. f has no positive cycles, $x \leftarrow f^{\langle n \rangle}(\vec{0})$ obtains a fixed point candidate (after n^2 evaluations of regulatory functions) that is checked in the next line according to Corollary 4.4. Finally, the running time of the algorithm is given by the total executed time in the applications of the regulatory functions, which in the worst case is $O(2^{|P|}n^{2+k})$. □

In order to make the Algorithm 4.1 faster, and following the ideas introduced in [11], we use a sequential update schedule that allows a faster convergence to the fixed points when they exist. By Theorem 4.3, the BNs without positive cycles have a sequential update schedule such that they converge to the only fixed point, when there exists, in only one step. However, determining such a schedule can be as difficult as finding the fixed points of the network, because it depends on the value of each local activation functions as shown in Example 4.2. We propose to use a sequential update schedule which depends only on the regulatory graph structure of input network. More precisely, given a FVS F and a PFVS P contained in it, we determine in polynomial time a sequential scheme such that the global activation function applied with this schedule in each

Algorithm 4.1: BasicFixedPoint

Input: f a BN with n components and P a PFVS of $G(f)$.
Output: S the set of fixed points of f.

```
S ← ∅;
if P ≠ ∅ then
    foreach a ∈ 𝔹^|P| do
        x ← fa^⟨n⟩(0⃗);
        if (fa(x) = x) ∧ (∀u ∈ P, f_u(x) = a(u)) then S ← S ∪ {x};
    end
else
    x ← f^⟨n⟩(0⃗);
    if (f(x) = x) then S ← {x};
end
return S
```

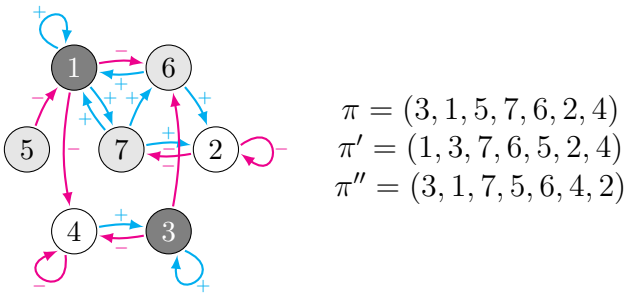


Figure 4.4: A digraph $G(f)$ and orders compatible with F and P. Set P is denoted for dark gray vertices and set F for light gray vertices.

iteration fixes at least the value of one vertex in $F \setminus P$ (in the case that this is possible), then the network is updated once more to fix the remaining vertices. Thus, the number of applications in the new algorithm is reduced from n to $|F| - |P| + 1$.

Definition 4.5. Given $f : \mathbb{B}^n \to \mathbb{B}^n$ a BN, F a FVS of $G(f)$ and $P \subseteq F$ a PFVS of $G(f)$, We say that a permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ on the set $[n]$ is an *order compatible with F and P* if it satisfies the following properties:

1. $\forall \pi_i \in (F - P), \forall \pi_j \notin (F - P),\ i > j.$
2. $\forall \pi_i \in P, \forall \pi_j \notin P,\ i < j.$
3. $\forall \pi_i, \pi_j \notin F,\ (\pi_i, \pi_j) \in A \Longrightarrow i < j.$

In the particular case of $P = \emptyset$, i.e. f is a BN without positive cycles, π is said to be compatible with F if it satisfies 1 and 3.

Example 4.4. Figure 4.4 shows examples of orders compatible with F and P. Dark gray vertices represent the PFVS P and light gray vertices represent $F \setminus P$.

Proposition 4.9. *Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a BN, such that the regulatory graph $G(f)$ does not have positive cycles ($P = \emptyset$), F be a FVS of $G(f)$, π be an order compatible with F, and $y \in \mathbb{B}^n$. y is the unique fixed point of f if and only if $\forall x \in \mathbb{B}^n, (f^\pi)^{\langle |F|+1 \rangle}(x) = y$.*

Proof. ($\Rightarrow$) Let us suppose that $y \in \mathbb{B}^n$ is the unique fixed point of f. Without loss of generality, let us suppose that $\pi = (1, 2, \ldots, n)$. Then, by Theorem 4.3, there exist

$I_1(f) \subsetneq I_2(f) \subsetneq \cdots \subsetneq I_{k-1}(f) \subsetneq I_k(f) = [n]$

such that:

$$\forall j \in \{1, \ldots, k\}, \forall v \in I_j(f), \forall t \geq j, \forall x \in \mathbb{B}^n, f_v^{\langle t \rangle}(x) = y_v.$$

Now, we prove that by iterating f according to the order π, the n vertices fix their value in m iterations (with $m \leq |F| + 1$).

To perform this, let us consider the following sequence of indices o_i defined as follows:

$$\begin{aligned} o_1 &= \min \{j \in \{1, \ldots, k\} : (I_j(f) \cap F) \neq \emptyset\}, \\ \forall p \geq 2, o_p &= \min \{j > o_{p-1} : ((I_j(f) \setminus I_{j-1}(f)) \cap F \neq \emptyset) \vee (j = k)\}. \end{aligned}$$

Notice that $o_1 < o_2 < \cdots < o_m = k$ and $m \leq |F| + 1$.

We prove that $\forall i \in \{1, \ldots, m\}, \forall u \in I_{o_i}, \forall t \geq i, \forall x \in \mathbb{B}^n, (f^\pi)_u^{\langle t \rangle}(x) = y_u$.

By contradiction, let us suppose that:

$$\exists i \in \{1, \ldots, m\}, \exists u \in I_{o_i}, \exists t \geq i, \exists x \in \mathbb{B}^n, (f^\pi)_u^{\langle t \rangle}(x) \neq y_u.$$

Let:

$$\begin{aligned} i_* &= \min \left\{ i \in \{1, \ldots, m\} : \exists u \in I_{o_i}, \exists t \geq i, \exists x \in \mathbb{B}^n, (f^\pi)_u^{\langle t \rangle}(x) \neq y_u \right\} \text{ and} \\ l_* &= \min \left\{ l \leq o_{i_*} : \exists u \in I_l, \exists t \geq i_*, \exists x \in \mathbb{B}^n, (f^\pi)_u^{\langle t \rangle}(x) \neq y_u \right\}. \end{aligned}$$

Let $u_* \in I_{l_*} \setminus I_{l_*-1}$ be such that $\exists t \geq i_*, \exists x \in \mathbb{B}^n, (f^\pi)_{u_*}^{\langle t \rangle}(x) \neq y_{u_*}$. Then, by Definition 4.1, $\forall t \geq i_*, \forall x \in \mathbb{B}^n$:

$$\begin{aligned} (f^\pi)_{u_*}^{\langle t \rangle}(x) &= f_{u_*}^\pi((f^\pi)^{\langle t-1 \rangle}(x)) \\ &= f_{u_*}((f^\pi)_i^{\langle t \rangle}(x) : i < u_*; (f^\pi)_i^{\langle t-1 \rangle} : i \geq u_*). \end{aligned}$$

Notice that $l_* > 1$, then:

$$\begin{aligned} (f^\pi)_{u_*}^{\langle t \rangle}(x) = f_{u_*}((f^\pi)_v^{\langle t \rangle}(x) : v < u_* \wedge v \in I_{l_*-1}; (f^\pi)_v^{\langle t \rangle}(x) : v < u_* \wedge v \notin I_{l_*-1}; \\ (f^\pi)_v^{\langle t-1 \rangle}(x) : v \geq u_* \wedge v \in I_{l_*-1}; (f^\pi)_v^{\langle t-1 \rangle}(x) : v \geq u_* \wedge v \notin I_{l_*-1}). \end{aligned}$$

By definition of l_*, if $v \in I_{l_*-1}$, then $(f^\pi)_v^{\langle t \rangle}(x) = y_v$. Moreover, If $v \geq u_*$, then $v \in F$ and, therefore, if $v \in F$ and $v \in I_{l_*-1}$, then $v \in I_{o_{(i_*-1)}}$ and, thus, if $v \geq u_*$ and $v \in I_{l_*-1}$, then $(f^\pi)_v^{\langle t-1 \rangle}(x) = y_v$. Also, if $i_* = 1$, then $\forall l \leq o_1, F \cap I_{l-1} = \emptyset$:

$$\begin{aligned} (f^\pi)_{u_*}^{\langle t \rangle}(x) &= f_{u_*}(y_v : v < u_* \wedge v \in I_{l_*-1}; (f^\pi)_v^{\langle t \rangle}(x) : v < u_* \wedge v \notin I_{l_*-1}; \\ &\quad y_v : v \geq u_* \wedge v \in I_{l_*-1}; (f^\pi)_v^{\langle t-1 \rangle} : v \geq u_* \wedge v \notin I_{l_*-1}) \\ &= f_{u_*}(y_v : v \in I_{l_*-1}; (f^\pi)_v^{\langle t \rangle}(x) : v < u_* \wedge v \notin I_{l_*-1}; \\ &\quad (f^\pi)_v^{\langle t-1 \rangle}(x) : v \geq u_* \wedge v \notin I_{l_*-1}). \end{aligned}$$

By Definition 4.1, $\forall t \geq i_*, \forall x \in \mathbb{B}^n, (f^\pi)_{u_*}^{\langle t \rangle}(x) = y_{u_*}$, which is a contradiction.

Therefore, in at most m iterations of f^π, all vertices in $I_{o_m}(f) = I_k(f)$ are fixed.

($\Leftarrow$) Let us suppose that $\forall x \in \mathbb{B}^n, (f^\pi)^{\langle |F|+1 \rangle}(x) = y$, then:

$$f^\pi(y) = f^\pi((f^\pi)^{\langle |F|+1 \rangle}(x)) = (f^\pi)^{\langle |F|+2 \rangle}(x) = (f^\pi)^{\langle |F|+1 \rangle}(f^\pi(x)) = y.$$

Since $f^\pi(y) = y$, then, y is a fixed point of f. □

Theorem 4.10. *Let F be a FVS of $G(f)$, whose regulatory functions can be evaluated in time $O(n^k)$, $k \geq 0$, and $P \subseteq F$ a PFVS of $G(f)$, Algorithm 4.2 finds the set of fixed points of f in time $O(2^{|P|}(|F| - |P| + 1)n^{1+k})$.*

Proof. As the Algorithm 4.1, the correctness of this algorithm is based on the fact that its execution demonstrates each of the necessary conditions according to the Proposition 4.7. The difference is that the way to find the fixed point of fa (for some $a \in \mathbb{B}^{|P|}$) is done according to Proposition 4.9, i.e. instead of executing n times fa, fa^π is executed $|F| - |P| + 1$ times. For this reason, the complexity of this algorithm is $O(2^{|P|}(|F| - |P| + 1)n^{1+k})$. □

Algorithm 4.2: FixedPoint

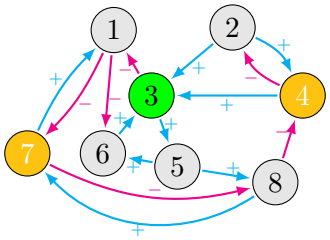
Input: f a BN with n components, F a FVS of $G(f)$, $P \subseteq F$ a PFVS of $G(f)$, π an order compatible with F and P.
Output: S the set of fixed points of f.

```
1  m ← |F − P|;
2  S ← ∅;
3  if P ≠ ∅ then
4  |  foreach a ∈ B^|P| do
5  |  |  x ← (fa^π)^⟨m+1⟩(0⃗);
6  |  |  if (fa(x) = x) ∧ (∀u ∈ P, f_u(x) = a(u)) then S ← S ∪ {x};
7  |  end
8  else
9  |  x ← (f^π)^⟨m+1⟩(0⃗);
10 |  if (f(x) = x) then S ← S ∪ {x};
11 end
12 return S
```

4.4.2 Example of the algorithms

Example 4.5. This example shows the operation of the FixedPoint-Algorithm with a Boolean network with $\tau^+ = 1$ and $\tau = 3$. Note that in the tables in steps 3 and 5, the purple cells represent the components that are fixed according to Definition 4.1

1. *Given the following network, where $\{3, 4, 7\}$ is a FVS and $\{3\}$ is a PFVS:*



$$f_1 = \overline{x_3} \vee x_7 \qquad f_5 = x_3$$
$$f_2 = \overline{x_4} \qquad f_6 = \overline{x_1} \vee x_5$$
$$f_3 = (x_2 \wedge x_4) \vee (x_2 \wedge x_6) \vee (x_4 \wedge x_6) \qquad f_7 = \overline{x_1} \vee x_8$$
$$f_4 = x_2 \vee \overline{x_8} \qquad f_8 = x_5 \wedge \overline{x_7}$$

2. *We choose an order compatible with the FVS and PFVS selected.*
   In this case, we choose $\pi = (3, 1, 2, 5, 6, 8, 4, 7)$.

3. *If* $a(3) = 0$, *the execution of* $(fa)^\pi$ *would be the following:*

		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
$(fa)^\pi$	3	0	0	0	0	0	0	0	0
	1	1	0	0	0	0	0	0	0
	2	1	1	0	0	0	0	0	0
	5	1	1	0	0	0	0	0	0
	6	1	1	0	0	0	0	0	0
	8	1	1	0	0	0	0	0	0
	4	1	1	0	1	0	0	0	0
	7	1	1	0	1	0	0	0	0
$((fa)^\pi)^2$	3	1	1	0	1	0	0	0	0
	1	1	1	0	1	0	0	0	0
	2	1	0	0	1	0	0	0	0
	5	1	0	0	1	0	0	0	0
	6	1	0	0	1	0	0	0	0
	8	1	0	0	1	0	0	0	0
	4	1	0	0	1	0	0	0	0
	7	1	0	0	1	0	0	0	0
$((fa)^\pi)^3$	3	1	0	0	1	0	0	0	0
	1	1	0	0	1	0	0	0	0
	2	1	0	0	1	0	0	0	0
	5	1	0	0	1	0	0	0	0
	6	1	0	0	1	0	0	0	0
	8	1	0	0	1	0	0	0	0
	4	1	0	0	1	0	0	0	0
	7	1	0	0	1	0	0	0	0

4. *Since* $(fa)(10010000) = 10010000$ *and* $f_3(1001000) = 0 = a(3)$, *then* 1001000 *is a fixed point of* f

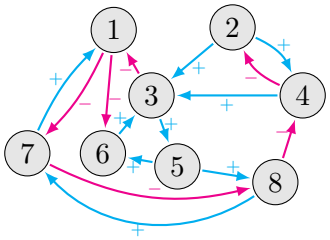
5. *If* $a(3) = 1$, *the execution of* $(fa)^\pi$ *would be the following:*

		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
$(fa)^\pi$	3	0	0	1	0	0	0	0	0
	1	0	0	1	0	0	0	0	0
	2	0	1	1	0	0	0	0	0
	5	0	1	1	0	1	0	0	0
	6	0	1	1	0	1	1	0	0
	8	0	1	1	0	1	1	0	1
	4	0	1	1	1	1	1	0	1
	7	0	1	1	1	1	1	1	1
$((fa)^\pi)^2$	3	0	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1
	2	1	0	1	1	1	1	1	1
	5	1	0	1	1	1	1	1	1
	6	1	0	1	1	1	1	1	1
	8	1	0	1	1	1	1	1	0
	4	1	0	1	1	1	1	1	0
	7	1	0	1	1	1	1	0	0
$((fa)^\pi)^3$	3	1	0	1	1	1	1	0	0
	1	0	0	1	1	1	1	0	0
	2	0	0	1	1	1	1	0	0
	5	0	0	1	1	1	1	0	0
	6	0	0	1	1	1	1	0	0
	8	0	0	1	1	1	1	0	1
	4	0	0	1	0	1	1	0	1
	7	0	0	1	0	1	1	1	1

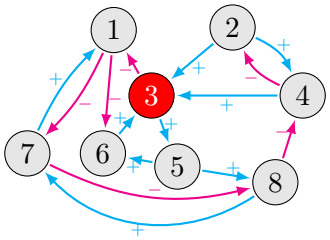
6. *Since* $(fa)(00101111) = 11101110$, *does not exist fixed point of* f *such that* $f_3(x) = 1$

Example 4.6. This example shows the operation of the PFVS-Algorithm. Green vertices represent vertices in P, red vertices represent vertices in $R \setminus O$, yellow vertices represent vertices in Y, orange vertices represent vertices in R and O and white vertices represent vertices in U.

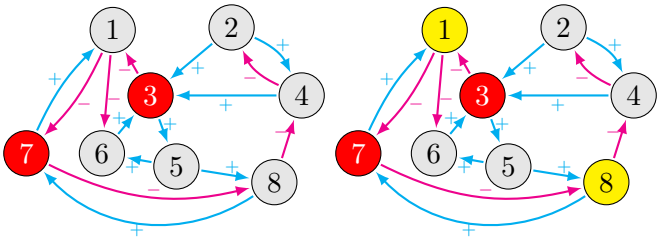
1. *Given a labeled digraph and an order over its vertices (3,7,2,5,1,4,8,6). All vertices are added to* U.



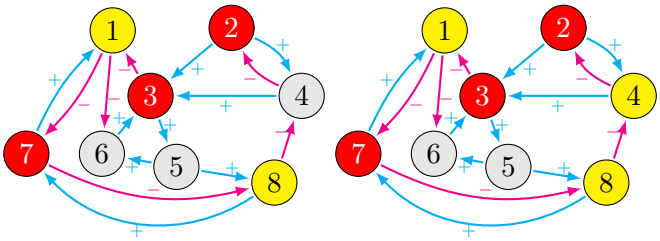
2. *The vertex in* U *with lower index (the vertex 3) is discarded from PFVS (added to* R*). Since* $\forall v \in U \cup Y$, *the subgraph induced by* $R \cup \{v\}$ *has no circuits, a new vertex is selected.*



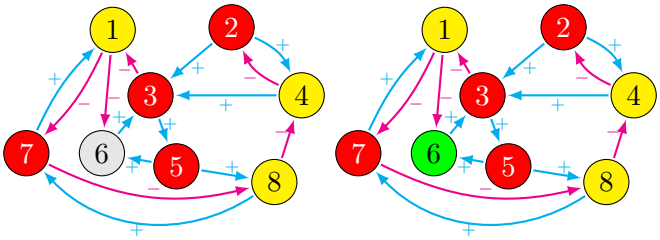
3. *The next vertex (7) is discarded from PFVS. Since the subgraph induced by* $R \cup \{1\}$ *has a negative circuit, (1) is added to* Y*. Similarly, (8) is added to* Y*.*



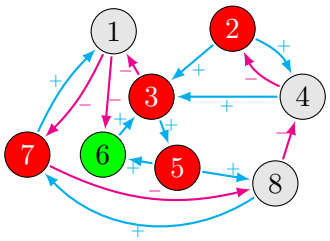
4. *The next vertex (2) is discarded from PFVS. (4) is added to* Y*.*



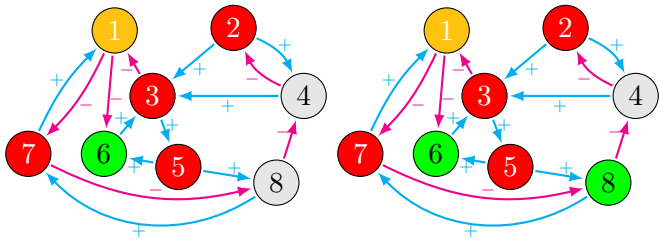
5. *The next vertex (5) is discarded from PFVS. Since the subgraph induced by* $R \cup \{6\}$ *has a positive circuit, (6) is included in the PFVS (added to* G*). Since* U *is empty, the first step is finished.*



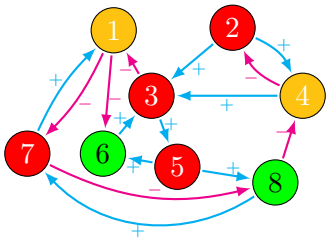
6. *For the second step (and following), all vertices in* Y *are added to* U*.*



7. *The vertex in U with lower index (1) is discarded from the PFVS (added to R and O). Since the subgraph induced by $R \cup \{8\}$ has a positive circuit (1,7,8,7,1), (8) is included in the PFVS.*



8. *The next vertex (4) is discarded from PFVS. Since U is empty, the second step is finished. Since no vertex in Y, the algorithm ends.*



4.4.3 Building a PFVS

FixedPoint algorithm requires as input a FVS F and a PFVS $P \subseteq F$ of the regulatory graph of a Boolean network. It is known that the problems of finding a minimum FVS and a minimum PFVS in a signed digraph are both NP-complete [44, 55]. In this section we propose a polynomial algorithm that allows to find a PFVS (not necessarily minimal) and a minimal FVS containing it.

Let $G(f) = (V, A)$ be the signed regulatory graph of a BN f with n components. Given $(v_1, v_2, \ldots, v_n)$ an order over V, Algorithm 4.3 classifies the vertices of $G(f)$ in the following sets:

- P: A set of vertices that is a PFVS of $G(f)$.
- O: A set of vertices such that $P \cup O$ is a minimal FVS of $G(f)$.
- R: The rest of the vertices of $G(f)$.

In addition, the algorithm considers the following auxiliary sets:

- Y: A set of vertices that covers some circuit.
- U: The vertices that have not yet been assigned to any set.

The operation of the algorithm is as follows:

First, we classify vertices with positive loop directly into P. Subsequently, the negative loops are removed from the arcs of $G(f)$.

Then, the first phase begins. All vertices that are not in P or R, are incorporated in U to be visited. Then, the first vertex in U (according to the input order) is incorporated into R (i.e., it is discarded from the PFVS) and if this vertex originally had a negative loop, it is incorporated into O (since that vertex has to be part of the FVS). Subsequently, for each vertex $v \in U \cup Y$, G'' (the subdigraph induced by $R \cup \{v\}$) is calculated, where, if G'' has a circuit, v is removed

from U and from Y (since v covers some circuit in $G(f)$), and then, if G'' has a positive circuit, v is incorporated in P (since v is the only unclassified vertex of the positive circuit of G''), and otherwise, v is incorporated into Y (since it covers some circuit, but no positive circuit is known, to add v to P). Once U is empty, the first phase is terminated.

If at the end of the first phase, there are vertices that are neither in P nor in R (for example, they are in Y), a second phase begins. The only difference of this phase (and subsequent ones) with respect to the first phase, is that if a vertex is sent to R it is automatically sent to O (since this phase, all unclassified vertices cover some circuit, but being discarded, it means that they do not cover any positive circuit, so they are part of the FVS).

If at the end of the second phase some unclassified vertices remain, a third phase begins under the same conditions of the second phase, and so on.

Notice that, the constructed PFVS is not minimal, since a vertex can be included in P because it covers a positive circuit (not necessarily a positive cycle), but the FVS is minimal, because the algorithm works looking for cycles in $G(f) - P - O$, abstract, once we add a vertex to $P \cup O$ every cycle covered for this vertex it is eliminated of $G(f) - P - O$, so the only reason because a vertex is included in $P \cup O$ is that there exists a cycle that is not covered for another vertex in $P \cup O$, therefore $P \cup O$ is a minimal FVS.

A simpler implementation of force-subroutine

The following algorithm is the way we implement the force-subroutine.
A vertex $v \in V(G)$ is an *ancestor* of a vertex u if $v \in R$ and exists a path $P(v, u)$ such that $\forall w \in P(v, u)$, $w \in R$. The set of ancestors of u is denoted by $\mathrm{Anc}(u)$. A vertex $v \in V(G)$ is a *descendant* of a vertex u if $v \in U \cup Y$ and exists a path $P(v, u)$ such that $\forall w \in P(v, u), w \neq u$ implies $w \in R$. The set of descendants of u is denoted by $\mathrm{Dec}(u)$. The sets $\mathrm{Bef}(u)$ and $\mathrm{Aft}(u)$ contain the elements related to u (the last vertex added to R) that can form a circuit. If there is an arc that goes from a vertex $a \in \mathrm{Aft}(u)$ to a vertex $b \in \mathrm{Bef}(u)$, it means that there is a circuit formed by:

- The path from b to u,
- The path from u to a and
- The arc from a to b.

The sign of this circuit determines how the vertex a is classified (If the circuit is positive, a is added to P. If the circuit is negative, a is added to Y).

The sets $\mathrm{Anc}(u), \mathrm{Dec}(u), \mathrm{Bef}(u)$ and $\mathrm{Aft}(u)$ contain elements of the form (v, σ), where v is a vertex, and σ is a sign ($+$ or $-$). Let x be an element of any of these sets, we denote $v(x)$ to the vertex associated with the element x, and we denote $\sigma(x)$ to the sign associated with the element x. The notation $\overline{\mathrm{Dec}(u)}$ represents all the elements of the $\mathrm{Dec}(u)$ set but with the opposite sign.

Heuristics for the order of the vertices

Different orders of the vertices as input of PFVS algorithm can generate different PFVS as outputs, so it may be necessary to define some heuristics to choose an appropriate order of the vertices. In this chapter we use random order and Min-order defined as follows:

- Min-order: the vertices are ordered according:

Algorithm 4.3: PFVS

Input: The signed regulatory graph $G(f)$ of a BN f with n components and $(v_1, \ldots, v_n)$ an ordered sequence of V.

Output: A PFVS P of $G(f)$, and a FVS F of $G(f)$ such that $P \subseteq F$.

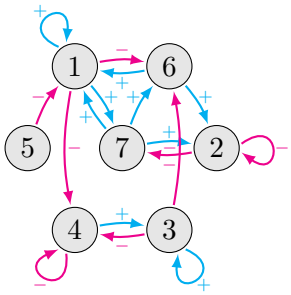
$V^{\oplus} \leftarrow \{u \in V : (u, u)$ is a positive arc of $G(f)\}$;
$V^{\ominus} \leftarrow \{u \in V : (u, u)$ is a negative arc of $G(f)\}$;
$G' \leftarrow G(f) - \{(u, u) \in A : u \in V^{\ominus}\}$;
$P \leftarrow V^{\oplus}$; $O \leftarrow \emptyset$; $R \leftarrow \emptyset$;
phase $\leftarrow 1$;
while $(P \cup R) \neq V$ **do**
    $U \leftarrow V \setminus (P \cup R)$;
    $Y \leftarrow \emptyset$;
    while $U \neq \emptyset$ **do**
        $u \leftarrow$ vertex in U with lowest index;
        $U \leftarrow U \setminus \{u\}$; $R \leftarrow R \cup \{u\}$;
        if (phase $> 1) \vee (u \in V^{\ominus})$ **then** $O \leftarrow O \cup \{u\}$;
        `// Force-subroutine`
        foreach $v \in U \cup Y$ **do**
            $G'' \leftarrow G'[R \cup \{v\}]$;
            if G'' *has a circuit* **then**
                $U \leftarrow U \setminus \{v\}$; $Y \leftarrow Y \setminus \{v\}$;
                if G'' *has a positive circuit* **then** $P \leftarrow P \cup \{v\}$;
                else $Y \leftarrow Y \cup \{v\}$;
            end
        end
        `// end Force-subroutine`
    end
    phase $\leftarrow$ phase $+ 1$;
end
$F \leftarrow P \cup O$;
return P *and* F

Algorithm 4.4: Force-subroutine

```
   // During initialization
 1 foreach v ∈ V(G) do
 2 |  Anc(v) ← ∅; Dec(v) ← ∅;
 3 end
   // Force-subrutine, "u" is the selected vertex in PFVS-Algorithm
 4 Bef(u) ← Anc(u) ∪ {(u, +)};
 5 Aft(u) ← ∅;
 6 temp ← ∅;
 7 foreach v ∈ N⁺(u) do
 8 |  if v ∈ U ∪ Y then Aft(u) ← Aft(u) ∪ {(v, σ(u, v))};
 9 |  if v ∈ R then
10 |  |  if σ(u, v) = + then temp ← Dec(v);
11 |  |  else temp ← Dec(v)‾;
12 |  end
13 end
14 while temp ≠ ∅ do
15 |  foreach t ∈ temp do
16 |  |  temp ← temp \ {t};
17 |  |  if v(t) ∈ U ∪ Y then Aft(u) ← Aft(u) ∪ {t};
18 |  |  if v(t) ∈ R then
19 |  |  |  foreach d ∈ Dec(v(t)) do
20 |  |  |  |  temp ← temp ∪ {(v(d), σ(d) · σ(t))}
21 |  |  |  end
22 |  |  end
23 |  end
24 end
25 foreach (a, b) ∈ Aft(u) × Bef(u) do
26 |  if (v(a), v(b)) ∈ A(G) then
27 |  |  U ← U \ {v(a)};
28 |  |  Y ← Y \ {v(a)};
29 |  |  σ ← σ(a) · σ(b) · σ(v(a), v(b));
30 |  |  if σ = + then P ← P ∪ {v(a)};
31 |  |  else
32 |  |  |  Y ← Y ∪ {v(a)};
33 |  |  |  Dec(b) ← Dec(b) ∪ {(a, σ(a) · σ(b))};
34 |  |  |  Anc(a) ← Anc(a) ∪ {(b, σ(a) · σ(b))};
35 |  |  end
36 |  else
37 |  |  Dec(b) ← Dec(b) ∪ {(a, σ(a) · σ(b))};
38 |  |  Anc(a) ← Anc(a) ∪ {(b, σ(a) · σ(b))};
39 |  end
40 end
```

1. Their degrees, from lowest to highest.
2. If two or more vertices have equal degree then by in-degree, from lowest to highest.
3. If the equality persists, in alphabetical order.

Figure 4.5 shows an example of Min-order.



	1	2	3	4	5	6	7
degree	8	5	5	5	1	5	5
in-degree	4	3	2	3	0	3	2

Min-order: $(5, 7, 3, 6, 4, 2, 1)$

Figure 4.5: A signed regulatory graph $G(f)$ and the order of its vertices according to Min-order.

4.4.4 Random Boolean networks with a given τ and τ^+

In order to evaluate the performance of our algorithms we generate random BNs with a given number of n components, a minimum FVS τ and a minimum PFVS τ^+. These networks have random regulatory graphs constructed from a vertex set with positive loops P with $|P| = \tau^+$, a vertex set with negative loops $F - P$ with $|F - P| = \tau - \tau^+$ and an additional set of $n - \tau$ vertices. The arcs of the digraph, different from the loops, are randomly chosen in such a way their orientations are mostly forward and the backward arcs are those ending at the first vertex of P as shown in Figure 4.6 (a). Thus, the resulting digraph is strongly connected. Furthermore, the signs of the arcs are also randomly chosen. In this way, P is a minimum PFVS and $F := F - P \cup P$ is a minimum FVS of the digraphs. The minimum in-degree and out-degree of the vertices were set to 2 and the maximum values of the in-degrees and out-degrees are unbounded. Next, we replace the loops by cycles of length 2 of the same sign without changing the value of τ and τ^+ of the digraph in order to have different possible PFVS and FVS and keeping the minimum value in-degree and out degree equal to 2 and with maximum values non-bounded as shown in Figure 4.6 (b). The regulatory functions used are of the type strong-inhibition, i.e. they are defined as:

$$f_j(x_1, \ldots, x_n) = \bigwedge_{i:\sigma(i,j)=-1} \bar{x}_i \wedge \bigvee_{i:\sigma(i,j)=1} x_i.$$

In some cases we also used BNs of type AND-NOT (i.e. with regulatory functions like the strong-inhibitions where the $\vee$-connector is replaced by the $\wedge$-connector).

Strong-inhibition Boolean networks and AND-NOT networks have been used in different models of regulatory biological systems [39, 82].

4.5 Results

We first tested the running time of FixedPoint algorithm in random BNs with a given number of n components and known minimum PFVS and FVS. Furthermore, these networks have strongly

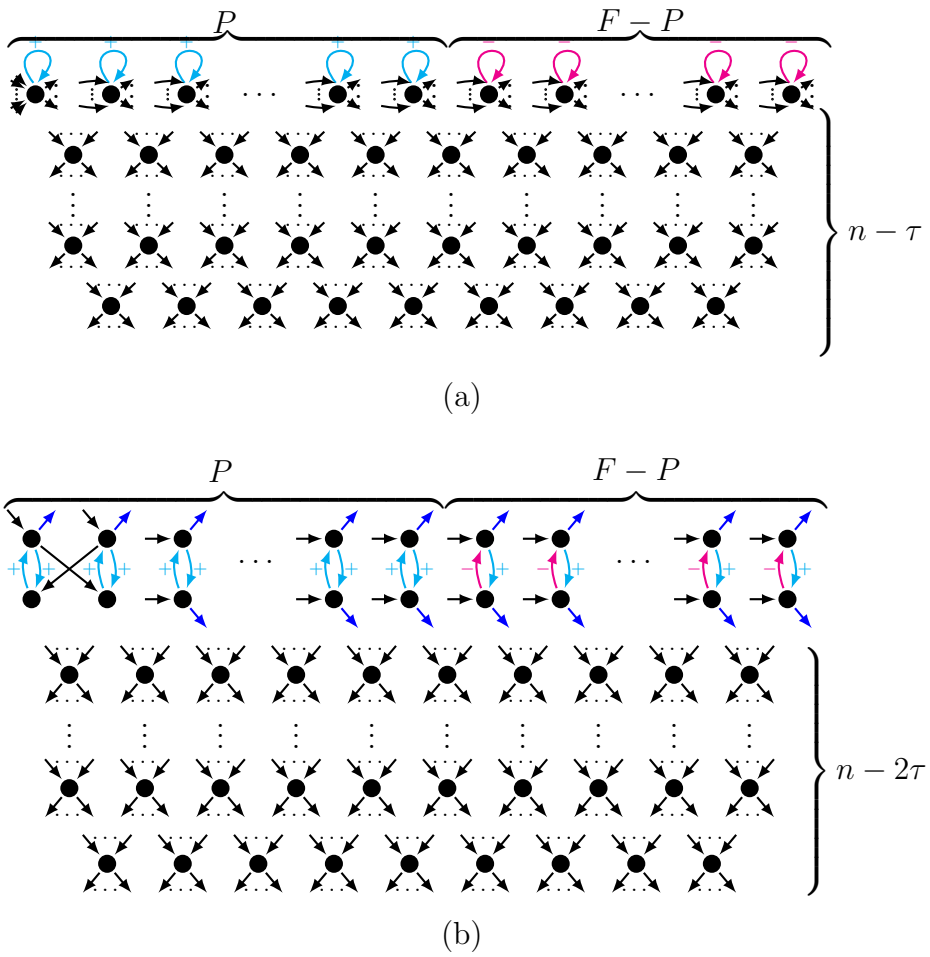


Figure 4.6: Random Boolean networks.

connected regulatory graphs, with minimum in-degree and out-degree greater than or equal to 2 and maximum in-degree and out-degree unbounded.

The average times obtained in seconds with one hundred networks tested in each case, are shown in Figures 4.7 and 4.8. We can see that the time performance of FixedPoint algorithm grows exponentially with the value of τ^+ and polynomially with the size of the network.

The results of PFVS algorithm using different heuristics to chose the order of the vertices are presented in Figure 4.9. We can see that running time of the algorithm depends mainly on the size of the network and not on the sizes of the positive feedback vertex sets. Besides, in the constructed random networks as well as in networks from the literature described in Table 4.1 the performance of the min-order heuristic is better than the random one.

The correctness and time performance of FixedPoint algorithm and PFVS algorithm together were also tested with networks from the literature where the sets of fixed points obtained coincides with those published. The results of these tests were obtained with a PC 3.60GHz Intel Core i7 processor with 16GB of RAM and can be directly checked with the implementation FixedPointsCollector located at `http://www.inf.udec.cl/~lilian/FPCollector/`. We can observe that the running times obtained in these networks are short, probably due to the small size of their positive feedback vertex sets. The latter is also observed in another family of published networks, which are shown in Table 4.2, where the times obtained are compared with those obtained when executing the latest available versions of [81] and [84] algorithms, using the same PC. In

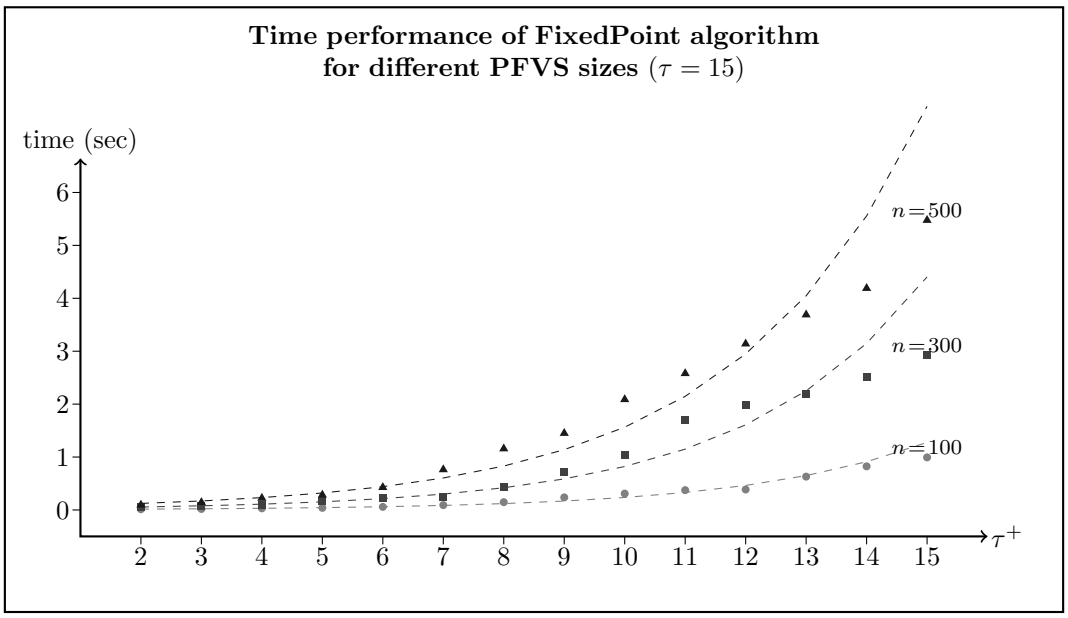


Figure 4.7: Results of FixedPoint algorithm .

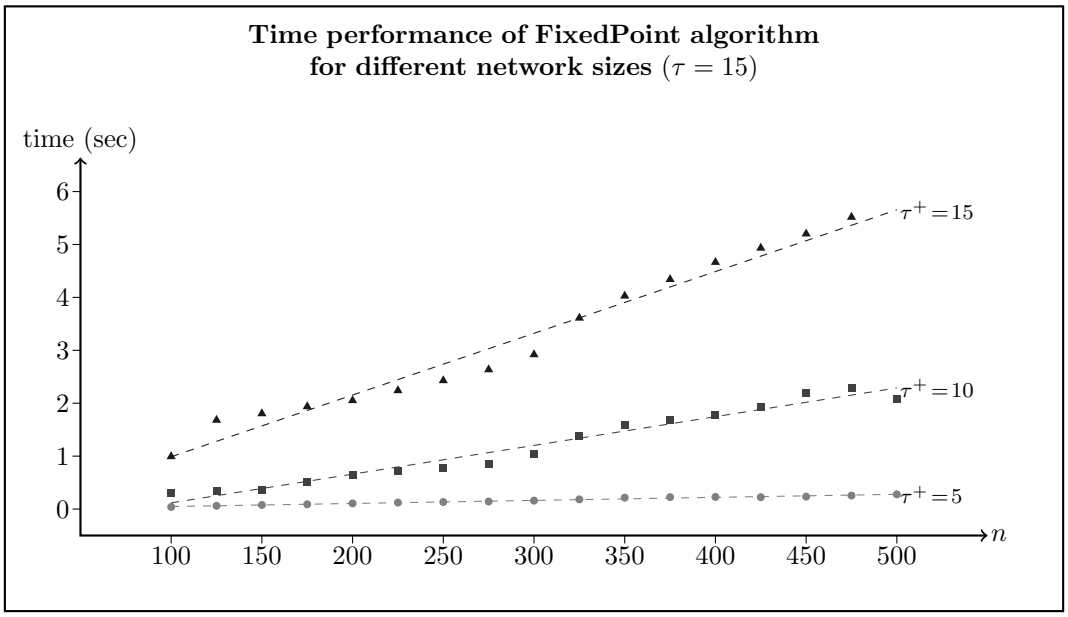


Figure 4.8: Results of FixedPoint algorithm .

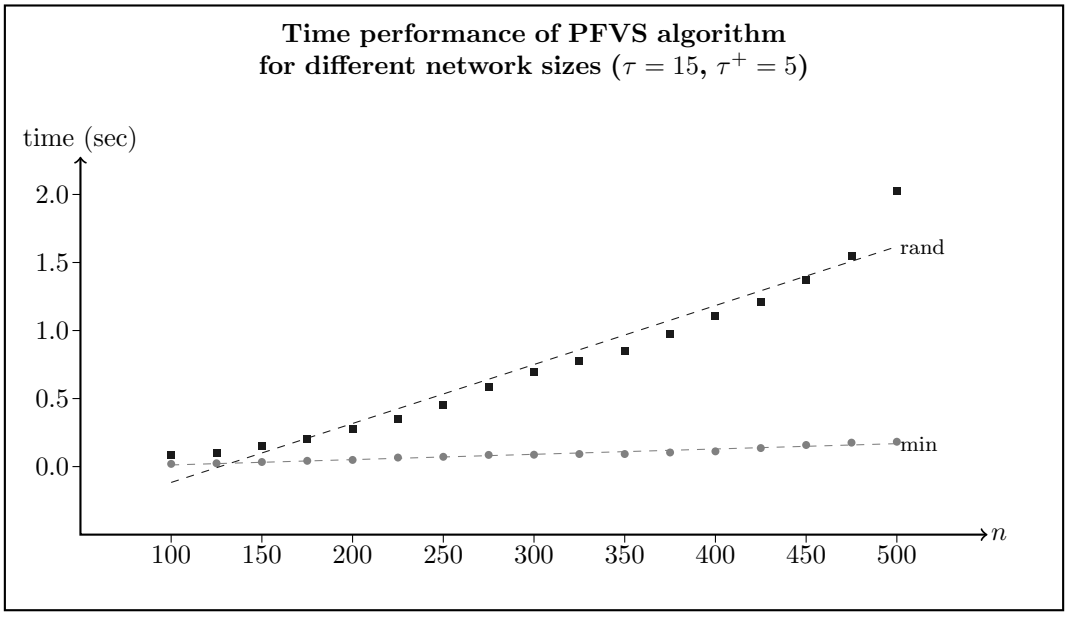


Figure 4.9: Results of PFVS algorithm .

Table 4.1: Performance of FixedPoint algorithm in real networks.

Network	n	**FP**	$\|P\|$	**T(min)**	**T(rand)**
Cancer cell [83]	8	5	4	6.952	8.248
Cancer backbone [83]	8	12	5	7.019	7.321
Budding yeast [50]	11	7	6	5.963	7.036
Fission yeast [25]	10	12	4	8.980	9.416
Arabidopsis Thaliana [74]	13	10	7	8.938	8.919
T-helper cell [54]	23	3	3	7.407	13.512
T-cell receptor [48]	40	1	1	12.782	29.895
HGF [75]	66	2	3	17.286	61.969
Apostosis model 1 [49]	12	2	2	6.923	8.181
Apostosis model 2 [49]	12	4	2	6.870	9.170
Apostosis model 3 [49]	3	4	2	5.143	6.174
Drosophila melanogaster [4]	60	10	9	19.140	64.565
Ventral spinal cord [51]	8	5	4	6.751	7.126

$|P|$ is the size the PFVS obtained with min-order.

T(min) and T(rand) correspond to the time in milliseconds of the PFVS-Alg.+FP-Alg. using Min-order and random-order (in this case, its considered the execution of $n/2$ random permutations to choose the smallest PFVS), respectively.

Table 4.2: Time Performance Comparison.

Network	n	$\|P\|$	Our algor.	Veliz-Cuba	Zañudo & Albert
HGF Signaling in Keratinocytes [75]	68	3	0.451	2.347	1.752
T Cell Receptor Signaling [72]	101	1	0.347	2.355	1.839
Signaling in Macrophage Activation [61]	321	3	0.464	2.406	12.369
Yeast Apostosis [47]	73	3	0.378	2.370	34.059
Influenza A Virus Replication Cycle [52]	131	12	0.407	3.599	107.969
T-LGL Survival network (v. 2011) [71]	60	10	0.580	3.152	18.825
EGFR & ErbB Signaling [73]	104	2	0.356	1.185	>1 hour
Signal Transduction in Fibroblasts [40]	139	39	>3 days	39.122	>1 hour
Random strong-inhibition BNs with minimum in-degree 2	300	15	8.508	4321.026	>6 hours
Random strong-inhibition BNs with minimum in-degree 6	300	15	5.387	>1 hour	>6 hours

$|P|$ is the size the PFVS obtained with min-order. Times presented in seconds according our simulations.

Table 4.2, it can be seen that, for networks with a small τ^+, our times are shorter than those obtained by the Veliz-Cuba and Zañudo & Albert algorithms, however, the same does not occur when τ^+ is large. Also, the difference between Veliz-Cuba and Zañundo & Albert algorithms is coherent with the results shown in [81].

On the other hand, the algorithms FixedPoint algorithm and PFVS algorithm were also tested together to find the fixed point sets in random strong-inhibition BNs and AND-NOT networks with set values $n = 300$, $\tau^+ = 15$ and $\tau = 20$ and different values of k corresponding to its minimum in-degree and minimum out-degree of the its regulatory graphs. The results obtained and displayed in Figure 4.10 show that the running times of FixedPoint algorithm +PFVS algorithm do not increase significantly with the value of k, which means that both algorithms can work well together in BNs with large values (even unbounded) of in-degree and out-degree as long as the found PFVS is small and the regulatory functions of the network can be evaluated in polynomial time. The latter differentiates our algorithm from others whose performance depends strongly on the in-degree or out-degree of the network [81, 85, 84].

4.6 Conclusions and future work

In the modeling of biological systems by Boolean networks usually the regulatory graph of the network is known as well as the type of interaction between their components (activation, inhibition). In this way, it seems natural to use this information to determine the fixed points of a network. In this chapter, we have constructed FixedPoint algorithm to find the set of fixed points of a Boolean network based mainly on the structure of the positive cycles of its regulatory graph and to a lesser extent on the size of the network. This can be considered an improvement to the results obtained by [2], since the set of states to test is smaller and, in the case that $\tau = \tau^+$, our algorithm works the same way. The theoretical foundation of the algorithm is given by Theorem 4.3, which provides a nice characterization of the dynamical behavior of Boolean networks without positive cycles and with a fixed point.

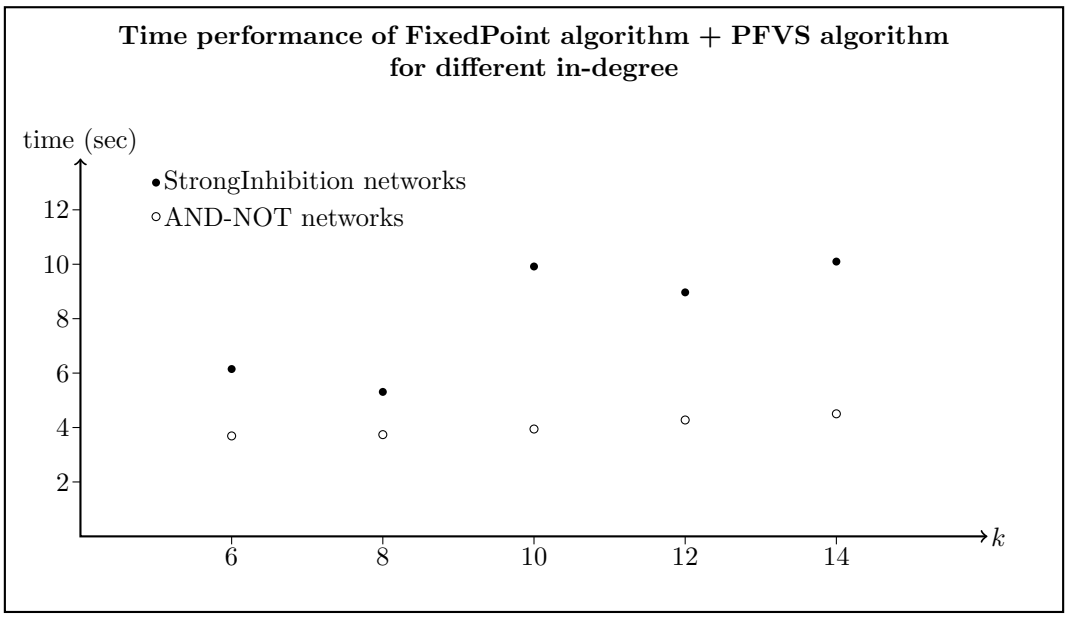


Figure 4.10: Performance of FixedPoint algorithm + PFVS algorithm in random strong-inhibition BNs and AND-NOT networks with different values k of minimum in-degree and out-degree.

On the other hand, our algorithms work well in BNs with large values (even unbounded) of in-degree and out-degree as long as the found PFVS is small and the regulatory functions of the network can be evaluated in polynomial time. This is an advantage over others algorithms whose performance depends strongly on the in-degree or out-degree of the network.

The efficiency of FixedPoint algorithm depends mainly on the size of the input PFVS. Due to this, it is important to have a PFVS as close to the minimum as possible. In this sense, it is clear that PFVS algorithm can be improved. One improvement could be done implementing an efficient algorithm to decide the existence of a positive cycle in a signed digraph. This latter problem, which is equivalent to the existence of an even cycle in a digraph [55], is a surprisingly difficult problem and whose algorithmic complexity was unknown for a long time. Although Robertson, Seymour and Thomas finally proved that this decision problem can be solved in polynomial time [69], the implementation of such algorithm is not easy to do which makes it one of our future challenges. Another way to improve PFVS algorithm is the possibility of building a fixed-parameter tractable algorithm to determine a PFVS of minimum size, i.e. an algorithm whose complexity depends mainly on this size and not on the size of the regulatory graph. It is known that this type of algorithm exists to solve the minimum FVS problem in digraphs [23]. However, it is an open problem, at our knowledge, in the case of minimum PFVS in signed digraphs.

In future work it would be important to explore methods to reduce the size of a network that conserve τ^+. Also, since the efficiency of the FixedPoint algorithm depends on the size of a input PFVS, it is important to study the relationship between τ^+ and other parameters of the regulatory graph of a Boolean network such as: minimum and maximum in-degree, out-degree and degree distributions.

Chapter 5

Dynamically Equivalent Network problem

5.1 Introduction

Parallel digraphs, which are preliminary presented by F. Robert [67, 68], calling them Gauss-Seidel operator, are a widely used tool over the years. Thanks to [36, 12], equivalence classes have been defined between different update schedules based on their update digraph, so that elements in the same class have the same dynamic behavior.

In this sense, in [21] the dynamics of discrete neural networks with deterministic update schedules is studied and in [10] it is studied how many different dynamics can exist in a Boolean network when the update schedule changes. In the particular case of disjunctive networks, in [8] it is studied the complexity of deciding whether there exists a limit cycle of a given length k for some update schedule, and in [34] it is classified them according to the robustness of their dynamics concerning changes in the update schedule, all this using parallel digraphs.

However, to our knowledge, the following questions have been little explored: What other networks have the same dynamics as that of a given network? What dynamics are only yielded by a parallel schedule? Research closer to this one is [37], with the difference being that while authors go in one direction, our research goes in the opposite direction, i.e., the authors in [37] study among other things how the network changes when it is updated with a given sequential schedule, while in this chapter we are interested in studying whether it is possible to obtain a given network from some other network updated with some block-sequential schedule.

This chapter addresses the above questions. For that, our approach as follows: in Section 5.2 we define the notations that are used. Then, in Section 5.3, we formally define the problem and prove that it is NP-hard in the general case. In Section 5.4, we restrict our problem to disjunctive Boolean networks. Later, in Section 5.5, we present an algorithm that decides the problem defined in Section 5.4 in polynomial time for disjunctive (conjunctive) networks. Finally, in the last section, the conclusions reached are presented.

5.2 Definition and notation

Definition 5.1. Given $f : \mathbb{B}^n \to \mathbb{B}^n$ a Boolean network, f is called a disjunctive Boolean network if: $\forall u \in [n],\ f_u(x) = 1 \iff (\exists v \in N_f^-(u), x_v = 1)$.

We observe that the global transition function of a disjunctive Boolean network is completely described by its interaction graph.

Definition 5.2. Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network, $x^t = (x_1^t, \ldots, x_n^t) \in \mathbb{B}^n$ a state configuration and $s = B_1, B_2, \ldots, B_m$ a block-sequential update schedule. The dynamical behavior of f *updated according to* s is given by:

$$\forall v \in B_1, \qquad x_v^{t+1} = f_v(x^t). \tag{5.1}$$

$$\forall v \notin B_1, \qquad x_v^{t+1} = f_v(x_u^{t+1} : s(u) < s(v); x_u^t : s(u) \geq s(v)) \tag{5.2}$$

The expression in (5.1) is because when updating the elements in B_1, no other elements have been updated. The expression in (5.2) is because at the time of updating x_v, if its dependency (x_u) belongs to a previous block it has already been updated (x_u^{t+1}) and if its dependency belongs to a later block (or the same block) it takes its value without updating (x_u^t).

This definition is an interpretation of what was introduced by F. Robert in [67], where the origin of this expression is explained in depth.

This is equivalent to applying a function f^s to x:

$$x^{t+1} = f^s(x^t),$$

where f^s is defined by:

$$\forall v \in B_1, \qquad f_v^s(x) = f_v(x). \tag{5.3}$$

$$\forall v \notin B_1, \qquad f_v^s(x) = f_v(f_u^s(x) : s(u) < s(v); x_u : s(u) \geq s(v)) \tag{5.4}$$

It is easy to prove that f^s is equivalent to:

$$f^s = f^{B_m} \circ f^{B_{m-1}} \circ \cdots \circ f^{B_2} \circ f^{B_1}$$

with $f^{B_i} : \mathbb{B}^n \to \mathbb{B}^n$ given by:

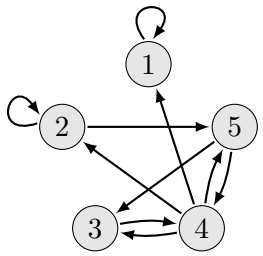
$$\forall x \in \mathbb{B}^n, \quad f_v^{B_i}(x) = \begin{cases} x_v & \text{if } v \notin B_i \\ f_v(x) & \text{if } v \in B_i. \end{cases}$$

Note that, under this definition, $f = f^{s_p}$. In [12] was proved that if $s \sim_{G(f)} s'$, then $f^s = f^{s'}$. Moreover, if f is a disjunctive Boolean network, then, by definition, f^s is also a disjunctive Boolean network. This is because the family of disjunctive Boolean functions is closed under composition.

In this way, the dynamical behavior of f updated according to s is equivalent to the dynamical behavior of f^s updated according to the parallel schedule.

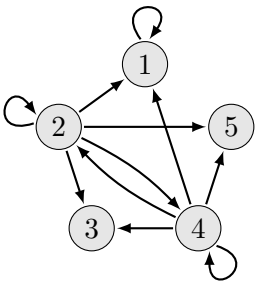
An example of a Boolean network f updated according to a block-sequential update schedule s is shown in Figure 5.1(a) and Figure 5.1(c).

Figure 5.1 shows that, obtaining $G(f^s)$, that is, the interaction graph that presents the actual dependencies of the local functions of f updated according to s, is not a simple task. In fact, it was proved to be a DP-complete problem [60]. For example, f_5^s is a constant function 0, when the function f_5 depends on x_2 and x_4. Obtaining $G(f^s)$ depends on the local functions of f and how they interact with each other. For this reason, to study this, we use the *parallel digraph*. This digraph represents the potential effective dependencies of a network if it were to be updated in parallel.



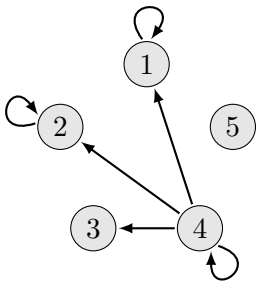
$f_1(x) = x_1 \vee x_4$
$f_2(x) = x_2 \wedge x_4$
$f_3(x) = x_4 \wedge \neg x_5$
$f_4(x) = x_3 \wedge \neg x_5$
$f_5(x) = x_2 \wedge \neg x_4$
$s = \{2\}\,\{5\}\,\{3\}\,\{4\}\,\{1\}$

(a)



$f_1^s(x) = x_1 \vee (x_4 \wedge \neg(x_2 \wedge x_4 \wedge \neg x_4))$
$f_2^s(x) = x_2 \wedge x_4$
$f_3^s(x) = x_4 \wedge \neg(x_2 \wedge x_4 \wedge \neg x_4)$
$f_4^s(x) = x_4 \wedge \neg(x_2 \wedge x_4 \wedge \neg x_4)$
$f_5^s(x) = x_2 \wedge x_4 \wedge \neg x_4$

(b)



$f_1^s(x) = x_1 \vee x_4$
$f_2^s(x) = x_2 \wedge x_4$
$f_3^s(x) = x_4 \wedge \neg(0) = x_4$
$f_4^s(x) = x_4 \wedge \neg(0) = x_4$
$f_5^s(x) = (x_2 \wedge x_4) \wedge \neg x_4 = 0$

(c)

Figure 5.1: (a) A Boolean network f and an update schedule s (b) The parallel digraph $G_P(f, s)$ (c) The effective network f^s.

5.3 Dynamically equivalent networks problem

In this chapter we focus on the inverse problem, i.e. given a dynamical behavior of a Boolean network, we want to know if there exists a Boolean network with an update schedule different from the parallel that produces the same dynamical behavior.

Definition 5.3. Let $f, h : \mathbb{B}^n \to \mathbb{B}^n$ be two Boolean networks and s an update schedule. We say that (h, s) is *dynamically equivalent to* f if $h^s = f$. Moreover, if $h \neq f$, or $h = f$ and $s \nsim_{G(f)} s_p$, we say that (h, s) and f are *non-trivially dynamically equivalent*.

By Equation (2.1), remember that if $h = f$, for every s equivalent to s_p, we have $h^s = f$. And there exists $s \neq s_p$ equivalent to s_p if and only if $G(f)$ is not strongly connected. Indeed, if $G(f)$ is not strongly connected, then there is at least one source (initial) strongly connected component. Then, the two-block schedule s wherein the second block are the vertices of the source component and in the first block, the rest of the vertices, is equivalent to s_p. On the other hand, if $G(f)$ is strongly connected and s is equivalent to s_p, then between any pair of vertices (u, v), there exists a fully positive path from u to v, so by transitivity, $s(u) \geq s(v)$. Therefore, for any pair of vertices u, v, $s(u) = s(v)$, and thus s is the parallel schedule.

Example 5.1. Let $f : \mathbb{B}^2 \to \mathbb{B}^2$ be the Boolean network defined by $f(x_1, x_2) = (x_2, x_1)$ (see Figure 5.2(a)), let us prove that it does not exist another network non-trivially dynamically equivalent to f.

Note that the only update schedules that are not equivalent to s_p are $s = \{1\}\,\{2\}$ and $s' = \{2\}\,\{1\}$.

Let us suppose that there exists a Boolean network h such that $h^s = f$, where $s = \{1\}\,\{2\}$. And let $x \in \{0, 1\}^2$ be such that $h_2(x_2, x_2) \neq x_1$. Then,

$$h^s(x_1, x_2) = (h_1^s(x_1, x_2), h_2^s(x_1, x_2)).$$

Since $1 \in B_1$,

$$h_1^s(x_1, x_2) = h_1(x_1, x_2) = f_1(x_1, x_2) = x_2.$$

Thus,

$$h_2^s(x_1, x_2) = h_2(h_1^s(x_1, x_2), x_2) = h_2(x_2, x_2) \neq x_1 = f_2(x_1, x_2).$$

Therefore, there is no network h with update schedule s that are non-trivially dynamically equivalent to f.

Analogously, it is possible to show that there is no network h non-trivially dynamically equivalent to f when $s' = \{2\}\,\{1\}$.

Example 5.2. On the other hand, if we consider the function $f'(x) = (f'_1(x), f'_2(x)) = (x_1, x_1)$ (see Figure 5.2(b)) which only differ with f in the first local activation function, and $s = \{2\}\,\{1\}$, the Boolean network $h'(x) = (h'_1(x), h'_2(x)) = (x_1 \vee x_2, x_1)$ satisfies $h'^s = f'$.

Figure 5.2: Interaction digraph of the Boolean network from Example 5.1 and Example 5.2.

Using the above definition, we introduce the following problem:

> Dynamically equivalent networks problem (DEN problem)
>
> **Input:** A Boolean network f (encoded as a Boolean formula for each f_i).
>
> **Question:** does there exists a Boolean network h and an update schedule s, such that (h, s) is non-trivially dynamically equivalent to f?

The universe of possible solutions is very large: for n components, there are $O(2^{n^{2^n}})$ possible Boolean networks and $O(n!)$ possible update schedules [26]. The following result shows a relation between different solutions:

Theorem 5.1. *If there exists a solution to DEN problem then there exists a solution to DEN problem with a block-sequential update schedule with two blocks.*

To prove the previous theorem, we use the following lemma:

Lemma 5.2. *Let $h, f : \mathbb{B}^n \to \mathbb{B}^n$ be Boolean networks and $s = B_1, B_2, \ldots, B_m$ with $m > 1$. If $h^s = f$, then there exists $\bar{h}$ and $\bar{s}$ such that $\bar{h}^{\bar{s}} = f$ where $\bar{s} = B_1, B_2, \ldots, B_{m-1} \cup B_m$.*

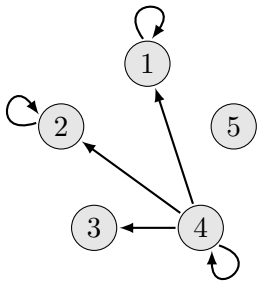
Proof. Without loss of generality, let us suppose that there exists $u \in B_{m-1}$ and $v \in B_m$ such that $(u, v) \in A(h)$. If this condition does not hold, $\bar{s} \sim_{G(\bar{h})} s$. We define $\bar{h}$ as follows:

$$\bar{h}_v(x) = \begin{cases} h_v(x) & \forall v \notin B_m, \\ h_v(f^{B_{m-1}}(x)) & \forall v \in B_m. \end{cases}$$

Finally:

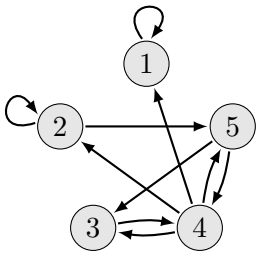
$$\forall v \in B_j \wedge j < m, \quad \bar{h}_v^{\bar{s}}(x) = \bar{h}_v(\bar{h}^{B'_{j-1}} \circ \bar{h}^{B'_{j-2}} \circ \cdots \circ \bar{h}^{B'_1}(x)). \tag{5.5}$$
$$= h_v(h^{B_{j-1}} \circ h^{B_{j-2}} \circ \cdots \circ h^{B_1}(x)). \tag{5.6}$$
$$= h_v^s(x) = f_v(x). \tag{5.7}$$
$$\forall v \in B_m, \quad \bar{h}_v^{\bar{s}}(x) = \bar{h}_v(h^{B'_{m-2}} \circ h^{B'_{m-3}} \circ \cdots \circ h^{B'_1}(x)). \tag{5.8}$$
$$= \bar{h}_v(h^{B_{m-2}} \circ h^{B_{m-3}} \circ \cdots \circ h^{B_1}(x)). \tag{5.9}$$
$$= h_v(h^{B_{m-1}}(h^{B_{m-2}} \circ h^{B_{m-3}} \circ \cdots \circ h^{B_1}(x))). \tag{5.10}$$
$$= h_v(h^{B_{m-1}} \circ h^{B_{m-2}} \circ h^{B_{m-3}} \circ \cdots \circ h^{B_1}(x)). \tag{5.11}$$
$$= h_v^s(x) = f_v(x). \tag{5.12}$$

For this reason, $\bar{h}^{\bar{s}} = f$. □



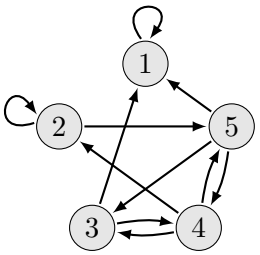
$f_1(x) = x_1 \vee x_4$
$f_2(x) = x_2 \wedge x_4$
$f_3(x) = x_4$
$f_4(x) = x_4$
$f_5(x) = 0$

(a)



$h_1(x) = x_1 \vee x_4$
$h_2(x) = x_2 \wedge x_4$
$h_3(x) = x_4 \wedge \neg x_5$
$h_4(x) = x_3 \wedge \neg x_5$
$h_5(x) = x_2 \wedge \neg x_4$
$s = \{2\}\,\{5\}\,\{3\}\,\{4\}\,\{1\}$

(b)



$\bar{h}_1(x) = x_1 \vee (x_3 \wedge \neg x_5)$
$\bar{h}_2(x) = x_2 \wedge x_4$
$\bar{h}_3(x) = x_4 \wedge \neg x_5$
$\bar{h}_4(x) = x_3 \wedge \neg x_5$
$\bar{h}_5(x) = x_2 \wedge \neg x_4$
$\bar{s} = \{2\}\,\{5\}\,\{3\}\,\{4, 1\}$

(c)

Figure 5.3: (a) A Boolean network f (b) A solution with 5 blocks (c) A solution with 4 blocks.

Notice that, $\bar{s}$ is non-equivalent to s if and only if there exists an arc from some vertex in B_{m-1} to a vertex in B_m.

Proof of Theorem 5.1. If there exists a solution with an update schedule with $k > 2$ blocks, applying the Lemma 5.2 successively, it is possible to construct a solution with 2 blocks. □

To understand the real complexity of the problem, let us see the following theorem:

Theorem 5.3. *DEN is NP-Hard.*

Proof. To prove NP-Hardness we show that 3-SAT $\leq_p$ DEN. Let ϕ be a 3-CNF formula in variables $x_1, \ldots, x_n$. Without loss of generality, let us consider that ϕ has only non-trivial clauses; a non-trivial clause C_i being a clause such that for each variable $x_j \in C_i$, we have $\bar{x_j} \notin C_i$. Note

that eliminating trivial clauses from ϕ is a simple task. Now, we consider $f : \mathbb{B}^{n+2} \to \mathbb{B}^{n+2}$ as follows:

$$\begin{aligned} \forall u \in [n], \qquad f_u(x) &= x_u, \\ f_{n+1}(x) &= \phi(x_1, \ldots, x_n) \vee x_{n+2} \\ f_{n+2}(x) &= x_{n+1}. \end{aligned}$$

See $G(f)$ in Figure 5.4(a).

($\Rightarrow$) If ϕ is satisfiable, we consider the function $\bar{f} = f$ and the update schedule $s = \{1, \ldots, n\}\{n+1, n+2\}$. Since ϕ is satisfiable and there exists $x \in \mathbb{B}^n$ such that $\phi(x) = 0$ (because ϕ has only non-trivial clauses), f_{n+1} depends on x_u for some $u \in [n]$, so s is not equivalent to the parallel update schedule, and $\bar{f}^s = f$.

($\Leftarrow$) If ϕ is not satisfiable $\forall u \in [n], f_u(x) = x_u$, $f_{n+1}(x) = x_{n+2}$ and $f_{n+2}(x) = x_{n+1}$. See $G(f)$ in Figure 5.4(b). We see that the sub-graph induced by vertices $n+1$ and $n+2$ is isomorphic to the digraph presented in Figure 5.2(a). And, as in that example, it is shown that there is no Boolean network that is non-trivially dynamically equivalent to the disjunctive Boolean network with this interaction sub-graph. Then there is also no Boolean network that is non-trivially dynamically equivalent to f. In this way, any update schedule that preserves the dynamical behavior is equivalent to the parallel. □

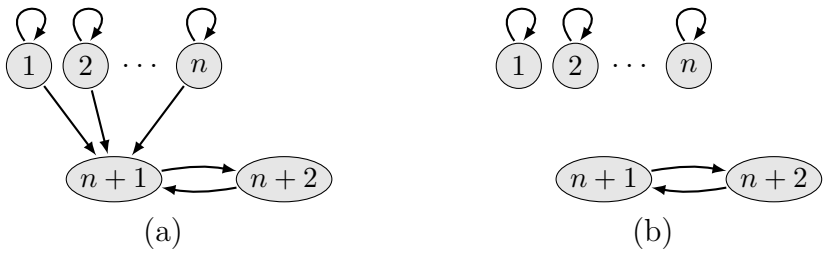


Figure 5.4: Interaction digraph of the transformation defined in Theorem 5.3.

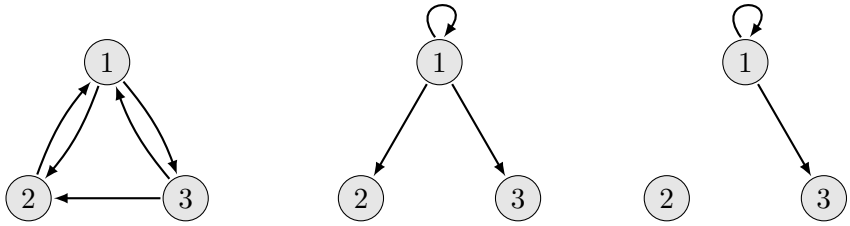
5.4 Dynamically equivalent disjunctive networks problem

As we can see, in the general case, the DEN problem is hard, therefore, we focus on disjunctive networks, defining the following problem:

> Dynamically equivalent disjunctive networks problem (D-DEN problem)
>
> **Input:** A disjunctive Boolean network f (encoded as a Boolean formula for each f_i).
>
> **Question:** does there exists a disjunctive Boolean network h and an update schedule s, such that (h, s) is non-trivially dynamically equivalent to f?

Why only restrict ourselves to disjunctive Boolean networks h?

As can be seen in Figure 5.5, there are non-disjunctive networks that can generate disjunctive networks. But in this case, the equality between the parallel digraph and the effective digraph produced by the composition of functions, as analyzed in Remark 2.1, is lost.



$h_1(x) = x_2 \veebar x_3$
$h_2(x) = x_1 \veebar x_3$
$h_3(x) = x_1$

(a)

$h_1^s(x) = (x_1 \veebar x_1) \veebar x_1$
$h_2^s(x) = x_1 \veebar x_1$
$h_3^s(x) = x_1$
$s = \{3\}\,\{2\}\,\{1\}$

(b)

$h_1^s(x) = x_1$
$h_2^s(x) = 0$
$h_3^s(x) = x_1$

(c)

Figure 5.5: (a) A non-disjunctive Boolean network h (b) The parallel digraph $G_P(h, s)$ (a disjunctive network) (c) The effective network f^s (a disjunctive network).

Remark 5.1. *Based on the last part of Definition 2.4, it is possible to define the parallel digraph* $G_P(f, s) = ([n], A)$ *in terms of the labeled digraph as:*

$$A = \{(u, v) \in A(f) : \mathrm{lab}_s(u, v) = \oplus\} \cup \{(u, v) : \exists w \in N_f^-(v), (u, w) \in A \wedge \mathrm{lab}_s(w, v) = \ominus\}.$$

where the set $\{(u, v) : \exists w \in N_f^-(v), (u, w) \in A \wedge \mathrm{lab}_s(w, v) = \ominus\}$ *represents the arcs generated for the predecessors that were already updated in a previous block.*

An example of parallel digraph is shown in Figure 5.1(b).

In the same way, given a labeled digraph (G, lab) without fully negative cycles, we can define $G_P(G, \mathrm{lab}) = ([n], A)$ as:

$$\begin{aligned} A = &\{(u, v) \in A(G) : \mathrm{lab}(u, v) = \oplus\} \cup \\ &\{(u, v) \in [n] \times [n] : \exists w \in [n], (w, v) \in A(G) \wedge (u, w) \in A \wedge \mathrm{lab}(w, v) = \ominus\}. \end{aligned}$$

In this definition, if f is a disjunctive Boolean network and s is an update schedule, then $G_P(G(f), \mathrm{lab}_s) = G_P(f, s)$.

The following results are consequences of the definition of *parallel digraph*:

Lemma 5.4. *Let* $h, f : \mathbb{B}^n \to \mathbb{B}^n$ *be two disjunctive Boolean networks and* s *an update schedule such that* $h^s = f$*. Then, for* $u, v \in [n]$*:*

$$[(u, v) \in A(h) \wedge \mathrm{lab}_s(u, v) = \ominus] \Longrightarrow N_f^-(u) \subseteq N_f^-(v).$$

Proof. By Remark 2.1, we have that $h^s = f$ is equivalent to $G_P(h, s) = G(f)$. By contradiction, let us suppose that there exists $(u, v) \in A(h)$ such that $\mathrm{lab}_s(u, v) = \ominus$ and $N_f^-(u) \nsubseteq N_f^-(v)$. Since $N_f^-(u) \setminus N_f^-(v) \neq \emptyset$, let $w \in N_f^-(u) \setminus N_f^-(v)$, then, $(w, u) \in A(f)$ and $(u, v) \in A(h)$ with $\mathrm{lab}_s(u, v) = \ominus$, by definition of parallel digraph, there exists a vertex $u \in N_h^-(v)$, such that $(w, u) \in A(f)$ and $\mathrm{lab}_s(u, v) = \ominus$, therefore $(w, v) \in A(f)$, which is a contradiction because $w \notin N_f^-(v)$. □

Remark 5.2. *Based on Lemma 5.4, for all those disjunctive Boolean networks whose vertex neighborhoods are not comparable there is no network that is non-trivially dynamically equivalent. Some examples are the disjunctive Boolean networks $f : \mathbb{B}^n \to \mathbb{B}^n$ such that:*

- *Complete digraphs without loops, for $n \geq 2$, where:*

$$\forall v \in [n], \quad N_f^-(v) = [n] \setminus \{v\}$$

- *The double chain digraph with loops in the extreme vertices, for $n \geq 3$, where:*

$$\forall v \in [n], \quad N_f^-(v) = \begin{cases} \{1, 2\} & \text{if } v = 1 \\ \{n-1, n\} & \text{if } v = n \\ \{v-1, v+1\} & \text{otherwise} \end{cases}$$

- *The double cycle, for $n \geq 2$ (with exception of $n = 4$), where:*

$$\forall v \in [n], \quad N_f^-(v) = \begin{cases} \{n, 2\} & \text{if } v = 1 \\ \{n-1, 1\} & \text{if } v = n \\ \{v-1, v+1\} & \text{otherwise} \end{cases}$$

Note that the condition of Lemma 5.4 is necessary but not sufficient, as shown in the following example.

Example 5.3. The Figure 5.6 shows that given a disjunctive Boolean network f, if there are vertices $u, v \in [n]$ such that $N_f^-(u) \subseteq N_f^-(v)$, then a network non-trivially dynamically equivalent network does not necessarily exist. This Boolean network is known because the only equivalent dynamic network is the trivial one, since according to the schedule $\{1\}\{2\}$ there is no way to build the arc $(1, 2)$, and according to the schedule $\{2\}\{1\}$ there is no way to build the arc $(2, 1)$, (the loop in 1 cannot be included for any h, since it is not in $A(f)$). Note also that it is true that $N_f^-(1) = \{2\} \subseteq \{1, 2\} = N_f^-(2)$.

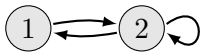


Figure 5.6: $N_f^-(1) \subseteq N_f^-(2)$ but the only network dynamically equivalent is the trivial one.

However, if we consider the case of equal neighborhoods, we obtain a sufficient condition.

Proposition 5.5. *Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a disjunctive Boolean network. There exists a disjunctive Boolean network h and an update schedule s, such that (h, s) is non-trivially dynamically equivalent to f and with only one negative arc $(u, v) \in A(h)$ if and only if the following conditions are satisfied:*

1. *$N_f^-(u) \subseteq N_f^-(v)$,*
2. *$u \notin N_f^-(v) \setminus N_f^-(u)$,*
3. *For every vertex $w \in N_f^-(v) \setminus N_f^-(u)$, it does not exist a path from u to w in $G(f) - v$.*

Proof. Since f and h are disjunctive Boolean networks, we have that $h^s = f$ is equivalent to $G_P(h, s) = G(f)$.

($\Rightarrow$ 1.) This is true according to Lemma 5.4.

($\Rightarrow$ 2.) Let us suppose that there exists a Boolean network h and an update schedule s, such that (h, s) is non-trivially dynamically equivalent to f with only one negative arc $(u, v) \in A(h)$ and $u \in N_f^-(v) \setminus N_f^-(u)$. Then $(u, v) \in A(f)$ (because $u \in N_f^-(v)$) and $(u, u) \notin A(f)$ (because $u \notin N_f^-(u)$). Since $(u, u) \notin A(f)$ the only way to create (u, v) in f is that there exists a vertex $w \in [n]$ such that $(u, w) \in A(f)$, $(w, v) \in A(h)$ and $\text{lab}_s(w, v) = \ominus$, but since (u, v) is the only negative arc of $G(h)$, this path does not exist, therefore $(u, v) \notin A(f)$, which is a contradiction.

($\Rightarrow$ 3.) Let us suppose that there exists a Boolean network h and an update schedule s, such that (h, s) is non-trivially dynamically equivalent to f and with only one negative arc $(u, v) \in A(h)$, in this case $s(u) < s(v)$. Also, let us suppose, there exists a vertex $w \in N_f^-(v) \setminus N_f^-(u)$, such that there exists a path from u to w in $G(f) - v$. Since (u, v) is the only negative arc, all arcs in the path from u to w in $G(f) - v$ are in $A(h)$ and their labels are $\oplus$, the same occurs with the arc (w, v) (because $w \in N_f^-(v)$), so $s(u) \geq s(w) \geq s(v)$. Therefore, $s(u) < s(v)$ and $s(u) \geq s(v)$ which is a contradiction.

($\Leftarrow$) Let u^* and v^* be two vertices in $[n]$ such that $N_f^-(u^*) \subseteq N_f^-(v^*)$ and for every vertex $w \in N_f^-(v^*) \setminus N_f^-(u^*), w \neq u^*$ and it does not exists a path from u^* to w in $G(f) - v^*$. We define the Boolean network $h : \mathbb{B}^n \to \mathbb{B}^n$ such that:

$$A(h) = \left(A(f) \setminus \left\{(w, v^*) : w \in N_f^-(u^*)\right\}\right) \cup \left\{(u^*, v^*)\right\}.$$

Notice that:

$$N_h^-(v) = \begin{cases} N_f^-(v) & \text{if } v \neq v^* \\ \{u^*\} \cup N_f^-(v^*) \setminus N_f^-(u^*) & \text{if } v = v^* \end{cases}$$

Let $s = B_1 B_2$ where:

$$B_2 = \{v^*\} \cup \{w \in V(h) : \text{there exists a path from } w \text{ to } v^* \text{ in } G(h) - (u^*, v^*)\}$$

Note that, $B_1 = V(h) \setminus B_2$ is not empty, since condition 3 ensures that the only path from u^* to v^* is the arc (u^*, v^*), so $u^* \in B_1$. Also, the only arc from a vertex in B_1 to a vertex in B_2 is (u^*, v^*), because if there exists a vertex $u \in B_1$ such that u is in the in-neighborhood of a vertex $v \in B_2$ ($v \neq v^*$), then there is a path from u to v^* and, therefore $u \in B_2$ which is a contradiction. In this way, for all $v \in V(h)$ if $v \neq v^*$, $(u, v) \in A(h^s)$ is equivalent to $(u, v) \in A(h)$, therefore $(u, v) \in A(h^s)$ if and only if $(u, v) \in A(f)$. Now, $(u, v^*) \in A(h^s)$ is equivalent to: $u \in N_f^-(v^*) \setminus N_f^-(u^*)$ ($\text{lab}_s(u, v) = \oplus$) or $u \in N_{h^s}^-(u^*) = N_f^-(u^*)$, since $\text{lab}_s(u^*, v^*) = \ominus$. In this way, $(u, v^*) \in A(h^s)$ if and only if $(u, v^*) \in A(f)$.

Finally, it has been proven that there exists a disjunctive Boolean network $h \neq f$ and $s \nsim_{G(h)} s_p$ such that $G_P(h, s) = G(f)$. An example of f, h and s can be seen at Proposition 5.5. □

Corollary 5.6. *Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a disjunctive Boolean network. If there exist $u, v \in [n]$ such that $N_f^-(u) = N_f^-(v)$, then there exists a disjunctive Boolean network h and an update schedule s such that (h, s) is non-trivially dynamically equivalent to f.*

Proof. If $N_f^-(u^*) = N_f^-(v^*)$ then the conditions of Proposition 5.5 are satisfied. □

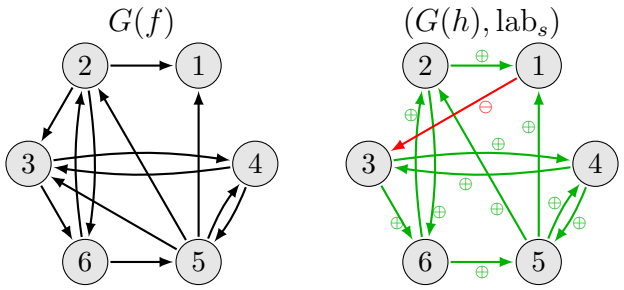


Figure 5.7: An example of h, f and $(u^*, v^*) = (1, 3)$. Note that $N_f^-(1) = \{2, 5\} \subseteq \{2, 4, 5\} = N_f^-(3)$ and it does not exists a path from 4 to 1 in $G(f) - 3$.

5.5 Algorithm to decide D-DEN Problem

To design a strategy to recognize all the vertices that meet the necessary condition given by Lemma 5.4, we introduce the following definitions:

Definition 5.4. Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a disjunctive Boolean network. We define the following set of arcs:

$$A^{\ominus}(f) = \{(u, v) \in [n] \times [n] : u \neq v \wedge N_f^-(u) \subseteq N_f^-(v)\}.$$

This set represents all arcs in $[n]\times[n]$ that can be labeled $\ominus$. Note that the inclusion relationship of these sets is transitive, because if $N_f^-(u) \subseteq N_f^-(w)$ and $N_f^-(w) \subseteq N_f^-(v)$, then $N_f^-(u) \subseteq N_f^-(v)$.

Remark 5.3. *Note that in terms of $A^{\ominus}(f)$ we can reinterpret the previous lemmas as follows:*

1. *By Lemma 5.4, if $A^{\ominus}(f) = \emptyset$, then there is no network (h, s) that is non-trivially dynamically equivalent to f.*
2. *By Corollary 5.6, if $A^{\ominus}(f)$ induces a digraph with at least one cycle, then there exist at least two vertices with equal in-neighborhoods, for this reason, there exists a network non-trivially dynamically equivalent to f.*
3. *By Proposition 5.5, if $|A^{\ominus}(f)| = 1$, there exists a network non-trivially dynamically equivalent to f if and only if for all $w \in N_f^-(v) \setminus N_f^-(u)$, $w \neq u$ and it does not exist a path from u to w in $G(f) - v$.*

Definition 5.5. Given a partially labeled digraph (G, lab) we denoted the sets of arcs $\text{lab}^{\oplus}[G, \text{lab}]$ and $\text{lab}^{\ominus}[G, \text{lab}]$ as follows:

$$\text{lab}^{\oplus}[G, \text{lab}] = \{a \in A(G) : \text{lab}(a) = \oplus\}$$
$$\text{lab}^{\ominus}[G, \text{lab}] = \{a \in A(G) : \text{lab}(a) = \ominus\}$$

Definition 5.6. Given $n \in \mathbb{N}$ and two sets of arcs $A^-, A^+ \subseteq [n] \times [n]$, such that $A^- \cap A^+ = \emptyset$, we define (G, lab) the *labeled digraph induced by* $[n]$, A^- *and* A^+, denoted by $G[A^-, A^+]$, as follows:

- $V(G) = [n]$
- $A(G) = A^- \cup A^+$
- $\forall a \in A(G), \text{lab}(a) = \begin{cases} \ominus & \text{if } a \in A^- \\ \oplus & \text{if } a \in A^+ \end{cases}$

Definition 5.7. Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a disjunctive Boolean network and $A^- \subseteq A^{\ominus}(f)$, we define the operator $\mathcal{G}_{\text{lab}}(f, A^-)$ as the output of the Algorithm 5.1, where:

$$(A^+)^* = \{(u, v) : \text{ there exists a path from } u \text{ to } v \text{ in } G[A^+]\}.$$

Algorithm 5.1: $\mathcal{G}_{\text{lab}}(f, A^-)$

Input: A disjunctive Boolean network $f : \mathbb{B}^n \to \mathbb{B}^n$ and a subset $A^- \subseteq A^{\ominus}(f)$ of $G = G(f)$ such that $G[A^-]$ is acyclic.

Output: A labeled digraph $G[A^-, A^+]$.

1 $A^+ \leftarrow \{(u, v) \in A(f) : \forall w \in N_f^+(u), (w, v) \notin A^-)\}$;
2 **if** $(A^+)^* \cap A^- = \emptyset$ **then return** $G[A^-, A^+]$;
3 **else return** $\mathcal{G}_{\text{lab}}(f, A^- \setminus (A^+)^*)$;

Note that the result of the $\mathcal{G}_{\text{lab}}(f, A^-)$ operator is a labeled digraph for which its parallel digraph, if update, is equal to $G(f)$. To prove its correctness, we can classify the arcs of $G(f)$ into two classes:

- Directly explained arcs: Those that are in the digraph $G(h)$ and have positive label.
- Indirectly explained arcs: Those arcs (u, v) that need an arc $(u, w) \in A(f)$ and $(w, v) \in A(h)$ with negative label.

Clearly, an indirectly explained arc (u, v) needs that arc (u, w) is directly or indirectly explained.

In the case of Algorithm 5.1, for each of the arcs $(u, v) \in A(f)$, we have two options:

- either there exists w such that (u, w) is in $A(f)$ and (w, v) is in A^-, so (u, v) is not added to A^+, and clearly (u, v) is an indirectly explained arc, or else
- there is no such w, therefore (u, v) is added to A^+ and thus directly explained.

In this way, to be sure that all the arcs of $A(f)$ can be explained, it is strictly necessary that at least one of the arcs of $A(f)$ is directly explained.

This can be guaranteed from the following proposition:

Proposition 5.7. *Let $h, f : \mathbb{B}^n \to \mathbb{B}^n$ be two disjunctive Boolean networks. If for all arc $(u, v) \in A(f)$, (u, v) is a indirectly explained arc then the set of negative arcs in h has at least one cycle.*

Proof. Let (u, v_0) be an arc indirectly explained, then there exists a vertex v_1 such that $(u, v_1) \in A(f)$ and $(v_1, v_0) \in A(h)$ with negative label. And so on, we can construct a succession of vertices $v_0, v_1, \ldots, v_n$ that fulfill this condition.

Without loss of generality, let us consider $v_n, \ldots, v_0$ the longest path of negative arcs in $A(h)$ that satisfy this condition (Figure 5.8).

And for the case of (u, v_n), it is necessary that it can be explained indirectly (initial premise), but there does not exist a vertex v_{n+1} such that $(u, v_{n+1}) \in A(f)$ and $(v_{n+1}, v_n) \in A(h)$ with negative label (since, in that case, $v_n, \ldots, v_0$ would not be the longest path). Therefore, that value j such that $(u, v_j) \in A(f)$ and $(v_j, v_n) \in A(h)$ with negative label, must necessarily be in the set $\{0, \ldots, n-1\}$, thus forming a cycle in the set of negative arcs in $G(h)$. □

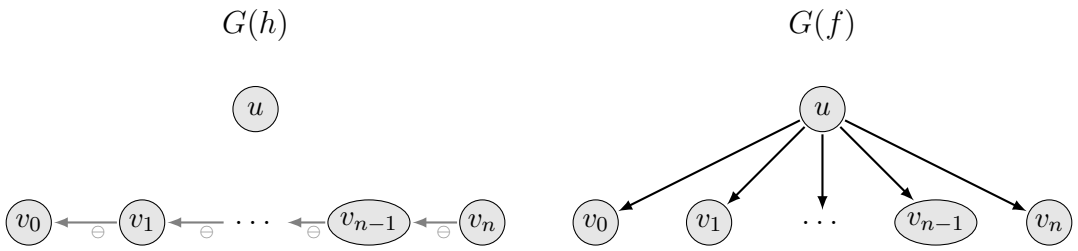


Figure 5.8: Explanation of Proposition 5.7.

In addition to each arc of f being explained (directly or indirectly) another interesting condition is that the resulting labeled graph is update. Condition $(A^+)^* \cap A^- = \emptyset$ eliminates several simple cases, but it is not sufficient as can be seen in Figure 5.9. To find an update solution, based on this one, it is necessary to study some properties previously.

An interesting set to study is the set of positive arcs generated by $\mathcal{G}_{\text{lab}}$, i.e. $\text{lab}^{\oplus}[\mathcal{G}_{\text{lab}}(f, A^-)]$. An important characteristic of this set is that for all disjunctive Boolean network h and update schedule s such that $h^s = f$, we have $\text{lab}^{\oplus}[\mathcal{G}_{\text{lab}}(f, A^-)] \subseteq A(h)$.

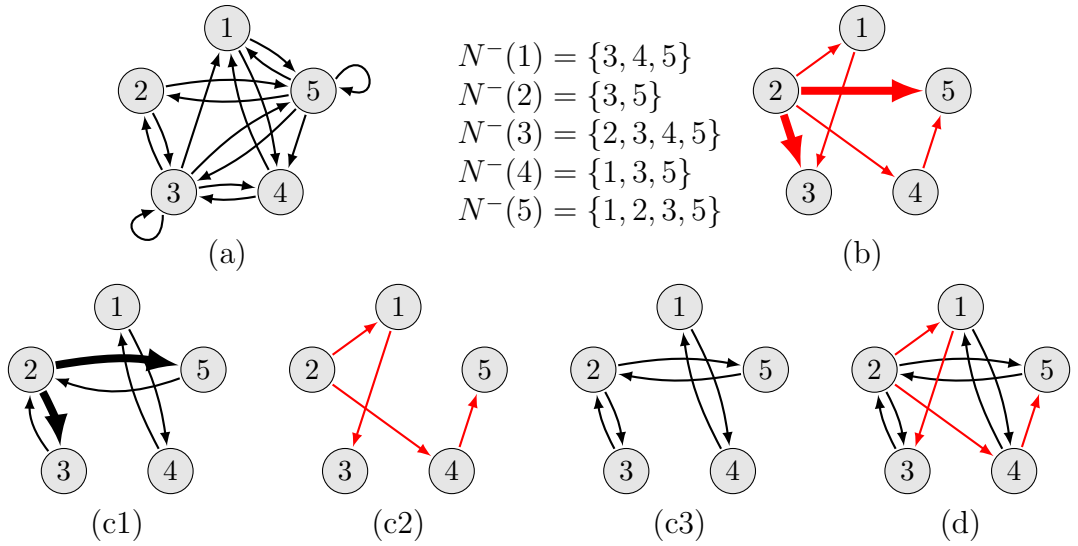


Figure 5.9: (a) A disjunctive Boolean network f. (b) $A^{\ominus}(f)$. (c) Iterations of $\mathcal{G}_{\text{lab}}(f, A^{\ominus}(f))$. (d) $\mathcal{G}_{\text{lab}}(f, A^{\ominus}(f))$ which is not update, because according to the labeled digraph there should be an update schedule s such that $s(2) < s(1) < s(3) \leq s(2)$, which is a contradiction.

The following result shows that given two sets of negative arcs (subsets of $A^{\ominus}(f)$), the positive arcs of $\mathcal{G}_{\text{lab}}$ on the larger set are also positive arcs of $\mathcal{G}_{\text{lab}}$ on the smaller set.

Proposition 5.8. *Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a disjunctive Boolean network and $A'' \subseteq A' \subseteq A^{\ominus}(f)$, then* $\text{lab}^{\oplus}[\mathcal{G}_{\text{lab}}(f, A')] \subseteq \text{lab}^{\oplus}[\mathcal{G}_{\text{lab}}(f, A'')]$.

Proof. Let us suppose that $A'' \subseteq A' \subseteq A^{\ominus}(f)$.

Let $(u, v) \in \text{lab}^{\oplus}[\mathcal{G}_{\text{lab}}(f, A')]$, then $\forall w \in N_f^+(u), (w, v) \notin A'$.

Since $A'' \subseteq A'$, then $\forall w \in N_f^+(u), (w, v) \notin A''$, and therefore, $(u, v) \in \text{lab}^{\oplus}[G(f), A'']$. Hence, $\text{lab}^{\oplus}[G(f), A'] \subseteq \text{lab}^{\oplus}[G(f), A'']$. □

The following result allows us to ensure that if there exists an acyclic set of negative arcs $A^- \subseteq A^{\ominus}(f)$ such that $\mathcal{G}_{\text{lab}}(f, A^-)$ is an update digraph, then $G_P(\mathcal{G}_{\text{lab}}(f, A^-)) = G(f)$.

Proposition 5.9. *Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a disjunctive Boolean network and $A^- \subseteq A^{\ominus}(f)$ such that $G[A^-]$ is acyclic. If $\mathcal{G}_{\text{lab}}(f, A^-)$ is an update digraph, then $G_P(\mathcal{G}_{\text{lab}}(f, A^-)) = G(f)$.*

Proof. $[G_P(\mathcal{G}_{\text{lab}}(f, A^-)) \subseteq G(f)]$ Let $(u, v) \in A(G_P(\mathcal{G}_{\text{lab}}(f, A^-)))$. We have two cases:

- If $(u, v) \in \text{lab}^{\oplus}[\mathcal{G}_{\text{lab}}(f, A^-)]$, then $(u, v) \in A(f)$.
- Otherwise, if $(u, v) \notin \text{lab}^{\oplus}[\mathcal{G}_{\text{lab}}(f, A^-)]$, since $(u, v) \in A(G_P(\mathcal{G}_{\text{lab}}(f, A^-)))$, then, by definition of parallel digraph, there exists a vertex w such that $(u, w) \in A(f)$ and $(w, v) \in A^-$. Since $(w, v) \in A^-$, then $N_f^-(w) \subseteq N_f^-(v)$. For this reason, since $(u, w) \in A(f)$, then $(u, v) \in A(f)$.

$[G(f) \subseteq G_P(\mathcal{G}_{\text{lab}}(f, A^-))]$ Let $(u, v) \in A(f)$. We have two cases:

- If it does not exist $w \in [n]$ such that $(u, w) \in A(f)$ and $(w, v) \in A^-$, then $(u, v) \in \text{lab}^{\oplus}[\mathcal{G}_{\text{lab}}(f, A^-)]$, therefore, $(u, v) \in A(G_P(\mathcal{G}_{\text{lab}}(f, A^-)))$.
- If there exists $w \in [n]$ such that $(u, w) \in A(f)$ and $(w, v) \in A^-$, then, by definition of parallel digraph, $(u, v) \in A(G_P(\mathcal{G}_{\text{lab}}(f, A^-)))$.

Hence, if $(u, v) \in A(f)$, then $(u, v) \in A(G_P(\mathcal{G}_{\text{lab}}(f, A^-)))$. Therefore, since $G_P(\mathcal{G}_{\text{lab}}(f, A^-)) \subseteq G(f)$ and $G(f) \subseteq G_P(\mathcal{G}_{\text{lab}}(f, A^-))$, we have $G_P(\mathcal{G}_{\text{lab}}(f, A^-)) = G(f)$. □

Remark 5.4. *Note that if $G[A^-]$ has a cycle, it is not possible to calculate $\mathcal{G}_{\text{lab}}(f, A^-)$. Also, it is not necessary, because by Corollary 5.6 we have a solution for the studied problem.*

The following proposition shows that if there is a solution (with negative arcs A^- and positive arcs B), then the set of negative arcs of $\mathcal{G}_{\text{lab}}(f, A^-)$ is exactly A^- and the set of positive arcs of $\mathcal{G}_{\text{lab}}(f, A^-)$ is a subset of B. Therefore, the solution obtained using $\mathcal{G}_{\text{lab}}(f, A^-)$ is minimal in the number of arcs.

Proposition 5.10. *Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a disjunctive Boolean network and $A^- \subseteq A^{\ominus}(f)$ such that $G[A^-]$ is acyclic. If there exists $B \subseteq [n] \times [n]$ such that $A^- \cap B = \emptyset$, $G[A^-, B]$ is an update digraph and $G_P(G[A^-, B]) = G(f)$, then $\text{lab}^{\ominus}[(\mathcal{G}_{\text{lab}}(f, A^-)] = A^-$ and $\text{lab}^{\oplus}[\mathcal{G}_{\text{lab}}(f, A^-)] \subseteq B$.*

Proof. Let us suppose that there exists $B \subseteq [n] \times [n]$ such that $A^- \cap B = \emptyset$ and $G_P(G[A^-, B]) = G(f)$.

If the $\mathcal{G}_{\text{lab}}(f, A^-)$ operator is applied, note that $(A^+)^* \cap A^- = \emptyset$, since if there is an arc (or a path) in A^+ that coincides with an arc in A^-, then there is no solution with A^- as negative arcs (because it breaks the update condition, since $s(u) \geq s(w_0) \geq \cdots \geq s(w_n) \geq s(v)$ (according to the positive path), and $s(u) < s(v)$ (according to the negative arc), which is a contradiction). For this reason, the $\mathcal{G}_{\text{lab}}$ operator does not make a new recursive call, hence $\text{lab}^{\ominus}[(\mathcal{G}_{\text{lab}}(f, A^-)] = A^-$. On the other hand, $\text{lab}^{\oplus}[\mathcal{G}_{\text{lab}}(f, A^-)]$ is not necessarily equal to B, because B, being part of an update solution, may contain arcs that $\mathcal{G}_{\text{lab}}$ omitted (because they are indirectly explained) and that do not affect the rest of the digraph (as can be seen in Figure 5.10).

For this reason, we can state that $\text{lab}^{\oplus}[\mathcal{G}_{\text{lab}}(f, A^-)]$ is a minimal set for the positive arcs of $\mathcal{G}_{\text{lab}}(f, A^-)$ and this together with A^- is a minimal set for the arcs of $\mathcal{G}_{\text{lab}}(f, A^-)$. □

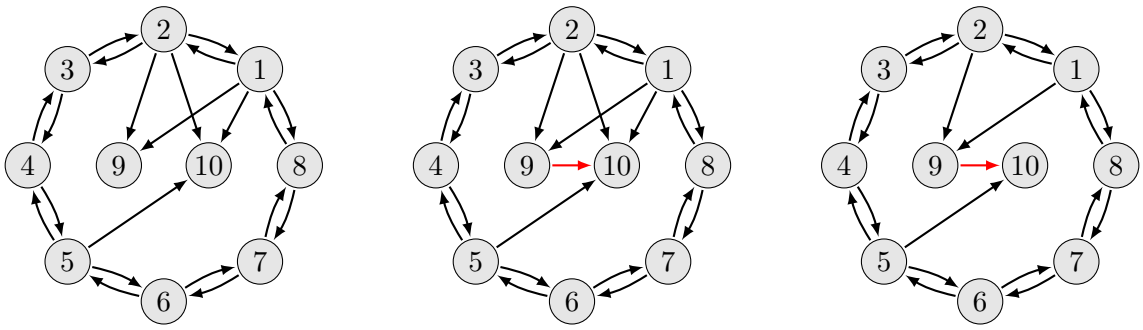


Figure 5.10: Example of $\text{lab}^{\oplus}[\mathcal{G}_{\text{lab}}(f, A^-)] \subseteq B$.

Definition 5.8. Let (G, lab) be a labeled digraph. A partition $\{V_1, V_2\}$ of $[n]$ is said to be *admissible* if satisfies the following conditions:

1. $\exists (u, v) \in A(G), u \in V_1 \wedge v \in V_2$,
2. $\forall (u, v) \in A(G), u \in V_1 \wedge v \in V_2 \Rightarrow \text{lab}(u, v) = \ominus$,
3. $\forall (u, v) \in A(G), u, v \in V_2 \Rightarrow \text{lab}(u, v) = \oplus$.

Lemma 5.11. *Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a disjunctive Boolean network and $\{V_1, V_2\}$ an admissible partition of $\mathcal{G}_{\text{lab}}(f, A^{\ominus}(f))$. If we define $A^- = \{(u, v) \in A(G) : u \in V_1 \wedge v \in V_2\}$, where G is the digraph of the resulting labeled digraph, then $\mathcal{G}_{\text{lab}}(f, A^-)$ is an update digraph and $G_P(\mathcal{G}_{\text{lab}}(f, A^-)) = G(f)$.*

Proof. We denoted by (G, lab) the labeled digraph obtained by $\mathcal{G}_{\text{lab}}$ operator.

To show that (G, lab) is an update digraph, we prove that if we define $s = V_1, V_2$, then $\text{lab} = \text{lab}_s$.

Note that for all (u, v) in $A(G)$ such that $u \notin V_1$ or $v \notin V_2$, $\text{lab}(u, v) = \text{lab}_s(u, v) = \oplus$, because all these arcs, if they appear in G (resulting from the $\mathcal{G}_{\text{lab}}$ operator), have a label $\oplus$ since they are not in A^-.

On the other hand, we prove that $\text{lab}^{\ominus}[G, \text{lab}] = A^-$. Note that when choosing the arcs in A^- only the arcs from V_1 to V_1 and from V_2 to V_1 have been removed from $\text{lab}^{\ominus}[G, \text{lab}]$ (those from V_1 to V_2 remain in A^- and there are no negative arcs from V_2 to V_2, since V_1 and V_2 is an admissible partition). For this reason, for every new arc (u, v) in A^+, $v \in V_1$. Therefore, in the first iteration of the $\mathcal{G}_{\text{lab}}$ operator, no edge is removed from A^-, hence $\text{lab}^{\ominus}[G, \text{lab}] = A^-$.

Thus, since the only negative arcs are from V_1 to V_2, we have $lab = lab_s$, with $s = V_1, V_2$, so (G, lab) is update.

Finally, since $\mathcal{G}_{\text{lab}}(f, A^{\ominus}(f))$ is update, according to Proposition 5.9, $G_P(\mathcal{G}_{\text{lab}}(f, A^{\ominus}(f))) = G(f)$. □

Theorem 5.12. *Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a disjunctive Boolean network. There exists a solution for D-DEN problem if and only if $A^{\ominus}(f)$ has a cycle or $\text{lab}^{\ominus}[\mathcal{G}_{\text{lab}}(f, A^{\ominus}(f))] \neq \emptyset$. Besides, if a solution exists, it can be found in polynomial time.*

Proof. If $A^{\ominus}(f)$ has a cycle, we have at least 2 vertices with the same input neighborhood. With those vertices with equal neighborhood we have the conditions of Corollary 5.6 and therefore there is a solution to the D-DEN problem.

On the contrary, if $A^{\ominus}(f)$ is acyclic, the first step is to compute $\mathcal{G}_{\text{lab}}(f, A^{\ominus}(f))$.

Next, we use the following algorithm to define the set V_2, where ComponentDigraph(G) is a digraph $(\hat{V}, \hat{A})$ defined as follows:

- $\hat{V} = \{G_1, G_2, \ldots, G_k\}$, where G_i are the strongly connected components of G.
- $(G_i, G_j) \in \hat{A}$ if and only if there exists an arc from a vertex in G_i to a vertex in G_j.

Algorithm 5.2: AdmPartition(G, lab)

Input: A labeled digraph (G, lab).
Output: A partition $\{V_1, V_2\}$ of $V(G)$.

1 $G^{\oplus} \leftarrow \text{lab}^{\oplus}[(G, \text{lab})]$;
2 $(\hat{V}, \hat{A}) \leftarrow \text{CD}(G^{\oplus})$;
3 $Q \leftarrow \{G_i \in \hat{V} : \nexists G_j \in \hat{V}, (G_j, G_i) \in \hat{A}\}$; `// source components`
4 $v^* \leftarrow$ Null;
5 **while** $v^* = Null$ **do**
6 $\quad G_q \leftarrow$ first element of Q;
7 $\quad$ **if** $\exists u \in G_q \wedge \exists (w, u) \in A(G), \text{lab}(w, u) = \ominus$ **then** $v^* \leftarrow u$;
8 $\quad$ **else** $Q \leftarrow Q \cup \{G_i \in \hat{V} : (G_q, G_i) \in \hat{A}\}$;
9 **end**
10 $V_2 \leftarrow \{v \in V(G)$: $\exists$ a path in $G^{\oplus}$ from v to some vertex in the same component of $v^*\}$;
11 $V_1 \leftarrow V(G) \setminus V_2$;
12 **return** $\{V_1, V_2\}$

Note that the resulting set V_2 is never empty since $\text{lab}^{\ominus}[\mathcal{G}_{\text{lab}}(f, A^{\ominus}(f))] \neq \emptyset$.

The strategy presented in this algorithm is to do a Breadth First Search in the digraph for strongly connected components of the positive arcs of the labeled digraph. The goal of the search is to find the first strongly connected component that receives a negative arc (which we call the pivot component). Once this component is found, a partition is created: in V_2 are all the vertices that can reach the pivot component by a path of positive arcs and in V_1 the rest of vertices.

Now, we prove that $\{V_1, V_2\}$ is an admissible partition.

- Note that in the arc (w, u) (with label $\ominus$) that activate the line 7 of Algorithm 5.2, which triggers the construction of V_2, $u \in V_2$ (by how the algorithm is defined) and $w \in V_1$ (because if $w \in V_2$, (w, u) it would have been removed from A^- by applying $\mathcal{G}_{\text{lab}}$ operator). Therefore, $\exists (u, v) \in A(G), u \in V_1 \wedge v \in V_2$,
- By contradiction, let us suppose that there exists an arc $(u, v) \in V_1 \times V_2$, such that $\text{lab}(u, v) = \oplus$. Note that since $v \in V_2$, there exists $k \geq 1$, $(v, p) \in A(\text{lab}^{\oplus}[(G, \text{lab})])^k$ where p is a pivot vertex (found from lines 1 to 7 of Algorithm 5.2) and since $u \in V_1$, it does not exist $k' \geq 1$, $(u, p) \in A(\text{lab}^{\oplus}[(G, \text{lab})])^{k'}$. Since we suppose that $(u, v) \in A(G)$ and $\text{lab}(u, v) = \oplus$, then there exists $k' = k + 1$, therefore, $u \in V_2$, which is a contradiction. Therefore $\forall (u, v) \in A(G), u \in V_1 \wedge v \in V_2 \implies \text{lab}(u, v) = \ominus$,

- By contradiction, let us suppose that there exists an arc $(u, v) \in V_2 \times V_2$, such that $\text{lab}(u, v) = \ominus$. Note that since $u, v \in V_2$, there exists $k, k' \geq 1$, $(v, p) \in A(\text{lab}^{\oplus}[(G, \text{lab})])^k$ and $(u, p) \in A(\text{lab}^{\oplus}[(G, \text{lab})])^{k'}$ where p is a pivot vertex (found from lines 1 to 7 of Algorithm 5.2). If $(u, v) \in A(G)$ and $\text{lab}(u, v) = \ominus$, then p would not have been chosen as a pivot (since v appears earlier in the poset), which is a contradiction. Therefore $\forall (u, v) \in A(G), u, v \in V_2 \implies \text{lab}(u, v) = \oplus$.

With what we learned above, we can build the following algorithm:

Algorithm 5.3: D-DENPSolve(f)

Input: A disjunctive Boolean network $f : \mathbb{B}^n \to \mathbb{B}^n$.
Output: (G, lab) an update digraph such that $G_P(G, \text{lab}) = G(f)$ if there exists a solution of the D-DEN problem with instance f, or Null otherwise.

```
1  if A^⊖(f) has a cycle then
2  |  Let u and v be two vertices such that N_f^-(u) = N_f^-(v);
3  |  return G[{(u,v)}, A(f) \ {(x,v) : x ∈ N_f^-(v)}]
4  end
5  else
6  |  (G, lab) ← G_lab(f, A^⊖(f));
7  |  if lab^⊖[(G, lab)] = ∅ then return Null;
8  |  if (G, lab) is update then return (G, lab);
9  |  {V_1, V_2} ← AdmPartition(G, lab);
10 |  A^- ← {(u,v) ∈ A(G, lab) : u ∈ V_1 ∧ v ∈ V_2};
11 |  return G_lab(f, A^-);
12 end
```

Given a disjunctive Boolean network $f : \mathbb{B}^n \to \mathbb{B}^n$, first we obtain the $A^{\ominus}(f)$, in this way, we know which arcs, of a possible digraph G, can be labeled $\ominus$. If $A^{\ominus}(f)$ has a cycle, by Corollary 5.6, there is a solution (lines 1 to 3).

Otherwise, if $A^{\ominus}(f)$ is acyclic, the operator $\mathcal{G}_{\text{lab}}$ can be applied. If the resulting labeled digraph (G, lab) has no negative arcs, then no neighborhood is contained in another one and, according to Lemma 5.4, there is no non-trivial solution, therefore the algorithm ends (line 7).

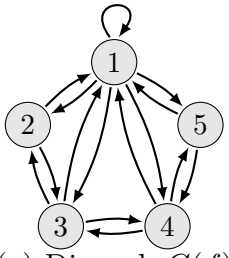
If $\text{lab}^{\ominus}[(G, \text{lab})]$ is not empty and (G, lab) is update, then we found a solution (line 8).

Finally, if (G, lab) is not update, since we know that $\text{lab}^{\ominus}[(G, \text{lab})] \neq \emptyset$, we can find an admissible partition of $\mathcal{G}_{\text{lab}}(f, A^{\ominus}(f))$ and with that partition build a solution (lines 9 to 11). □

Example 5.4. Given f a disjunctive Boolean network (Figure 5.11(a)), the first step is create the digraph $A^{\ominus}(f)$ (Figure 5.11(b)).

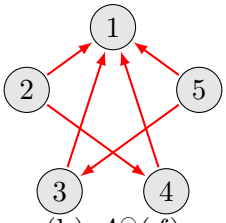
Since $A^{\ominus}(f)$ has no cycles, the algorithm continues. The next step is to get $\mathcal{G}_{\text{lab}}(f, A^{\ominus}(f))$. First, A^+ is calculated (Figure 5.11(c)). Then, since no edge (or path) of A^+ coincides with the arcs of $A^{\ominus}(f)$, the operator $\mathcal{G}_{\text{lab}}$ ends and the result is Figure 5.11(d) and its negative arcs are Figure 5.11(b). Since $\text{lab}^{\ominus}[\mathcal{G}_{\text{lab}}(f, A^{\ominus}(f))] \neq \emptyset$, the algorithm continues.

The labeled digraph of the operator $\mathcal{G}_{\text{lab}}$ (Figure 5.11(d)) is not update. For this reason, we look for an admissible partition. First, $G^{\oplus}$ (Figure 5.11(c)) is obtained. Then, the POSET digraph (Figure 5.11(e)) is generated, where we have three strongly connected components: $\{1\}, \{2, 3\}, \{4, 5\}$. We find negative arcs from the component $\{2, 3\}$ to $\{1\}$. With this, we can define $V_2 = \{1\}$ and $V_1 = \{2, 3, 4, 5\}$. With this set, we define the labeling in Figure 5.11(f). Finally, the parallel digraph of the labeling digraph in Figure 5.11(f) is exactly Figure 5.11(a).

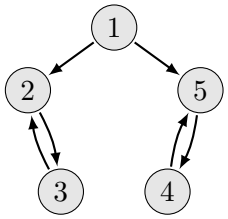


(a) Digraph $G(f)$

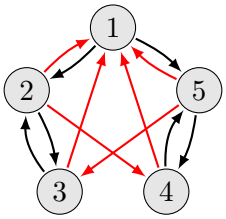
$N^-(1) = \{1, 2, 3, 4, 5\}$
$N^-(2) = \{1, 3\}$
$N^-(3) = \{1, 2, 4\}$
$N^-(4) = \{1, 3, 5\}$
$N^-(5) = \{1, 4\}$



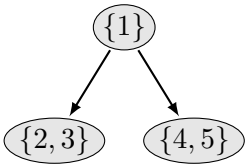
(b) $A^{\ominus}(f)$



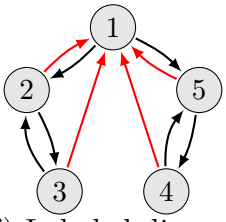
(c) A^+ according Algorithm 5.1



(d) $\mathcal{G}_{\text{lab}}(f, A^{\ominus}(f))$



(e) The POSET digraph according Algorithm 5.2



(f) Labeled digraph according V_1 and V_2

Figure 5.11: Figures of Example 5.4.

Theorem 5.13. *D-DEN can be decided in polynomial time.*

Proof. With what we learned above, we can build the following algorithm:

Algorithm 5.4: D-DENDecider(f)

Input: A disjunctive Boolean network $f : \mathbb{B}^n \to \mathbb{B}^n$.
Output: **True** if there exists a solution of the D-DEN problem with instance f, or **False** otherwise.

1 **if** $A^{\ominus}(f)$ *has a cycle* **then return** ***True***;
2 **if** $\text{lab}^{\ominus}[\mathcal{G}_{\text{lab}}(f, A^{\ominus}(f))] \neq \emptyset$ **then return** ***True***;
3 **return** ***False***;

Note that the simplicity of this algorithm lies in answering two questions:

- Does $A^{\ominus}(f)$ have a cycle?: If the answer is yes, we have at least two vertices with the same input neighborhood, therefore, according to Corollary 5.6, there is a solution.
- $\text{lab}^{\ominus}[\mathcal{G}_{\text{lab}}(f, A^{\ominus}(f))] \neq \emptyset$?: If the answer is yes, according to Theorem 5.12, there is a solution.

If the answer to both questions is no, then there is no solution, because all the candidates to be negative arcs in some solution have been discarded. □

5.6 Conclusions and future work

In this chapter, we present different approaches to the problem of dynamically equivalent networks.

As could be seen, solving the general problem, i.e., given a Boolean network f, finding another Boolean network $\bar{f}$ and an update schedule $\bar{s}$ such that $\bar{f}^{\bar{s}} = f$ is NP-Hard, since it is as difficult as 3-SAT, but it does present an approach to finding a possible solution: if there exists a solution with an update schedule with more than two blocks, then there exists a solution with an update schedule of only two blocks.

Now, if we restrict the problem to disjunctive networks, finding a disjunctive Boolean network h and an update schedule s such that $h^s = f$, this problem can be solved in polynomial time.

It is worth noting that the fact that in the labeled digraph contains an arc (u, v) whose label is negative only if $N_f^-(u) \subseteq N_f^-(v)$ in the parallel digraph is a very important result since it implies that any digraph whose neighborhoods are not comparable, has no other dynamically equivalent network different to the trivial one.

With all these results, there remain several ideas to explore, such as finding an algorithm that can solve the general problem, and explore enumeration algorithms, in the case we fix some element of the triplet (h, s, f). Another idea would be to analyze if for a Boolean network f, there exists a Boolean network h and an update schedule s such that s has some particularity (e.g.: s is a sequential update schedule) and that $h^s = f$.

Chapter 6

Subproblems of Dynamically Equivalent Network problem

6.1 Introduction

In the previous chapter, we present an algorithm that, given a disjunctive Boolean network, returns another Boolean network and another update schedule having the same dynamic behavior. But what happens if some of the parameters of the expected output are fixed a priori? In this chapter we analyze two variants that answer this question.

6.2 D-DEN problem with fixed update schedule

In the previous chapter, it was shown that given a disjunctive Boolean network f deciding whether if there exists a disjunctive Boolean network h and a non-trivial update schedule s such that $h^s = f$ can be solved in polynomial time. The next natural question is: Is there a way to find all the disjunctive Boolean networks h quickly? This section shows that by fixing one of the parameters we can answer this question.

6.2.1 Decision problem

Formally, we define the problem as follows:

> Dynamically equivalent disjunctive Boolean network with fixed schedule problem
>
> **Input:** f is a disjunctive Boolean network and s is an update schedule.
>
> **Question:** Does there exist h a disjunctive Boolean network such that $h^s = f$?

One of the main difficulties in studying the existence of a network h is that given f and s, we can be in different scenarios:

- There is only one disjunctive Boolean network h, such that $h^s = f$ (Figure 6.1).
- There is more than one disjunctive Boolean network h, such that $h^s = f$ (Figure 6.2).
- There is no disjunctive Boolean network h, such that $h^s = f$ (Figure 6.3).

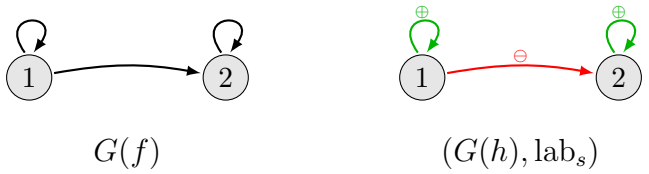


Figure 6.1: Given f and $s = \{1\}\{2\}$ there is only one BN h such that $h^s = f$.

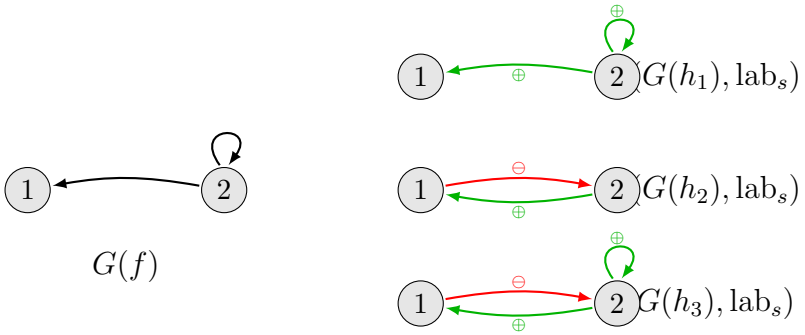


Figure 6.2: Given f and $s = \{1\}\{2\}$ there is three BNs h such that $h^s = f$.

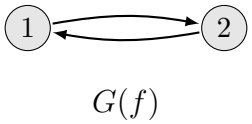


Figure 6.3: Given f and $s = \{1\}\{2\}$ there is no BN h such that $h^s = f$.

Analyzing the case with multiple preimages, it can be seen that $G(h_1) \subseteq G(h_3)$ and $G(h_2) \subseteq G(h_3)$, moreover, $G(h_3) = G(h_1) \cup G(h_2)$. Which leads us to prove Proposition 6.1.

Given $g, h : \mathbb{B}^n \to \mathbb{B}^n$ two disjunctive Boolean networks, we define $g \vee h : \mathbb{B}^n \to \mathbb{B}^n$ such that $(g \vee h)(x) = g(x) \vee h(x)$. It is easy to see that $\forall u \in [n], N^-_{g\vee h}(u) = N^-_g(u) \cup N^-_h(u)$, and, therefore, $G(g \vee h) = G(g) \cup G(h)$.

Proposition 6.1. *Let s be an update schedule, and let f, g and h be three disjunctive Boolean networks such that $g^s = h^s = f$, then $(g \vee h)^s = f$.*

To prove this, it is necessary to demonstrate the following lemma:

Lemma 6.2. *Let s be an update schedule and $f, h : \mathbb{B}^n \to \mathbb{B}^n$ be two disjunctive Boolean networks. If $G(f) \subseteq G(h)$, then $G(f^s) \subseteq G(h^s)$.*

Proof. Let us suppose $G(f) \subseteq G(h)$. Now, let (u, v) be any arc in $A(f^s)$, by induction on $s(v)$, we prove that $A(f^s) \subseteq A(h^s)$.

Basis. If $s(v) = 1$, since $(u, v) \in A(f^s)$, then $(u, v) \in A(f)$ and $s(u) \geq s(v)$. Since $G(f) \subseteq G(h)$, $(u, v) \in A(h)$ and $s(u) \geq s(v)$, therefore $(u, v) \in A(h^s)$

Hypothesis of induction. If $s(v) < k$, $(u, v) \in A(f^s) \implies (u, v) \in A(h^s)$.

Inductive step. Let $(u, v) \in A(f^s)$ be such that $s(v) = k$, we have two options:

- if $(u, v) \in A(f)$ and $s(u) \geq k$, then $(u, v) \in A(h)$ and $s(u) \geq k$, therefore $(u, v) \in A(h^s)$.
- if there exists $w \in V$ such that $s(w) < s(v) = k$, $(u, w) \in A(f^s)$ and $(w, v) \in A(f)$, then by hypothesis of induction $(u, w) \in A(h^s)$, and since $G(f) \subseteq G(h)$, $(w, v) \in A(h)$, therefore $(u, v) \in A(h^s)$.

□

Proof of Proposition 6.1. $[G(f) \subseteq G((g \vee h)^s)]$ Note that this is a result of Lemma 6.2, since $G(g) \subseteq G(g \vee h)$, then $G(f) = G(g^s) \subseteq G((g \vee h)^s)$.

$[G((g \vee h)^s) \subseteq G(f)]$ Let (u, v) be any arc in $A((g \vee h)^s)$, by induction on $s(v)$, we prove that $(u, v) \in A(f)$.

Basis. If $s(v) = 1$, since $(u, v) \in A((g \vee h)^s)$, then:

- $(u, v) \in A(g)$ and $s(u) \geq s(v)$, therefore $(u, v) \in A(g^s)$, or
- $(u, v) \in A(h)$ and $s(u) \geq s(v)$, therefore $(u, v) \in A(h^s)$.

Therefore, since $g^s = h^s = f$, then $(u, v) \in A(f)$.

Hypothesis of induction. If $s(v) < k$, $(u, v) \in A((g \vee h)^s) \Longrightarrow (u, v) \in A(f)$.

Inductive step. Let $(u, v) \in A((g \vee h)^s)$ be such that $s(v) = k$, we have two options:

- if $(u, v) \in A(g)$ (or $A(h)$) and $s(u) \geq k$, then $(u, v) \in A(g^s)$ (or $A(h^s)$), therefore $(u, v) \in A(f)$.
- if there exists $w \in V$ such that $s(w) < s(v) = k$, $(u, w) \in A((g \vee h)^s)$ and $(w, v) \in A(g)$ (or $A(h)$), then by hypothesis of induction $(u, w) \in A(f)$, and since $(w, v) \in A(g)$ (or $A(h)$) and $s(w) < s(v)$, then, by definition of parallel digraph, $(u, v) \in A(f)$.

□

Theorem 6.3. *Dynamically equivalent disjunctive Boolean network with fixed schedule can be solved in polynomial time.*

Given $f : \mathbb{B}^n \to \mathbb{B}^n$ a disjunctive Boolean network and an update schedule s, Algorithm 6.1 decides, if there exists, a disjunctive Boolean network h such that $h^s = f$. Furthermore, if h exists, it gives us the maximal disjunctive Boolean Network that satisfies the property.

To prove the correctness of the algorithm, the following lemmas are necessary:

Lemma 6.4. *Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a disjunctive Boolean network, s be an update schedule and h the Disjunctive Boolean network obtained by Algorithm 6.1. If there exists g such that $g^s = f$, then $G(g) \subseteq G(h)$.*

Proof. By contradiction, let us suppose that there exists $(u, v) \in A(g) \setminus A(h)$.

If $s(u) \geq s(v)$: then $(u, v) \in A(g^s)$, therefore $(u, v) \in A(f)$. Also, since $(u, v) \in A(f)$ and $s(u) \geq s(v)$, according to line 1 of Algorithm 6.1, $(u, v) \in A^+$, therefore $(u, v) \in A(h)$, which is a contradiction.

If $s(u) < s(v)$: since $(u, v) \notin A(h)$, then $N_f^-(u) \not\subseteq N_f^-(v)$ (according to line 1 of Algorithm 6.1). Also, there exists $w \in [n]$, such that $(w, u) \in A(f)$ and $(w, v) \notin A(f)$. Since $(w, u) \in A(f)$, and $(u, v) \in A(g)$, then, by definition of parallel digraph, $(w, v) \in A(f)$, which is a contradiction. □

Lemma 6.5. *Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a disjunctive Boolean network and s be an update schedule. If Algorithm 6.1 with inputs f and s returns Null then it does not exist a disjunctive Boolean network h such that $h^s = f$.*

Proof. By contradiction, let us suppose that there exists h a disjunctive Boolean network such that $h^s = f$ and Algorithm 6.1 returns Null.

Since Algorithm 6.1 returns Null, $\text{BN}([n], A^- \cup A^+)^s \neq f$. We have two scenarios:

1. There exists $(u, v) \in A(h) \setminus (A^- \cup A^+)$:
   - If $s(u) \geq s(v)$, then $(u, v) \in A(f)$, also (according to Algorithm 6.1) $(u, v) \in A^+$, which is a contradiction.
   - If $s(u) < s(v)$, then $N_f^-(u) \not\subseteq N_f^-(v)$, since it was discarded by line 1 of Algorithm 6.1. Also, let w be a vertex in $N_f^-(u) \setminus N_f^-(v)$ (non-empty), since $(w, u) \in A(f)$, $(u, v) \in A(h)$ and $s(u) < s(v)$, then $(w, v) \in A(f)$, which is a contradiction.
2. There exists $(u, v) \in (A^- \cup A^+) \setminus A(h)$:
   - If $s(u) \geq s(v)$, then $(u, v) \in A^+$, therefore, $(u, v) \in A(f)$. Since $(u, v) \in A(f)$, if $A(g) = A(h) \cup \{(u, v)\}$, then $g^s = f$, which is a contradiction to $\text{BN}([n], A^- \cup A^+)^s \neq f$
   - If $s(u) < s(v)$, then $(u, v) \in A^-$, therefore, $N_f^-(u) \subseteq N_f^-(v)$. Let $A(g) = A(h) \cup \{(u, v)\}$, if g^s has an arc $(u', v') \in A(f)$, then so does g^s. Therefore, if $g^s = f$, then $\text{BN}([n], A^- \cup A^+)^s = f$, which is a contradiction (Figure 6.4).

□

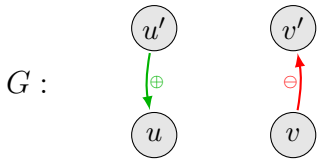


Figure 6.4: Consider that $(u', v') \notin A(f)$, then adding (u, v) with $s(u) < s(v)$, would generate $(u', v') \in A(f)$. But if we consider that $N_f^-(u) \subseteq N_f^-(v)$, then $(u', v) \in A(f)$, therefore $(u', v') \in A(f)$, which is a contradicition

Algorithm 6.1: MaximumDEDBN(f, s)

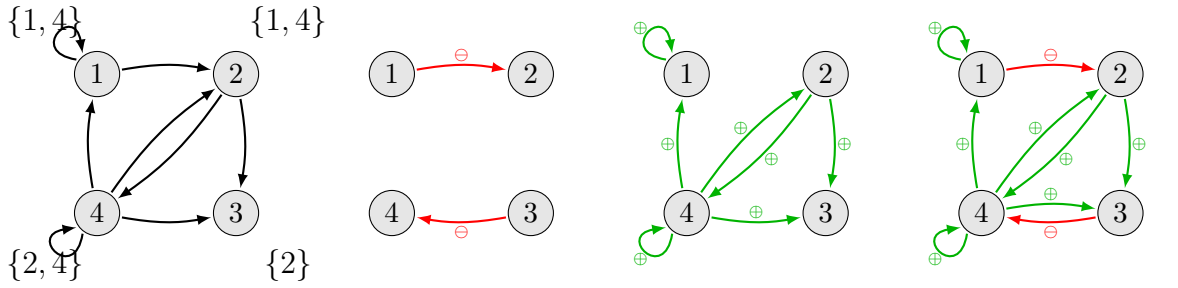
Input: A disjunctive Boolean network f and an update schedule s.
Output: The maximum disjunctive Boolean network h such that $h^s = f$ or NULL if it does not exist.

```
1 A⁻ ← {(u, v) ∈ A^⊖(f) : s(u) < s(v)};
2 if A⁻ = ∅ then return Null;
3 A⁺ ← {(u, v) ∈ A(f) : s(u) ≥ s(v)};
4 h ← BN([n], A⁻ ∪ A⁺);
5 if h^s = f then return h;
6 else return Null;
```

Proof of Theorem 6.3. Given a disjunctive Boolean network f and an update schedule s, we have an algorithm that finds the maximum dynamically equivalent disjunctive Boolean network. The running time, in the worst case, is $O(n^3)$, so we can conclude that the problem can be solved in polynomial time. □

Example 6.1. Given a disjunctive Boolean network f (Figure 6.5(a)) and the update schedule $s = \{3\}\{1\}\{2,4\}$, the first step is create the sets $A^- \subseteq A^{\ominus}(f)$ (Figure 6.5(b)) and A^+ (Figure 6.5(c)).

Finally, we check that $h^s = f$. Since all the arcs in f have been justified, the maximum preimage is the union of the arcs in A^+ and A^-.



(a) $= G(f)$ (b) $= A^-$ (c) $= A^+$ (d) $= (G(h), \text{lab}_s)$

Figure 6.5: A successful maximum preimage search with $s = \{3\}\{1\}\{2,4\}$.

Example 6.2. Given the disjunctive Boolean network f (Figure 6.6(a)) and the update schedule $s = \{1\}\{2\}$, the first step is create the set $A^- \subseteq A^{\ominus}(f)$ (Figure 6.6(b)). Then, A^+ (Figure 6.6(c)) is generated.

Finally, since $h^s = h \neq f$, there is no way to generate the arc $(1,2)$ with the schedule $\{1\}\{2\}$.

(a) $G(f)$ (b) A^- (c) $A^+ = A(h) = A(h^s)$

Figure 6.6: A failed maximum preimage search with $s = \{1\}\{2\}$ ($(1,2)$ cannot be created).

6.2.2 Enumeration of solutions with fixed schedule

As seen above, deciding the D-DEN problem with a fixed schedule can be solved in polynomial time, but Algorithm 6.1 returns the maximum dynamically equivalent disjunctive Boolean network with a fixed schedule, but what happens if we want to know all of them?

As seen in Figure 6.2, there are cases of f and s that have more than one solution and Algorithm 6.1 gives us the maximum solution in the number of arcs, if it exists, but how can we find the rest of the solutions?

To solve this new question, we define the following problem:

> Enumeration of dynamically equivalent disjunctive Boolean networks with fixed update schedule
>
> **Input:** f is a disjunctive Boolean network and s is an update schedule.
>
> **Question:** Is it possible to find the complete set of disjunctive Boolean networks h such that $h^s = f$?

To develop an algorithm that allows us to solve this problem, the following result is analyzed:

Lemma 6.6. *Let s be an update schedule and let h and h'' two disjunctive Boolean networks such that $G(h'') \subseteq G(h)$ and $h^s = h''^s = f$. Then, for all h' such that $G(h'') \subseteq G(h') \subseteq G(h)$, $h'^s = f$.*

Proof. Since $G(h'') \subseteq G(h') \subseteq G(h)$, by Lemma 6.2:

$$A(h''^s) \subseteq A(h'^s) \subseteq A(h^s) = A(f) \subseteq A(h'^s) \subseteq A(f).$$

For this reason, $h'^s = f$. □

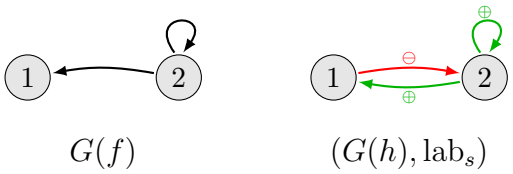
Given a disjunctive Boolean network f and an update schedule s. We define an algorithm that lists all the existing dynamically equivalent Boolean network with a fixed update schedule.

Algorithm 6.2: enumerationDEDBN(f, s)

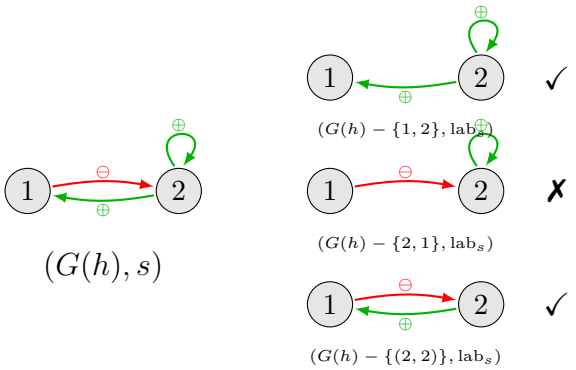
Input: A disjunctive Boolean network f and an update schedule s.
Output: A list with all the preimages of f and s.

```
h'' ← maximumDEDBN(f, s);
S ← {h''};
Q ← {h''};
while Q ≠ ∅ do
    Let h be an element of Q;
    Q ← Q \ {h};
    for (u, v) ∈ A(h) do
        A(h') ← A(h) \ {(u, v)};
        if h'^s = f then
            S ← S ∪ {h'};
            Q ← Q ∪ {h'};
        end
    end
end
return S
```

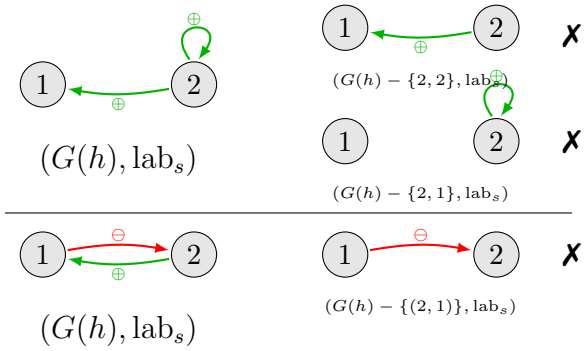
1. Given a disjunctive Boolean network f and an update schedule $s = \{1\}\{2\}$. If it exists, the maximum dynamically equivalent Boolean network (h'') is obtained according to Algorithm 6.1. Since $h''^s = f$, h'' is added to a solution set S and since any other dynamically equivalent Boolean network is a sub-network of h'', it is added to the revision set Q.



2. While Q is not empty, Let h be an element extracted from Q: For all $(u, v) \in A(h)$, is generated $A(h') = A(h) \setminus \{(u, v)\}$. If $h'^s = f$, h' is added to S and Q. If $h'^s \neq f$, h' is discarded.



3. The process is repeated for the remaining elements in Q



4. When there are no more sub-networks to analyze, the process is finished and the solution is the set S.

Figure 6.7: Example of enumerationDEDBN Algorithm.

Theorem 6.7. *The computational time of enumerationDEDBN is polynomial with respect to the number of solutions in the set S.*

Proof. First, the initial solution, if it exists, can be obtained in polynomial time (according to Theorem 6.3). Once a solution is obtained, the steps necessary to obtain another solution are: For each arc of the solution, that is, m times, construct the Boolean network h' in time $O(n^2)$ and check if h'^s (which is obtained in polynomial time) is equal to f. Finally, once the last solution is obtained, reaching the end of the algorithm takes in the worst case, $m^2 = O(n^4)$. With all this, we can affirm that the algorithm is time delay polynomial, that is, the time between the output of any

solution and the next one is bounded by a polynomial function of the input size, in the worst case. □

Theorem 6.8. *The computational time of enumerationDEDBN, in worst case, is exponential.*

Proof. Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a disjunctive Boolean network such that:

- $A(f) = \{(i,j) : i < j\} \cup \{(i,i) : i \in \{1,\dots,n-1\}\}$

(see Figure 6.8) and let s be a sequential update schedule $\{1\}\{2\}\dots\{n\}$, the number of dynamically equivalent disjunctive Boolean networks is equal to $2^{\binom{n-1}{2}} = 2^{O(n^2)}$.

Since enumerationDEDBN is time delay polynomial and there exists an example with an exponential number of solutions, the computational time of enumerationDEDBN, in the worst case, is exponential. □

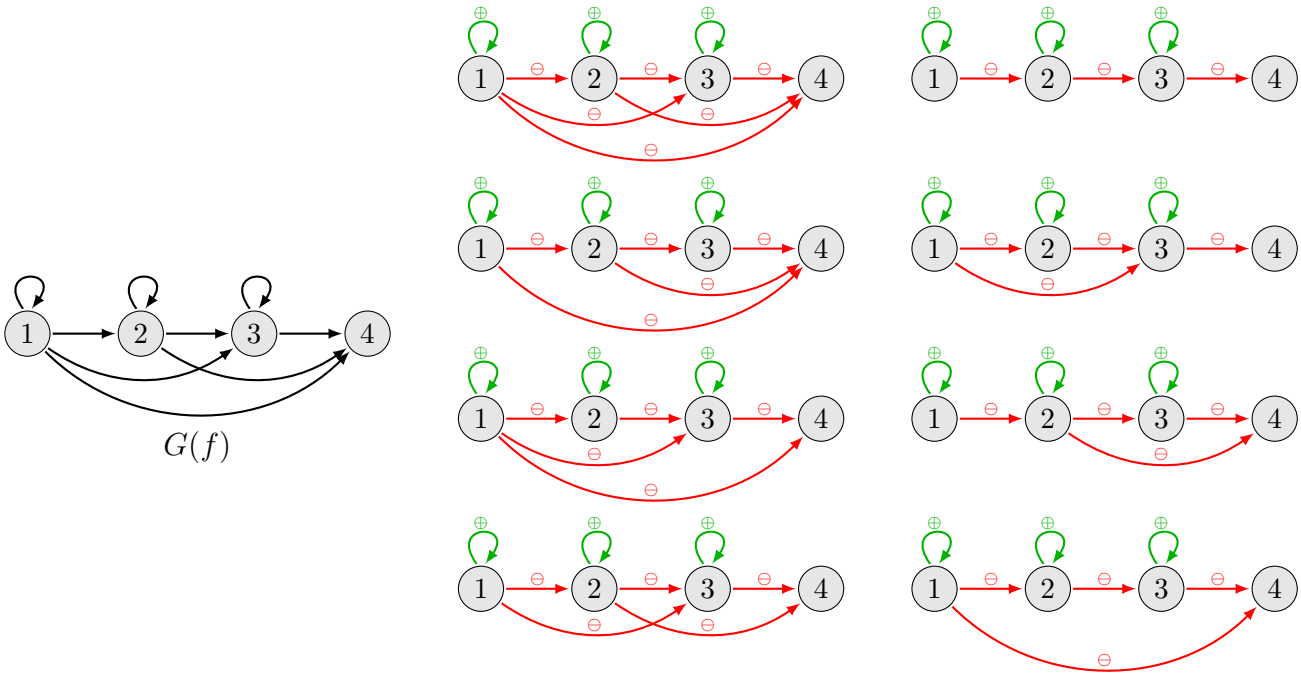


Figure 6.8: f and the different preimages when $s = \{1\}\{2\}\{3\}\{4\}$.

6.3 D-DEN problem with fixed Boolean network

It has already been analyzed how the algorithm behaves when we have the Boolean network f and an update schedule s. Does the same happen when we have two Boolean networks f and h?

6.3.1 Decision problem

This new problem is formalized as follows:

Dynamically equivalent disjunctive Boolean networks with fixed Boolean network problem

Input: h and f be two disjunctive Boolean networks.

Question: Does there exist an update schedule $s \nsim_{G(h)} s_p$ such that $h^s = f$?

Definition 6.1. Let $f, h : \mathbb{B}^n \to \mathbb{B}^n$ be two disjunctive Boolean networks, we say that an update schedule s is a *k-valid update schedule* if:

$$\forall u \in [n], s(u) \leq k \Longrightarrow N_f^-(u) = N_{h^s}^-(u)$$

In order to build a solution, the strategy would be as follows: as long as the next block to be built is still valid, incorporate as many vertices as possible. To verify the validity of a block we use the following procedures:

Given a k-valid schedule, the *s-divider procedure* generates a $(k+1)$-valid schedule based on the original schedule. This is done by trying to incorporate into the block $k+1$ as many vertices as possible (leaving the remaining vertices in the block $k+2$). If any vertex is "misplaced" in block $k+1$ (lines 12 to 14 of Algorithm 6.3), this vertex is removed from that block.

The process of refinement of block $k+1$ is performed several times until block $k+1$ is stabilized.

If block $k+1$ (set V of Algorithm 6.3) is not empty, then the schedule s' is a $(k+1)$-valid schedule. If the set V is empty, the procedure returns the original schedule, indicating that it is not possible to create a $(k+1)$-valid schedule.

Algorithm 6.3: s-divider(f, h, s, k)

Input: Two disjunctive Boolean networks $f, h : \mathbb{B}^n \to \mathbb{B}^n$ and s a *k-valid* update schedule with $k+1$ blocks.

Output: s' a $(k+1)$-valid update schedule if it exists, or s otherwise.

1 $A^-(h, f) \leftarrow A(h) \setminus A(f)$;
2 $A^+(h, f) \leftarrow (A(h) \cap A(f) \setminus A^{\ominus}(f)$;
3 $s' \leftarrow s$;
4 $V \leftarrow \emptyset$;
5 **forall the** $v \in [n], s(v) = k+1$ **do**
6 $\quad B \leftarrow \{u \in N_f^-(w) : w \in N_h^-(v) \wedge s(w) \leq k\}$;
7 $\quad F \leftarrow \{u \in N_h^-(v) : s(u) > k\}$;
8 $\quad$ **if** $N_f^-(v) = B \cup F$ **then** $V \leftarrow V \cup \{v\}$;
9 **end**
10 **repeat**
11 $\quad$ **for** $v \notin V \wedge s(v) = k+1$ **do** $s'(v) \leftarrow k+2$;
12 $\quad V^{\ominus} \leftarrow \{v \in V : \exists (u, v) \in A^-(h, f), s'(u) \geq s'(v)\}$;
13 $\quad V^{\oplus} \leftarrow \{v \in V : \exists (v, u) \in A^+(h, f), s'(v) < s'(u)\}$;
14 $\quad V^p \leftarrow \{v \in V : \exists (v, u) \in A(f), (v, u) \notin \text{potential}(f, h, s', k+1)\}$;
15 $\quad V \leftarrow V \setminus (V^{\ominus} \cup V^{\oplus} \cup V^p)$;
16 **until** $(V^{\ominus} \cup V^{\oplus} \cup V^p) = \emptyset$;
17 **if** $V \neq \emptyset$ **then return** s';
18 **return** *s;*

The *potential procedure* (Algorithm 6.4) analyzes, given a k-valid update schedule, all possible arcs that would *potentially* create some extension of that update schedule.

Thus, line 14 of Algorithm 6.3 verifies that all arcs of $A(f)$ can be created. If any arc of $A(f)$ does not appear in the potential arcs set, the block $k+1$ that is being analyzed in Algorithm 6.3 must be modified.

Algorithm 6.4: potential(f, h, s, k)

Input: Two disjunctive Boolean networks $f, h : \mathbb{B}^n \to \mathbb{B}^n$, an update schedule s.
Output: A set of arcs that can be generated by an extension of an schedule that contains the first k blocks of s.

`// A receives positive and potentially positive arcs`
1 $A \leftarrow \{(u,v) \in A(h) \cap A(f) : s(u) \geq s(v)\}$;
`// A⁻ receives negative and potentially negative arcs`
2 $A^- \leftarrow \{(u,v) \in A(h) : s(u) < s(v)\} \cup \{(u,v) \in A(h) \cap A^{\ominus}(f) : s(u) = s(v) = k+1\}$;
3 **repeat**
  `// Arcs in A' could be created and appear in h^s`
4 $A' \leftarrow \{(u,v) \notin A : \exists w \in [n], (w,v) \in A^- \wedge (u,w) \in A\}$;
5 $A \leftarrow A \cup A'$;
6 **until** $A' = \emptyset$;
7 **return** A;

Finally, all these procedures are summarized in Algorithm 6.5. The idea in this algorithm is to find an update schedule s such that $h^s = f$. In fact, this algorithm tries to maximize the number of vertices in the first blocks.

Algorithm 6.5: $s(h, f)$

Input: Two disjunctive Boolean networks $f, h : \mathbb{B}^n \to \mathbb{B}^n$.
Output: If there exists an update schedule s such that $h^s = f$, it returns s. Otherwise, returns Null .

1 $s \leftarrow s_p$;
2 $k \leftarrow 0$;
3 **while** $h^s \neq f$ **do**
4 $s' \leftarrow$ s-divider(f, h, s, k);
5 **if** $s' = s$ **then return** *Null*;
6 $k \leftarrow k+1$;
7 $s \leftarrow s'$;
8 **end**
9 **return** s;

The first step is to assume that the parallel schedule is solution for our problem (i.e., $h^s = f$). If it is not, we proceed to divide the schedule into a 2-block schedule (among all possible 2-block schedules, s-divider gives us the 1-valid schedule with the most vertices in the first block).

If this new 2-block schedule is not a solution to the problem, the algorithm calls s-divider again in order to split the second block into two new blocks.

This happens successively until two possible scenarios are reached:

Scenario 1: A schedule obtained by s-divider is a solution to the problem.

Scenario 2: The s-divider procedure is not able to split a k-valid schedule (which is not a solution) into some $(k+1)$-valid schedule, and therefore it is not possible to find any schedule not equivalent to the parallel one that is a solution.

First, let us note that if the algorithm $s(f,h)$ produces an update schedule, it is a solution to the stated problem. We can see that at each entry to the 'while' loop, the algorithm processes a k-valid update schedule. Starting with the parallel schedule, which is 0-valid. At the end of each stage of the loop, what we obtain is a $(k+1)$-valid update schedule, if it exists. Otherwise, the algorithm terminates and returns NULL. To construct the $(k+1)$-valid update schedule, we use the s-divider algorithm, which takes both Boolean networks and a k-valid update schedule as input, and returns a $(k+1)$-valid update schedule if it exists. If it does not exist, it returns the same input update schedule. Note that s-divider returns an update schedule with the minimum possible number of blocks because it places all possible vertices in the new block, ensuring that the update schedule is $(k+1)$-valid and allows, if necessary, the construction of a $(k+2)$-valid update schedule.

Now we must observe that if a solution exists, the algorithm always delivers a solution different from NULL. Remember that an update schedule has an associated arc labeling. Thus, when we have a k-valid update schedule, what we do is fix the labels of the arcs entering and exiting vertices with update function less than or equal to k. This way, the only arcs that can still be labeled differently are the arcs whose starting and ending vertices have an update function greater than k.

When entering s-divider the first thing done is to determine a set V of candidate vertices that could have an update function of $k+1$ in a $(k+1)$-valid update schedule. To do this, we ensure that the input neighborhood of the vertex in network f can be constructed from the input neighborhoods of network h and the proposed update schedule. Then, we begin removing vertices from this set according to three criteria.

The first criterion is that in the proposed update schedule, a positively labeled arc is generated, which we know must be positive according to $A^-(f,h)$. The second criterion is that with the proposed labeling, a negatively labeled arc is generated, which we know must be positive according to $A^+(f,h)$. Lastly, the criterion is that the update schedule prevents the creation of an arc in network f.

Note that removing vertices from the set V based on these three criteria does not affect the fact that the neighborhoods of the remaining vertices can still be constructed. This is because when moving a vertex to block $k+2$, the relationship between the update functions of both vertices remains intact.

In this way, when setting the update function of a vertex v in s-divider to $k+1$, if $(u,v) \in A(h)$ and $s(u) = k+2$ we set $\text{lab}(u,v) = \oplus$. By fixing this positive arc, the only arcs we prevent from being created are those of the form (u,v). However, since we are certain that these arcs were created since v is in V, this does not hinder the generation of any other arc.

On the other hand, if $(v,u) \in A(h)$ and $s(u) = k+2$ we set $\text{lab}(v,u) = \ominus$. In this case, the only arc that may not be created is (v,u). However, the set V^p prevents this from happening.

As mentioned above, Algorithm 6.5 returns an update schedule with the most vertices in the first blocks, so in the case of $h = f$, the answer is a block with all vertices, i.e., the parallel schedule s_p. Such a result is correct, but since what we are looking for is a schedule not equivalent to s_p, we study the following general result that allows to create a strategy slightly different when $h = f$.

Proposition 6.9. *Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network. If there exists s an update schedule with $k > 2$ blocks such that $f^s = f$, then there exists an update schedule s' with two blocks such that $f^{s'} = f$.*

To prove the previous proposition, we use the following lemma:

Lemma 6.10. *Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network. If there exists s an update schedule with $k > 1$ blocks such that $f^s = f$, then there exist at least $k-1$ update schedules with $k-1$ blocks that are also solutions.*

Proof. Let $h = f$ be two Boolean networks and let s be an update schedule with k blocks such that $h^s = f$. If we define the update schedule s' as follows:

$$\forall u \in [n], s'(u) = \begin{cases} s(u) & \text{if } s(u) \leq j \\ s(u) - 1 & \text{if } s(u) > j \end{cases}$$

for some $j \in \{1, \ldots, k\}$, the following occurs:

- $\forall (u, v) \in A(h)$ such that $s(u) = j$ and $s(v) = j + 1$: $\text{lab}_s(u, v) = \ominus$ and $\text{lab}_{s'}(u, v) = \oplus$. The problem is that if there is an arc $(w, v) \in A(h^s)$ that was formed because $lab_s(u, v) = \ominus$. Let us show that $(w, v) \in A(h^{s'})$. Let w be any element in $N_f^-(u) \cap N_f^-(v)$:
  - If $s(w) > (j + 1)$, then the arc (w, v) which was originally formed by (w, u) and (u, v) ($s(w) > j + 1$ and $j < j + 1$), is now formed directly by (w, v) (Since $N_f^-(u) \subseteq N_f^-(v)$ and $s'(w) \geq j + 1$).
  - If $s(w) = j + 1$, since $N_f^-(u) \subseteq N_f^-(v)$ the arc in $G(h^s)$ was already being formed by the arc (w, v) with both vertices in the same block.
  - If $s(w) = j$, then the arc (w, v) which was originally formed by (w, u) and (u, v) ($j = j$ and $j < j + 1$), is now formed directly by (w, v) (Since $N_f^-(u) \subseteq N_f^-(v)$ and $j \geq j + 1$).
  - If $s(w) < j$, then the arc (w, v) was originally formed by $(w, u) \in A(h^s)$ and (u, v). Since $(w, u) \in A(h^s)$, then there exists w' such that $\text{lab}_s(w, w') = \oplus$ and $\text{lab}_s(w', u) = \ominus$. Finally, since $s(w') < s(u)$ and $N_f^-(u) \subseteq N_f^-(v)$, then (w', v) is in $A(f)$ and therefore $(w, v) \in A(h^{s'})$.
- For all other arcs it is satisfied that $\text{lab}_s(u, v) = \text{lab}_{s'}(u, v)$, therefore, they are also in $A(h^{s'})$.

Since, this procedure can be applied to any pair of contiguous blocks, we have $k - 1$ update schedules with $k - 1$ blocks that are solutions. □

Proof of Proposition 6.9. Inductively, based on the fact that $h = f$, we can keep repeating this process until we get an update schedule with 2 blocks. □

With this result, Algorithm 6.6 looks for a 2-block update schedule and give a non-equivalent solution to s_p, if it exists.

6.4 Conclusions and future work

The results detailed in this chapter provide us with valuable insight by evidencing that, despite the inherent complexity of the general case, as demonstrated in the preceding chapter, its resolution in polynomial time is not affected by fixing one of its elements. Contrary to what might be expected,

Algorithm 6.6: $s^{=}(f)$

Input: A disjunctive Boolean network $f : \mathbb{B}^n \to \mathbb{B}^n$.

Output: If there exists an update schedule s (non-equivalent to s_p) such that $f^s = f$, it returns s. Otherwise, returns Null .

if $A(f) \cap A^{\ominus}(f) = \emptyset$ **then return** *Null*;
$V^- \leftarrow \{v \in [n] : \exists (u, v) \in A(f), (u, v) \in A^{\ominus}(f)\}$;
$A^+(f, f) \leftarrow A(f) \setminus A^{\ominus}(f)$;
for $w \in V^-$ **do**
    $s \leftarrow s_p$;
    $V \leftarrow [n] \setminus \{w\}$;
    repeat
        for $v \notin V$ **do** $s(v) \leftarrow 2$;
        $V^{\oplus} \leftarrow \{v \in V : \exists (v, u) \in A^+(f, f), s(v) < s(u)\}$;
        $V^p \leftarrow \{v \in V : \exists (v, u) \in A(f), (v, u) \notin A(f^s)\}$;
        $V \leftarrow V \setminus (V^{\oplus} \cup V^p)$;
    until $(V^{\oplus} \cup V^p) = \emptyset$;
    if $V \neq \emptyset \wedge (\exists (u, v) \in A(f), u \in V \wedge v \notin V)$ **then return** s;
end
return *Null*;

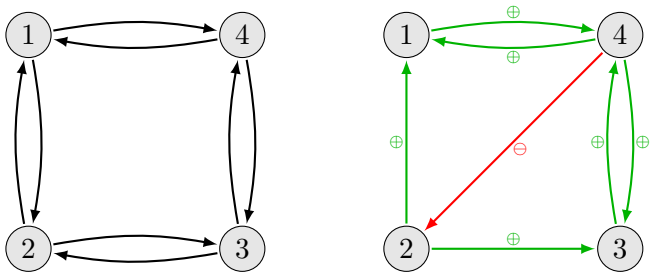
fixing one element not only does not complicate the resolution process, but, remarkably, provides us with solutions that are more intuitive and more accessible to understand. This discovery not only highlights the robustness of the methodology employed, but also opens up new perspectives for tackling related problems.

Additionally, the finding of a monotonicity in the solutions, at least when the update schedule is fixed, grants us the ability to explore the entire spectrum of solutions, allowing us to discover particular situations of relevance. These significant advances stand as the solid foundation on which we can build a deeper analysis of new problems, as in the following example:

Example 6.3. The double cycle network with $n = 4$ fulfill the following conditions:

- There does not exist an update schedule s not equivalent to s_p such that f^s is equivalent to f.
- But there exists a Boolean network h (different from f) with an update schedule s (not equivalent to s_p) such that h^s is equivalent to f (as can be seen in Figure 6.9).

In an additional aspect, the need arises to direct our efforts towards the evaluation of the existence of a monotonicity in the specific case of a fixed Boolean network h. This approach instigates us to investigate whether the presence of this monotonicity can provide a framework that allows us to systematically explore the vast universe of solutions in this particular context. This approach projects a crucial next step in our research, promising to broaden our understanding and contribute to the advancement of complex problem solving in this specific domain.



(a) $G(f)$ (b) $(G(h), \mathrm{lab}_s)$

Figure 6.9: (a) The double cycle network with $n = 4$. (b) A network dynamically equivalent to f with $s = \{1, 3, 4\}\,\{2\}$.

Chapter 7

Conclusions

7.1 English version

In this thesis we have addressed different issues that allow us to satisfy the goal of studying how interaction graphs change when evaluated with different update schedules.

For example, in Chapter 3 various results are presented, such as that the number of strongly connected components of an interaction graph are conserved when obtaining its parallel digraph, independent of the update schedule used to generate it. Furthermore, the transversal number and the packing number, two transcendental parameters when determining upper bounds on the number of fixed points in Boolean networks, were shown to not decrease when obtaining the parallel digraph. Finally, examples of update schedules that maintain the τ of the interaction graph and the ν in complete interaction graphs are presented. These results are not only useful tools for the development of this thesis but also show the importance of update schedules since their appropriate manipulation allows us to control certain properties in parallel digraphs.

In Chapter 4 FixedPoint algorithm is presented. This algorithm allows to find the set of fixed points of a Boolean network based mainly on the structure of the positive cycles of its regulatory graph and to a lesser extent on the size of the network. This can be considered an improvement to the results obtained by [2], since the set of states to test is smaller and, in the case that $\tau = \tau^+$, FixedPoint algorithm works the same way.

On the other hand, FixedPoint algorithm works well in Boolean networks with large values (even unbounded) of in-degree and out-degree as long as the found PFVS is small and the regulatory functions of the network can be evaluated in polynomial time. This is an advantage over others algorithms whose performance depends strongly on the in-degree or out-degree of the network.

The efficiency of FixedPoint algorithm depends mainly on the size of the input PFVS. Due to this, it is important to have a PFVS as close to the minimum as possible. In this sense, although PFVS algorithm can clearly be improved, it is an algorithm that delivers a response close to optimal (since it is bounded by the FVS that contains the PFVS) in polynomial time.

In future work it would be important to explore methods to reduce the size of a network that conserve τ^+. Also, since the efficiency of the FixedPoint algorithm depends on the size of a input PFVS, it is important to study the relationship between τ^+ and other parameters of the regulatory graph of a Boolean network such as: minimum and maximum in-degree, out-degree and degree distributions.

The contribution of these results to our main objective is that the most efficient way to execute this algorithm is to update the vertices in a certain special order, which is clearly associated with a sequential update schedule.

Later, in Chapter 5, different approaches to the problem of dynamically equivalent networks are presented.

As could be seen in this thesis, solving the general problem, i.e., given a Boolean network f, finding another Boolean network $\bar{f}$ and an update schedule $\bar{s}$ such that $\bar{f}^{\bar{s}} = f$ is NP-Hard, since it is as difficult as 3-SAT, but it does present an approach to finding a possible solution: if there exists a solution with an update schedule with more than two blocks, then there exists a solution with an update schedule of only two blocks.

Now, if we restrict the problem to disjunctive networks, finding a disjunctive Boolean network h and an update schedule s such that $h^s = f$, this problem can be solved in polynomial time.

It is worth noting that detecting that in the labeled digraph there exists an arc (u, v) whose label is negative if and only if $N_f^-(u) \subseteq N_f^-(v)$ in the parallel digraph, is a very important result since it implies that any digraph whose neighborhoods are not comparable, has no other dynamically equivalent network different to the trivial one.

With all these results, there remain several ideas to explore, such as finding an algorithm that can solve the general problem, and explore enumeration algorithms, in the case we fix some element of the triplet (h, s, f). Another idea would be to analyze if for a Boolean network f, there exists a Boolean network h and an update schedule s such that s has some particularity (e.g.: s is a sequential update schedule) and that $h^s = f$.

The interesting thing about this chapter is that we can no longer only find an update schedule that fulfills a certain property in the parallel digraph, as we saw in the previous chapters, but we can now build the update schedule that generated the parallel digraph, which opens doors to new research, combined with other areas related to Boolean networks.

Finally, in Chapter 6 the problem of dynamic equivalence between Boolean networks is further explored.

For example, to know if there exists a dynamically equivalent network h to a f and an update schedule s can be decided in polynomial time, moreover the result of the presented algorithm does not give the maximal dynamically equivalent network in the set of arcs, so that such a network we can explore the whole set of possible solutions to the problem with a polynomial delay. With these results, equivalence classes could be defined according to the dynamically equivalent networks.

Moreover, for the case when given h and f one wants to find the update schedule s such that $h^s = f$, it was also shown that it can be decided in polynomial time. Moreover, if $h = f$ an alternative algorithm is also presented that allows to verify if there exist a 2-block update schedule such that $h^s = f$. All these results combined with the results of the previous chapters, allow us to face new challenges, such as combining these results with other families of Boolean networks, or other models, such as cellular automata, in order to build new knowledge.

All these results are interesting new tools that, combined in the right way, would allow to explore a number of topics in the field of Boolean networks that have yet to be explored.

7.2 Versión en español

En esta tesis hemos abordado diferentes cuestiones que nos permiten satisfacer el objetivo de estudiar cómo cambian los grafos de interacción cuando se evalúan con diferentes esquemas de actualización.

Por ejemplo, en el Capítulo 3 se presentan diversos resultados, como que el número de componentes fuertemente conexas de un grafo de interacción se conserva al obtener su digrafo paralelo, independientemente del esquema de actualización utilizado para generarlo. Además, se demues-

tra que el número transversal y el número de empaquetamiento, dos parámetros trascendentales a la hora de determinar las cotas superiores del número de puntos fijos en redes Booleanas, no disminuyen al obtener el digrafo paralelo. Por último, se presentan ejemplos de esquemas de actualización que mantienen el τ del grafo de interacción y el ν en grafos de interacción completos. Estos resultados no sólo son herramientas útiles para el desarrollo de esta tesis sino que muestran la importancia de los esquemas de actualización ya que su adecuada manipulación nos permite controlar ciertas propiedades en los digrafos paralelos.

En el Capítulo 4 se presenta el algoritmo FixedPoint. Este algoritmo permite encontrar el conjunto de puntos fijos de una red Booleana basándose principalmente en la estructura de los ciclos positivos de su grafo regulatorio y en menor medida en el tamaño del grafo. Esto puede considerarse una mejora a los resultados obtenidos por [2], ya que el conjunto de estados a probar es menor y, en el caso de que $\tau = \tau^+$, el algoritmo FixedPoint funciona de la misma manera.

Por otro lado, el algoritmo FixedPoint funciona bien en redes Booleanas con valores grandes (incluso ilimitados) de in-degree y out-degree siempre que el PFVS encontrado sea pequeño y las funciones reguladoras de la red puedan evaluarse en tiempo polinomial. Esto supone una ventaja sobre otros algoritmos cuyo rendimiento depende en gran medida del grado de entrada o de salida de la red.

La eficacia del algoritmo FixedPoint depende principalmente del tamaño del PFVS de entrada. Debido a esto, es importante tener un PFVS lo más cercano posible al mínimo. En este sentido, aunque el algoritmo PFVS es claramente mejorable, se trata de un algoritmo que proporciona una respuesta cercana al óptimo (ya que está acotado por el FVS que contiene al PFVS) en tiempo polinomial.

En futuros trabajos sería importante explorar métodos para reducir el tamaño de una red que conserven τ^+. Además, dado que la eficiencia del algoritmo Fixed Point depende del tamaño de un PFVS de entrada, es importante estudiar la relación entre τ^+ y otros parámetros del grafo regulador de un grafo Booleano como: mínimo y máximo in-degree, out-degree y distribuciones de grado.

La contribución de estos resultados a nuestro objetivo principal es que la forma más eficiente de ejecutar este algoritmo es actualizar los vértices en un cierto orden especial, que está claramente asociado con un esquema de actualización secuencial.

Posteriormente, en el Capítulo 5, se presentan diferentes aproximaciones al problema de las redes dinámicamente equivalentes.

Como se ha podido ver en esta tesis, resolver el problema general, es decir, dado una red Booleana f, encontrar otra red Booleana $\bar{f}$ y un esquema de actualización $\bar{s}$ tal que $\bar{f}^{\bar{s}} = f$ es NP-Hard, ya que es tan difícil como 3-SAT, pero presenta una aproximación para encontrar una posible solución: si existe una solución con un esquema de actualización con más de dos bloques, entonces existe una solución con un esquema de actualización de sólo dos bloques.

Ahora bien, si restringimos el problema a redes disyuntivas, encontrando una red Booleana disyuntiva h y un esquema de actualización s tal que $h^s = f$, este problema puede resolverse en tiempo polinomial.

Cabe destacar que detectar que en el digrafo etiquetado existe un arco (u, v) cuya etiqueta es negativa si y sólo si $N_f^-(u) \subseteq N_f^-(v)$ en el digrafo paralelo, es un resultado muy importante ya que implica que cualquier digrafo cuyos vecindades no sean comparables, no tiene otro grafo dinámicamente equivalente diferente al trivial.

Con todos estos resultados, quedan varias ideas por explorar, como encontrar un algoritmo que pueda resolver el problema general, y explorar algoritmos de enumeración, en el caso de que fijemos algún elemento de la terna (h, s, f). Otra idea sería analizar si para una red Booleana f,

existe una red Booleana h y un esquema de actualización s tal que s tenga alguna particularidad (por ejemplo: s es un esquema de actualización secuencial) y que $h^s = f$.

Lo interesante de este capítulo es que ya no sólo podemos encontrar un esquema de actualización que cumpla una determinada propiedad en el digrafo paralelo, como vimos en los capítulos anteriores, sino que ahora podemos construir el esquema de actualización que generó el digrafo paralelo, lo que abre las puertas a nuevas investigaciones, combinadas con otras áreas relacionadas con las redes Booleanas.

Por último, en el Capítulo 6 se profundiza en el problema de la equivalencia dinámica entre redes Booleanas.

Por ejemplo, saber si existe una red h dinámicamente equivalente a una red f y un esquema de actualización s se puede decidir en tiempo polinomial, además el resultado del algoritmo presentado nos da la máxima red (en el conjunto de arcos) dinámicamente equivalente, por lo que dicha red nos permite explorar todo el conjunto de posibles soluciones al problema con un delay polinomial. Con estos resultados, se podrían definir clases de equivalencia según las redes dinámicamente equivalentes.

Además, para el caso en que dados h y f se quiera encontrar el esquema de actualización s tal que $h^s = f$, también se ha demostrado que se puede decidir en tiempo polinomial. Además, si $h = f$ también se presenta un algoritmo alternativo que permite verificar si existe un esquema de actualización de 2 bloques tal que $h^s = f$. Todos estos resultados combinados con los resultados de los capítulos anteriores, nos permiten afrontar nuevos retos, como combinar estos resultados con otras familias de redes Booleanas, u otros modelos, como los autómatas celulares, para construir nuevo conocimiento.

Todos estos resultados son nuevas e interesantes herramientas que, combinadas de la forma adecuada, permitirían explorar una serie de temas en el campo de las redes Booleanas que aún están por explorar.

7.3 Acknowledgments

Luis Cabrera-Crot is funded by CONICYT-PCHA/Doctorado Nacional/2016-21160885. It was also partially supported by ANID-Chile through Centro de Modelamiento Matemático (CMM), FB210005, BASAL funds for center of excellence from ANID-Chile through ECOS C19E02.

Bibliography

[1] Tatsuya Akutsu, Morihiro Hayashida, and Takeyuki Tamura. Integer programming-based methods for attractor detection and control of Boolean networks. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 5610–5617. IEEE, 2009.

[2] Tatsuya Akutsu, Satoru Kuhara, Osamu Maruyama, and Satoru Miyano. A system for identifying genetic networks from gene expression patterns produced by gene disruptions and overexpressions. *Genome Informatics*, 9:151–160, 1998.

[3] Tatsuya Akutsu, Avraham A Melkman, Takeyuki Tamura, and Masaki Yamamoto. Determining a singleton attractor of a Boolean network with nested canalyzing functions. *Journal of Computational Biology*, 18(10):1275–1290, 2011.

[4] Réka Albert and Hans G Othmer. The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in drosophila melanogaster. *Journal of theoretical biology*, 223(1):1–18, 2003.

[5] Juan A Aledo, Luis G Diaz, Silvia Martinez, and Jose C Valverde. On periods and equilibria of computational sequential systems. *Information Sciences*, 409:27–34, 2017.

[6] Noga Alon. Asynchronous threshold networks. *Graphs and Combinatorics*, 1:305–310, 1985.

[7] Julio Aracena. Maximum number of fixed points in regulatory Boolean networks. *Bulletin of Mathematical Biology*, 70(5):1398–1409, 2008.

[8] Julio Aracena, Florian Bridoux, Luis Gómez, and Lilian Salinas. Complexity of limit cycles with block-sequential update schedules in conjunctive networks. *Natural Computing*, 22:411–429, 2023.

[9] Julio Aracena, Luis Cabrera-Crot, and Lilian Salinas. Finding the fixed points of a Boolean network from a positive feedback vertex set. *Bioinformatics*, 37(8):1148–1155, 2021.

[10] Julio Aracena, Jacques Demongeot, Eric Fanchon, and Marco Montalva. On the number of different dynamics in Boolean networks with deterministic update schedules. *Mathematical biosciences*, 242(2):188–194, 2013.

[11] Julio Aracena, Maximilien Gadouleau, Adrien Richard, and Lilian Salinas. Fixing monotone Boolean networks asynchronously. *Information and Computation*, page 104540, 2020.

[12] Julio Aracena, Eric Goles, Andrés Moreira, and Lilian Salinas. On the robustness of update schedules in Boolean networks. *Biosystems*, 97(1):1–8, 2009.

[13] Julio Aracena, Luis Gómez, and Lilian Salinas. Limit cycles and update digraphs in Boolean networks. *Discrete Applied Mathematics*, 161(1):1–12, 2013.

[14] Julio Aracena, Mauricio González, Alejandro Zuñiga, Marco A Mendez, and Verónica Cambiazo. Regulatory network for cell shape changes during drosophila ventral furrow formation. *Journal of Theoretical Biology*, 239(1):49–62, 2006.

[15] Julio Aracena, Adrien Richard, and Lilian Salinas. Fixed points in conjunctive networks and maximal independent sets in graph contractions. *Journal of Computer and System Sciences*, 88:145–163, 2017.

[16] Julio Aracena, Adrien Richard, and Lilian Salinas. Number of fixed points and disjoint cycles in monotone Boolean networks. *SIAM Journal on Discrete Mathematics*, 31(3):1702–1725, 2017.

[17] Jørgen Bang-Jensen and Gregory Z Gutin. *Digraphs: theory, algorithms and applications*. Springer Science & Business Media, 2008.

[18] Christopher L Barrett, Harry B Hunt, Madhav V Marathe, SS Ravi, Daniel J Rosenkrantz, and Richard E Stearns. Complexity of reachability problems for finite discrete dynamical systems. *Journal of Computer and System Sciences*, 72(8):1317–1345, 2006.

[19] Christopher L Barrett and Christian M Reidys. Elements of a theory of computer simulation i: sequential ca over random graphs. *Applied Mathematics and Computation*, 98(2-3):241–259, 1999.

[20] Florian Bridoux, Pierre Guillon, Kévin Perrot, Sylvain Sené, and Guillaume Theyssier. On the cost of simulating a parallel Boolean automata network by a block-sequential one. In *International Conference on Theory and Applications of Models of Computation*, pages 112–128. Springer, 2017.

[21] Eric Goles Chacc. *Comportement oscillatoire d'une famille d'automates cellulaires non uniformes*. PhD thesis, Institut National Polytechnique de Grenoble-INPG; Université Joseph-Fourier-Grenoble I, 1980.

[22] Madalena Chaves, Reka Albert, and Eduardo D Sontag. Robustness and fragility of Boolean models for genetic regulatory networks. *Journal of Theoretical Biology*, 235(3):431–449, 2005.

[23] Jianer Chen, Yang Liu, Songjian Lu, Barry O'sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *Journal of the ACM (JACM)*, 55(5):21, 2008.

[24] Omar Colón-Reyes, Reinhard Laubenbacher, and Bodo Pareigis. Boolean monomial dynamical systems. *Annals of Combinatorics*, 8(4):425–439, 2005.

[25] Maria I Davidich and Stefan Bornholdt. Boolean network model predicts cell cycle sequence of fission yeast. *PloS one*, 3(2):e1672, 2008.

[26] Jacques Demongeot, Adrien Elena, and Sylvain Sené. Robustness in regulatory networks: a multi-disciplinary approach. *Acta Biotheoretica*, 56(1-2):27–49, 2008.

[27] Vincent Devloo, Pierre Hansen, and Martine Labbé. Identification of all steady states in large networks by logical analysis. *Bulletin of mathematical biology*, 65(6):1025–1051, 2003.

[28] Elena Dubrova and Maxim Teslenko. A sat-based algorithm for finding attractors in synchronous boolean networks. *IEEE/ACM transactions on computational biology and bioinformatics*, 8(5):1393–1399, 2011.

[29] Adrien Fauré, Aurélien Naldi, Claudine Chaouiya, and Denis Thieffry. Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. *Bioinformatics*, 22(14):e124–e131, 2006.

[30] Patrik Floréen and Pekka Orponen. On the computational complexity of analyzing hopfield nets. *Complex Systems*, 1989.

[31] Eric Goles, Marco Montalva-Medel, Henning Mortveit, and Salvador Ramirez-Flandes. Block invariance in elementary cellular automata. *Journal of Cellular Automata*, 10, 2015.

[32] Eric Goles, Pedro Montealegre, Ville Salo, and Ilkka Törmä. Pspace-completeness of majority automata networks. *Theoretical Computer Science*, 609:118–128, 2016.

[33] Eric Goles and Mathilde Noual. Block-sequential update schedules and Boolean automata circuits. In *Automata 2010-16th Intl. Workshop on CA and DCS*, pages 41–50. Discrete Mathematics and Theoretical Computer Science, 2010.

[34] Eric Goles and Mathilde Noual. Disjunctive networks and update schedules. *Advances in Applied Mathematics*, 48(5):646–662, 2012.

[35] Eric Goles and Gonzalo A Ruz. Dynamics of neural networks over undirected graphs. *Neural Networks*, 63:156–169, 2015.

[36] Eric Goles and Lilian Salinas. Comparison between parallel and serial dynamics of Boolean networks. *Theoretical Computer Science*, 396(1):247–253, 2008.

[37] Eric Goles and Lilian Salinas. Sequential operator for filtering cycles in Boolean networks. *Advances in Applied Mathematics*, 45(3):346–358, 2010.

[38] Eric Goles-Chacc, Françoise Fogelman-Soulié, and Didier Pellegrin. Decreasing energy functions as a tool for studying threshold networks. *Discrete Applied Mathematics*, 12(3):261–277, 1985.

[39] Zhiwei He, Meng Zhan, Shuai Liu, Zebo Fang, and Chenggui Yao. An algorithm for finding the singleton attractors and pre-images in strong-inhibition Boolean networks. *PloS one*, 11(11):e0166906, 2016.

[40] Tomáš Helikar, John Konvalina, Jack Heidel, and Jim A Rogers. Emergent decision-making in biological signal transduction networks. *Proceedings of the National Academy of Sciences*, 105(6):1913–1918, 2008.

[41] Franziska Hinkelmann, Madison Brandon, Bonny Guang, Rustin McNeill, Grigoriy Blekherman, Alan Veliz-Cuba, and Reinhard Laubenbacher. Adam: analysis of discrete models of biological systems using computer algebra. *BMC bioinformatics*, 12(1):295, 2011.

[42] Sui Huang. Gene expression profiling, genetic networks, and cellular states: an integrating concept for tumorigenesis and drug discovery. *Journal of Molecular Medicine*, 77(6):469–480, 1999.

[43] Abdul Salam Jarrah, Reinhard Laubenbacher, and Alan Veliz-Cuba. The dynamics of conjunctive and disjunctive Boolean network models. *Bulletin of Mathematical Biology*, 72(6):1425–1447, 2010.

[44] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

[45] S. A. Kauffman. Metabolic stability and epigenesis in randomly connected nets. *Journal of Theoretical Biology*, 22:437–467, 1969.

[46] Stuart A Kauffman. *The origins of order: Self-organization and selection in evolution*. Oxford University Press, USA, 1993.

[47] Laleh Kazemzadeh, Marija Cvijovic, and Dina Petranovic. Boolean model of yeast apoptosis as a tool to study yeast and human apoptotic regulations. *Frontiers in physiology*, 3:446, 2012.

[48] Steffen Klamt, Julio Saez-Rodriguez, Jonathan A Lindquist, Luca Simeoni, and Ernst D Gilles. A methodology for the structural and functional analysis of signaling and regulatory networks. *BMC bioinformatics*, 7(1):56, 2006.

[49] Koichi Kobayashi. Design of fixed points in Boolean networks using feedback vertex sets and model reduction. *Complexity*, 2019, 2019.

[50] Fangting Li, Tao Long, Ying Lu, Qi Ouyang, and Chao Tang. The yeast cell-cycle network is robustly designed. *Proceedings of the National Academy of Sciences*, 101(14):4781–4786, 2004.

[51] Anna Lovrics, Yu Gao, Bianka Juhász, István Bock, Helen M Byrne, András Dinnyés, and Krisztián A Kovács. Boolean modelling reveals new regulatory connections between transcription factors orchestrating the development of the ventral spinal cord. *PloS one*, 9(11):e111430, 2014.

[52] Alex Madrahimov, Tomáš Helikar, Bryan Kowal, Guoqing Lu, and Jim Rogers. Dynamics of influenza virus and human host interactions during infection and replication cycle. *Bulletin of mathematical biology*, 75(6):988–1011, 2013.

[53] Avraham A Melkman, Takeyuki Tamura, and Tatsuya Akutsu. Determining a singleton attractor of an AND/OR Boolean network in o (1.587 n) time. *Information Processing Letters*, 110(14-15):565–569, 2010.

[54] Luis Mendoza and Ioannis Xenarios. A method for the generation of standardized qualitative dynamical systems of regulatory networks. *Theoretical Biology and Medical Modelling*, 3(1):13, 2006.

[55] Marco Montalva, Julio Aracena, and Anahí Gajardo. On the complexity of feedback set problems in signed digraphs. *Electronic Notes in Discrete Mathematics*, 30:249–254, 2008.

[56] Henning Mortveit and Christian Reidys. *An introduction to sequential dynamical systems*. Springer Science & Business Media, 2007.

[57] Henning S Mortveit. Limit cycle structure for block-sequential threshold systems. In *International Conference on Cellular Automata*, pages 672–678. Springer, 2012.

[58] Henning S Mortveit and Christian M Reidys. Discrete, sequential dynamical systems. *Discrete Mathematics*, 226(1-3):281–295, 2001.

[59] Aurélien Naldi, Céline Hernandez, Nicolas Levy, Gautier Stoll, Pedro T Monteiro, Claudine Chaouiya, Tomáš Helikar, Andrei Zinovyev, Laurence Calzone, Sarah Cohen-Boulakia, et al. The colomoto interactive notebook: accessible and reproducible computational analyses for qualitative biological networks. *Frontiers in physiology*, 9:680, 2018.

[60] Kévin Perrot. *Études de la complexité algorithmique des réseaux d'automates*. PhD thesis, Aix-Marseille Université, 2022.

[61] Sobia Raza, Kevin A Robertson, Paul A Lacaze, David Page, Anton J Enright, Peter Ghazal, and Tom C Freeman. A logic-based diagram of signalling pathways central to macrophage activation. *BMC systems biology*, 2(1):36, 2008.

[62] CM Reidys. Acyclic orientations of random graphs. *Advances in Applied Mathematics*, 21(2):181–192, 1998.

[63] Élisabeth Remy, Paul Ruet, and Denis Thieffry. Graphic requirements for multistability and attractive cycles in a Boolean dynamical framework. *Advances in Applied Mathematics*, 41(3):335–350, 2008.

[64] Adrien Richard. Fixed points and connections between positive and negative cycles in Boolean networks. *Discrete Applied Mathematics*, 243:1–10, 2018.

[65] Adrien Richard. Positive and negative cycles in Boolean networks. *Journal of theoretical biology*, 463:67–76, 2019.

[66] Søren Riis. Utilising public information in network coding. *General Theory of Information Transfer and Combinatorics*, 4123:866–897, 2006.

[67] F. Robert. *Discrete iterations: a metric study*, volume 6 of *Series in Computational Mathematics*. Springer-Verlag, Berlin, 1986.

[68] François Robert. *Les systemes dynamiques discrets*, volume 19. Springer Science & Business Media, 1995.

[69] Neil Robertson, Paul D Seymour, and Robin Thomas. Permanents, pfaffian orientations, and even directed circuits. *Annals of Mathematics*, 150(3):929–975, 1999.

[70] Daniel Rolf. Improved bound for the ppsz/schöning-algorithm for 3-sat. *Journal on Satisfiability, Boolean Modeling and Computation*, 1:111–122, 2006.

[71] Assieh Saadatpour, Rui-Sheng Wang, Aijun Liao, Xin Liu, Thomas P Loughran, István Albert, and Réka Albert. Dynamical and structural analysis of a t cell survival network identifies novel candidate therapeutic targets for large granular lymphocyte leukemia. *PLoS computational biology*, 7(11), 2011.

[72] Julio Saez-Rodriguez, Luca Simeoni, Jonathan A Lindquist, Rebecca Hemenway, Ursula Bommhardt, Boerge Arndt, Utz-Uwe Haus, Robert Weismantel, Ernst D Gilles, Steffen Klamt, et al. A logical model provides insights into t cell receptor signaling. *PLoS computational biology*, 3(8), 2007.

[73] Regina Samaga, Julio Saez-Rodriguez, Leonidas G Alexopoulos, Peter K Sorger, and Steffen Klamt. The logic of egfr/erbb signaling: theoretical properties and analysis of high-throughput data. *PLoS computational biology*, 5(8), 2009.

[74] Yara-Elena Sanchez-Corrales, Elena R Alvarez-Buylla, and Luis Mendoza. The arabidopsis thaliana flower organ specification gene regulatory network determines a robust differentiation process. *Journal of theoretical biology*, 264(3):971–983, 2010.

[75] Amit Singh, Juliana M Nascimento, Silke Kowar, Hauke Busch, and Melanie Boerries. Boolean approach to signalling pathway modelling in HGF-induced keratinocyte migration. *Bioinformatics*, 28(18):i495–i501, 2012.

[76] Takeyuki Tamura and Tatsuya Akutsu. Algorithms for singleton attractor detection in planar and nonplanar AND/OR Boolean networks. *Mathematics in Computer Science*, 2(3):401–420, 2009.

[77] Takeyuki Tamura and Tatsuya Akutsu. Detecting a singleton attractor in a Boolean network utilizing sat algorithms. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 92(2):493–501, 2009.

[78] René Thomas. Boolean formalization of genetic control circuits. *Journal of Theoretical Biology*, 42(3):563–585, 1973.

[79] René Thomas and Richard d'Ari. *Biological feedback*. CRC press, 1990.

[80] Alan Veliz-Cuba. Reduction of Boolean network models. *Journal of theoretical biology*, 289:167–172, 2011.

[81] Alan Veliz-Cuba, Boris Aguilar, Franziska Hinkelmann, and Reinhard Laubenbacher. Steady state analysis of Boolean molecular network models via model reduction and computational algebra. *BMC bioinformatics*, 15(1):221, 2014.

[82] Alan Veliz-Cuba, Boris Aguilar, and Reinhard Laubenbacher. Dimension reduction of large sparse and-not network models. *Electronic Notes in Theoretical Computer Science*, 316:83–95, 2015.

[83] Lijian Yang, Yan Meng, Chun Bao, Wangheng Liu, Chengzhang Ma, Anbang Li, Zhan Xuan, Ge Shan, and Ya Jia. Robustness and backbone motif of a cancer network regulated by mir-17-92 cluster during the g1/s transition. *PloS one*, 8(3):e57009, 2013.

[84] Jorge GT Zañudo and Réka Albert. An effective network reduction approach to find the dynamical repertoire of discrete dynamic networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 23(2):025111, 2013.

[85] Shu-Qin Zhang, Morihiro Hayashida, Tatsuya Akutsu, Wai-Ki Ching, and Michael K Ng. Algorithms for finding small attractors in Boolean networks. *EURASIP Journal on Bioinformatics and Systems Biology*, 2007(1):20180, 2007.

[86] Yi Ming Zou. An algorithm for detecting fixed points of Boolean network. In *2013 ICME International Conference on Complex Medical Engineering*, pages 670–673. IEEE, 2013.