



Universidad de Concepción
Dirección de Postgrado
Facultad de Ingeniería - Programa de Magíster en Ciencias de la Computación

Esquemas parciales de actualización en redes Booleanas



Tesis para optar al grado de Magíster en Ciencias de la Computación

EDUARDO ALFONSO PALMA ÁVILA
CONCEPCIÓN-CHILE
2015

Profesor Guía: Lilian Salinas Ayala
Julio Aracena Lucero
Dpto. de Ingeniería Informática y Ciencias de la Computación, Facultad de Ingeniería
Universidad de Concepción

Resumen

Las redes Booleanas, originalmente introducidas por Kauffman [16, 17], son el modelo más simple para representar redes regulatorias génicas. A pesar de su simplicidad ellas proveen un marco teórico en el cual diferentes fenómenos pueden ser reproducidos y estudiados. Muchos modelos regulatorios publicados en la literatura del área de la biología calzan dentro de este marco [26].

Las redes cuentan con un esquema de actualización, que determina cuál es el orden en que se activan o inhiben los genes. En relación a esto, en ocasiones se estará en presencia de etiquetados parciales que indican un orden parcial. En estos casos interesa determinar los esquemas de actualización que cumplen con dicho orden.

En esta tesis se pretende estudiar analítica y algorítmicamente el conjunto de extensiones de un etiquetado parcial que hacen que el digrafo resultante sea de actualización. Esto representa un importante avance hacia la búsqueda de esquemas que preservan una propiedad dinámica; como, por ejemplo, un ciclo límite dado, pero no necesariamente todo el comportamiento dinámico de una red.

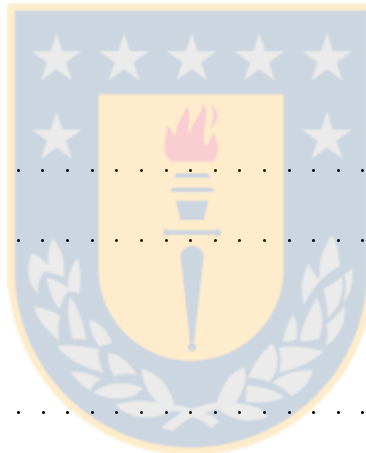
Agradecimientos

Agradezco a CONICYT por financiar mis estudios de posgrado por medio de la beca de Magister nacional. También por haber financiado el Proyecto FONDECYT 1131013 “Limit Cycles and Deterministic Update in Boolean Networks” en el cual se enmarca esta tesis.



Índice general

1	Introducción	1
2	Marco Teórico	5
2.1	Definiciones y notación	5
2.2	Trabajos previos	10
3	Problema	12
3.1	Complejidad	12
4	Algoritmos	16
4.1	Verifica	16
4.1.1	Caminos reversos	17
4.2	Forzar	18
4.2.1	Propiedades	20
4.2.2	Extensión maximal	20
4.3	Reducción del digrafo	22
4.4	Dividir para conquistar	25
4.4.1	División por puentes	25



4.4.2	Dividir por componentes fuertemente conexas	26
4.5	Algoritmo principal	27
4.6	Ejemplos de casos extremos	34
4.6.1	Digrafos Completos	34
4.6.2	Cadenas bidireccionales no circulares	35
4.7	Efectos en la salida frente a cambios en la entrada del algoritmo	36
4.7.1	Agregar una etiqueta a un arco sin etiqueta	37
4.7.2	Agregar un nuevo arco etiquetado	37
4.7.3	Cambiar una etiqueta \oplus por \ominus o viceversa	38
5	Aplicaciones	39
5.1	Implementación	39
5.2	Red del ciclo celular mamífero	40
5.3	Red del ciclo celular de la levadura de fisión	43
6	Conclusiones	47
	Bibliografía	50
7	Anexo	52
7.1	Algoritmos	52
7.1.1	Floyd - Warshall	52
7.1.2	Tarjan	53



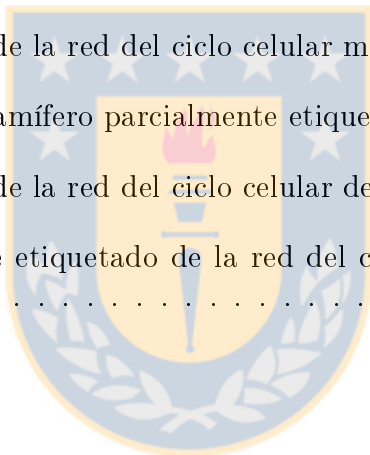
Lista de Tablas

4.1	Definición del operador \odot	17
4.2	Resumen de resultados sobre digrafos completos	35
4.3	Resumen de resultados sobre cadenas bidireccionales no circulares	36
5.1	Notación para los nodos de la red del ciclo celular mamífero (G_M).	40
5.2	Funciones de activación local de la red del ciclo celular mamífero.	41
5.3	Atractores de la red del ciclo celular mamífero sincrónicamente actualizada.	42
5.4	Notación para los nodos de red del ciclo celular de la levadura de fisión.	44
5.5	Funciones lógicas de la red de ciclo celular de la levadura de fisión.	44
5.6	Ciclo límite de la red del ciclo celular de la levadura de fisión actualizada sincrónicamente.	45
5.7	Tabla resumen de resultados.	46

Índice de figuras

2.1	Digrafo de actualización y etiquetado asociados a una red Booleana y esquema de actualización	7
2.2	Digrafo reverso	8
2.3	Ejemplos camino reverso y camino reverso negativo	9
2.4	Ejemplo de digrafo etiquetado que no es de actualización y ciclo prohibido	10
3.1	Construcción de orientación acíclica G' a partir de un grafo G	13
3.2	Equivalencia entre orientación acíclica G_0 y el digrafo de actualización (G', lab_{G_0})	14
4.1	Ejemplo matriz de caminos reversos	18
4.2	Ejemplos de arcos forzados	19
4.3	Reducción de un digrafo etiquetado	23
4.4	División por puentes	26
4.5	Ejemplo división por componentes fuertes conexas	27
4.6	a) Archivo de entrada. b) Digrafo etiquetado (G, lab) asociado a la entrada.	30
4.7	Arcos etiquetados inicialmente en (G, lab)	30
4.8	Grafo fundamental de la entrada	30
4.9	Subdigrafos de la división por puentes	31
4.10	Soluciones del subdigrafo inducido por los nodos 3 y 4.	31

4.11	Soluciones del subdigrafo inducido por los arcos 0, 1 y 2.	31
4.12	División por CFC en el ejemplo	32
4.13	Solución de subdigrafo después de división por CFC	32
4.14	Arco forzado en subproblema	32
4.15	El conjunto $\mathcal{S}(G, \text{lab})$ del ejemplo	33
4.16	Archivo de salida del ejemplo	34
4.17	Ejemplo del digrafo completo de $n = 4$ (K_4).	35
4.18	Ejemplo del digrafo cadena bidireccional $n = 5$	36
5.1	a) Archivo de entrada. b) Digrafo etiquetado (G, lab) asociado a la entrada.	39
5.2	Digrafo de interacción de la red del ciclo celular mamífero.	41
5.3	Red del ciclo celular mamífero parcialmente etiquetado (G_M, lab_M)	43
5.4	Digrafo de interacción de la red del ciclo celular de la levadura de fisión.	45
5.5	El digrafo parcialmente etiquetado de la red del ciclo celular de la levadura de fisión (G_F, lab_F)	46



Capítulo 1

Introducción

Una red Booleana es un sistema de n variables Booleanas interactuando, el cual evoluciona en un tiempo discreto de acuerdo a una regla de regulación y a un esquema de actualización predefinido. La estructura de una red puede ser representada por un digrafo, llamado grafo de interacción, donde los vértices son las componentes de la red, y donde hay un arco de una componente a otra cuando la evolución de una depende de la otra. Cada arco del digrafo puede ser etiquetado según el orden relativo definido por el esquema de actualización. El digrafo de interacción etiquetado de esta forma es conocido como digrafo de actualización.

Las redes Booleanas tienen muchas aplicaciones. En particular, en los trabajos de Kauffman [16, 17] y Thomas [26, 27]; son usadas como modelos de redes génicas. En este contexto los atractores son asociados a distintos tipos de células definidas por patrones de actividad génica. Por ejemplo, los ciclos límites son asociados a los ciclos mitóticos en las células [3, 15, 16].

En el esquema original de las redes Booleanas todos los nodos son actualizados en cada intervalo de tiempo en paralelo (también llamada actualización sincrónica). Otro esquema conocido en la literatura es el secuencial donde cada nodo se actualiza en una secuencia definida en cada intervalo de tiempo. Un esquema más general, introducido por Robert en [22] y conocido como esquema bloque secuencial, consiste en considerar que el conjunto de nodos de la red está particionado en bloques ordenados según una secuencia dada y que los nodos pertenecientes a un bloque particular se actualizan en simultáneo. Esto generaliza los casos previos porque el paralelo corresponde a tener un solo bloque y el secuencial a bloques de tamaño uno.

Las diferencias en los comportamientos dinámicos de las redes Booleanas con distintos esquemas de actualización, se han estudiado desde el punto de vista experimental y estadístico principalmente [4, 6, 7, 9]. Más recientemente, Schmal et al. han estudiado en [24] las redes Booleanas que siguen una trayectoria confiable en el espacio de estados, la cual puede ser robusta contra perturbaciones en el esquema de actualización. Por otro lado, existen solo unos pocos estudios analíticos de este tema [12, 23]. Aracena et al. en [1] estudiaron el número de comportamientos dinámicos posibles de una red Booleana con diferentes esquemas de actuali-

zación. Goles y Salinas han hecho un análisis comparativo en [13] sobre los atractores en redes Booleanas con esquemas de actualización paralelos y secuenciales. Algunos trabajos analíticos sobre los esquemas de actualización han estudiado una clase especial de redes dinámicas discretas, llamados sistemas dinámicos secuenciales. Para esta clase de redes, Mortveit y Reidys han estudiado en [20] el conjunto de esquemas de actualización secuencial que preservan todo el comportamiento dinámico, mientras que Hansson et al. han hecho lo análogo en [14] para el conjunto de atractores en ciertas clases de autómatas celulares.

Se dice que dos esquemas de actualización son equivalentes o que pertenecen a una misma clase de equivalencia, si producen el mismo comportamiento dinámico. En [2, 23] se demostró que si dos esquemas tienen asociados el mismo digrafo etiquetado (digrafo de actualización), entonces son equivalentes.

Dado a que el número de configuraciones posibles de una red Booleana es finito, el comportamiento límite de las trayectorias de una red iterada contiene una secuencia periódica de configuraciones. Cuando el largo de esta secuencia es mayor que uno, se está en presencia de un ciclo límite, en caso contrario se hablará de un punto fijo. Estos últimos permanecen invariantes respecto al esquema de actualización, sin embargo, los ciclos límites de una red Booleana son muy sensibles a los cambios en el esquema. Este es un motivo importante para centrar el estudio en los diversos esquemas de actualización. Los ciclos límite y puntos fijos son llamados atractores.

Muchas veces el orden relativo en que se actualizan los nodos se conoce parcialmente, por ejemplo por falta de información o la imposición de condiciones necesarias para observar una propiedad dinámica de la red. Esto define el digrafo parcialmente etiquetado.

En esta tesis se trabaja con digrafos parcialmente etiquetados. En particular, se estudia analítica y algorítmicamente el problema de enumerar el conjunto de digrafos totalmente etiquetados que son compatibles con el etiquetado parcial y que son digrafos de actualización (UDEP).

La tesis comienza en el Capítulo 2 definiendo conceptos e introduciendo la notación que es utilizada durante toda la investigación. También se presentan resultados importantes que fueron usados en el trabajo y que corresponden a estudios previos encontrados en la literatura.

Lo primero que se estudia es la complejidad de UDEP. Esta es esencial para abordar el problema y conocer las limitaciones teóricas de un algoritmo que lo resuelva. Es por esto que en el Capítulo 3 se encuentra la definición del problema general y su problema de conteo asociado. Se demuestra que existe una reducción polinomial del problema de las orientaciones acíclicas en un grafo [18] al problema de conteo asociado a UDEP. Por lo tanto, es posible afirmar que este último pertenece a la clase $\#P$ -completo, y por ende UDEP es un problema difícil de tratar computacionalmente.

En el Capítulo 4 se presentan y justifican teóricamente los algoritmos que permiten resolver UDEP. Se comienza mostrando el algoritmo Verificar que permite identificar si el digrafo etiquetado ingresado corresponde a uno de actualización. Dentro de la misma sección se presenta una adaptación del algoritmo Floyd-Warshall, que permite encontrar los caminos reversos y reversos

negativos dentro de un digrafo etiquetado. La cual es útil para el algoritmo Verificar y el algoritmo Forzar. Justamente se continua el capítulo formalizando la idea más importante a la hora de acortar el espacio de búsqueda, la cual consiste en forzar arcos. Esta está basada en utilizar los conceptos de camino reverso y reverso negativo, y su relación con los ciclos prohibidos para no explorar soluciones no válidas.

En la Sección 4.3 se presenta el algoritmo Reducir que permite obtener el digrafo reducido de uno etiquetado. Esto se hace utilizando el algoritmo de Tarjan para componentes fuertemente conexas [25]. Este proceso permite disminuir en ocasiones la cantidad de nodos y arcos con la que se trabaja. Otra herramienta utilizada en el diseño del algoritmo para resolver el problema, consiste en dividir este en subproblemas. Tomando en cuenta lo realizado en [21] se consideran divisiones del digrafo, mediante la detección de puentes en el grafo fundamental asociado y componentes fuertemente conexas sobre el digrafo reverso extendido, que entrega el mismo número de soluciones de una manera más compacta. Finalmente, se presenta el algoritmo principal que encuentra la solución a UDEP llamado EtiquetadosUpdate que reduce, verifica y fuerza antes de llamar al algoritmo Etiquetar que recursivamente busca cada una de las extensiones totales que hacen que el digrafo sea de actualización.

En la Sección 4.6, se aplica la implementación del algoritmo EtiquetadosUpdate en los digrafos completos que son los que entregan un mayor número de soluciones para un $n := |V(G)|$ dado. Los resultados obtenidos para este último son muy buenos en cuanto al tiempo de ejecución. No obstante, pese a la compresión de datos por las divisiones los archivos con los etiquetados empiezan a crecer rápidamente obteniéndose tamaños cada vez más inmanejables. Como ejemplo el digrafo completo de 11 nodos entrega un archivo de salida de 1,3 GB, aunque en menos de 664 segundos.

En la Sección 4.7 se estudia la robustez del algoritmo EtiquetadosUpdate, es decir, cómo se ven afectados los resultados que entrega este respecto a modificaciones en la entrada. Específicamente se analizan las situaciones de: etiquetar uno sin etiqueta, agregar un arco etiquetado y cambiar un arco etiquetado por el signo contrario. El análisis consiste en ideas para aprovechar la respuesta entregada por el algoritmo y no calcular todo de nuevo dentro de lo posible. Al final se presentan ideas para hacer que el algoritmo sea más robusto que el actual.

Finalmente, en el Capítulo 5 se explican detalles importantes de la implementación del algoritmo y se ejecutan pruebas sobre la red del ciclo celular mamífero [10, 8] y la del ciclo celular de la levadura de fisión [5, 11]. En ambos casos se describe la red y se presenta su único ciclo límite. Después, se resuelve UDEP para el digrafo sin etiquetas y también para el digrafo parcialmente etiquetado, luego de considerar restricciones que preservan el ciclo límite respectivo. Con los resultados obtenidos se llega a la conclusión de que hay un gran ahorro tanto en tiempo como en el espacio de búsqueda al etiquetar algunos arcos. En el caso de la red del ciclo celular mamífero se pasa de 466 712 a 1 440 soluciones, mientras que en la del ciclo celular de la levadura de fisión de 80 080 a 288 posibles esquemas no equivalentes candidatos a preservar el ciclo. Es relevante hacer notar que el proceso de etiquetado previo debido a las restricciones se hace con un trabajo muy simple, es decir, que es muy poco el costo de realizarlo en comparación con la reducción de candidatos. Posteriormente, se tomó un esquema como representante por cada etiquetado candidato y se iteró sobre la red para comprobar cuales de estos preservaban

verdaderamente el ciclo.



En este capítulo se presentan los conceptos y la notación que se utilizará en esta Tesis. Se mencionan también resultados importantes de trabajos previos realizados en otras investigaciones relacionadas al tema de interés.

2.1 Definiciones y notación

En esta sección se presenta la notación y todas las definiciones que son utilizada en esta tesis. Estas han sido tomadas principalmente de la literatura, aunque también se han creado nuevos conceptos útiles para este trabajo.

Un *grafo* consiste de un conjunto finito y no vacío de elementos llamados vértices o nodos, y un conjunto finito de pares no ordenados llamados aristas. Dado un grafo $G = (V, E)$ se hará referencia a sus vértices por $V(G)$ y a sus aristas por $E(G)$.

Un *digrafo* $G = (V, A)$ se diferencia de un grafo en que, en vez de un conjunto de aristas, posee un conjunto finito de pares ordenados de vértices llamados arcos. Estos son referenciados por $A(G)$; y para todo arco (u, v) , u es llamado nodo de entrada y v de salida.

Se llama *grafo fundamental* de un digrafo G al grafo G_U , donde:

$$\begin{aligned} V(G_U) &= V(G) \\ E(G_U) &= \{uv \mid (u, v) \in A(G) \vee (v, u) \in A(G)\} \end{aligned}$$

Sea $G = (V, A)$ un digrafo y $V' \subseteq V$, se define $G[V']$ como el *subdigrafo inducido* por V' , es decir, aquel donde $V(G[V']) = V'$ y $A(G[V']) = \{(u, v) \in A(G) \mid u, v \in V'\}$.

Dado un digrafo G , para cada $v \in V(G)$ el conjunto de nodos que entran a v es denotado por

$$N_G^-(v) = \{u \in V(G) : (u, v) \in A(G)\}.$$

Análogamente, el conjunto de nodos que salen de v es denotado por

$$N_G^+(v) = \{u \in V(G) : (v, u) \in A(G)\}.$$

Un arco $(v, v) \in A(G)$ es llamado *bucle* de G .

Se dice que hay un *camino dirigido* de u a v , dos vértices de un digrafo G , si existe una secuencia de vértices (w_1, \dots, w_l) donde $\forall i \in 1, \dots, l-1 : (w_i, w_{i+1}) \in A(G)$ con $w_1 = u$ y $w_l = v$.

Un digrafo *fuertemente conexo* G es uno donde $\forall u, v \in V(G)$ existe un camino dirigido de u a v .

Las *componentes fuertemente conexas (CFC)* de un digrafo son subdigrafos inducidos fuertemente conexos maximales, donde la condición de maximal indica que al agregar un vértice cualquiera al subdigrafo mencionado este deja de ser fuertemente conexo. Los vértices de estos subdigrafos forman una partición de vértices del digrafo.

Una *red Booleana* $N = (F, s)$ está definida por un conjunto finito V de n elementos; n variables de estado $x_v \in \{0, 1\}$, $v \in V$; una función $F = (f_v)_{v \in V} : \{0, 1\}^n \rightarrow \{0, 1\}^n$ llamada *función de activación global*, la cual está compuesta por funciones $f_v : \{0, 1\}^n \rightarrow \{0, 1\}$ llamadas *funciones de activación locales*; y un *esquema de actualización* definido por una función $s : V \rightarrow \{1, \dots, n\}$ tal que $s(V) = \{1, \dots, m\}$ para algún $m \leq n$. Un bloque de s es un conjunto $B_i = \{v \in V : s(v) = i\}$, $1 \leq i \leq m$. Si s tiene solamente un bloque, entonces se dice que es un esquema de actualización paralelo. En este caso, se escribirá $s = s_p$. Si s es una permutación sobre el conjunto $\{1, \dots, n\}$, se dirá que es un esquema de actualización secuencial. En cualquier otro caso, s será un esquema de actualización bloque secuencial.

La iteración de una red Booleana con una función de actualización s está dada por:

$$x_i^{r+1} = f_i(x_1^{l_1}, \dots, x_n^{l_n}),$$

donde $l_j = r$ si $s(i) \leq s(j)$ y $l_j = r + 1$ si $s(i) > s(j)$.

Esto es equivalente a aplicar la función $F^s : \{0, 1\}^n \rightarrow \{0, 1\}^n$ de forma paralela, con $F^s(x) = (f_1^s(x), \dots, f_n^s(x))$ definida por:

$$f_i^s(x) = f_i(g_{i,1}^s(x), \dots, g_{i,n}^s(x)),$$

donde la función $g_{i,j}^s$ es definida por $g_{i,j}^s(x) = x_j$ si $s(i) \leq s(j)$ y $g_{i,j}^s(x) = f_j^s(x)$ si $s(i) > s(j)$. Por lo que F^s corresponde al comportamiento dinámico de $N = (F, s)$.

El digrafo asociado a la función $F = (f_v)_{v \in V}$, llamado *digrafo de interacción*, es el digrafo dirigido $G^F = (V, A)$, donde $(u, v) \in A(G)$ si y solo si f_v depende de x_u ; es decir, si existe $x \in \{0, 1\}^n$ tal que $f_v(x) \neq f_v(\bar{x}^u)$, con \bar{x}^u diferente de x solamente en la posición u . Note que

si f_v es constante, entonces $N_{G^F}^-(v) = \emptyset$. En la Figura 2.1a se observa un ejemplo de digrafo de interacción. Notar que x_i hace referencia al estado del nodo i como se definió previamente.

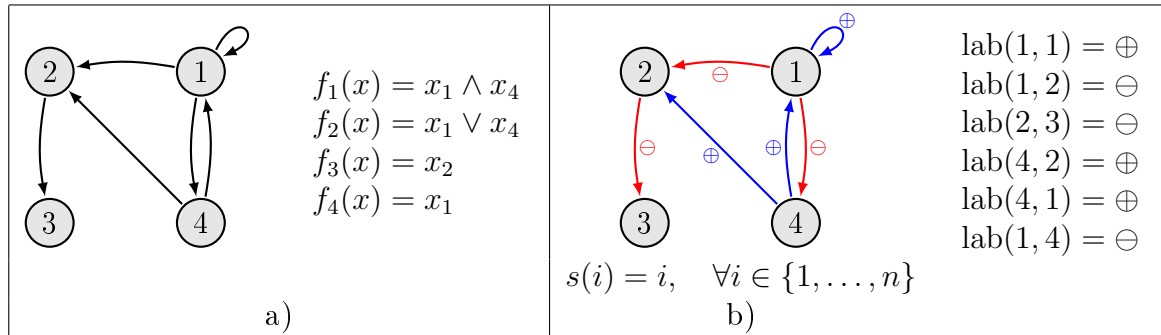


Figura 2.1: a) Digrafo asociado a la red Booleana. b) Digrafo etiquetado asociado a la red Booleana y el esquema de actualización.

Dado un digrafo G , una *función de etiqueta* es cualquier función $\text{lab} : A(G) \rightarrow \{\oplus, \ominus, \circ\}$. Se le llamará digrafo etiquetado a (G, lab) y se representará igual que G , pero agregando las etiquetas correspondientes sobre sus arcos. Por simplicidad de los dibujos, en ocasiones, se utilizarán colores para reemplazar las etiquetas. Los arcos etiquetados con \oplus se pintan azules, los con \ominus se colorean rojos y los con \circ de color negro. En la Figura 2.1b se aprecia un digrafo etiquetado en el cual aparecen los arcos con sus etiquetas y también coloreados de acuerdo a su función lab .

Dada una función de etiqueta lab , llamaremos *soporte* al conjunto:

$$\text{Sup}(\text{lab}) = \{a \in A(G) \mid \text{lab}(a) \neq \circ\}.$$

Los arcos que pertenecen al soporte también son llamados *arcos etiquetados*, los arcos restantes de G son llamados *arcos sin etiqueta*.

Se dice que (G, lab) es *totalmente etiquetado* si $\text{Sup}(\text{lab}) = A(G)$. En otro caso, se dice que está *parcialmente etiquetado*.

Una función de etiqueta $\widetilde{\text{lab}} : A(G) \rightarrow \{\oplus, \ominus, \circ\}$ es una *extensión* de $\text{lab} : A(G) \rightarrow \{\oplus, \ominus, \circ\}$ si

$$\forall a \in \text{Sup}(\text{lab}), \widetilde{\text{lab}}(a) = \text{lab}(a).$$

Además, si $\text{Sup}(\widetilde{\text{lab}}) = A(G)$, diremos que $\widetilde{\text{lab}}$ es una *extensión total*.

Dado un grafo $G = (V, A)$ y una función de etiquetado lab . Sea $a \in A(G)$ tal que $\text{lab}(a) = \circ$, se define la *extensión simple* $\text{lab}^{a=\oplus}$ como

$$\forall e \in A \setminus \{a\}, \text{lab}^{a=\oplus}(e) = \text{lab}(e) \text{ y } \text{lab}^{a=\oplus}(a) = \oplus.$$

Análogamente definimos la extensión simple $\text{lab}^{a=\ominus}$.

En [23] se definió el *digrafo de actualización* asociado a una red $N = (F, s)$ como (G, lab_s) ; donde lab_s es la función de etiquetado asociada al esquema s , que está dada por:

$$\text{lab}_s(i, j) = \begin{cases} \oplus & \text{si } s(i) \geq s(j) \\ \ominus & \text{si } s(i) < s(j) \end{cases}$$

En la Figura 2.1b se muestra un digrafo de actualización. En él se puede apreciar que se cumple $\text{lab} = \text{lab}_s$.

De esta forma, dado un etiquetado cualquiera $\text{lab} : A(G) \rightarrow \{\ominus, \oplus\}$, se dice que el digrafo (G, lab) es de actualización si existe s tal que $(G, \text{lab}) = (G, \text{lab}_s)$.

En este trabajo se extiende el concepto de *digrafo de actualización* a los parcialmente etiquetados y se dice que (G, lab) cumple con la propiedad, si existe una extensión total lab' tal que (G, lab') es de actualización.

No todo etiquetado parcial corresponde a un esquema de actualización. Esto se debe a que, como ya se dijo, el etiquetado representa un conjunto de restricciones que debería cumplir algún esquema s con respecto a ciertos pares de nodos, lo que puede llevar a contradicciones. Por ejemplo, teniendo los nodos $\{1, 2, 3\}$, podría existir un etiquetado parcial lab que indique las restricciones: $\text{lab}(1, 2) = \ominus$, $\text{lab}(2, 3) = \ominus$ y $\text{lab}(3, 1) = \ominus$, lo que significa que para algún s , $s(1) < s(2)$ (el nodo 1 se actualiza antes que el 2), y por otro lado, $s(2) < s(3)$ y $s(3) < s(1)$, por lo que $s(2) < s(1)$.

Se denota por $\mathcal{S}(G, \text{lab})$ al conjunto de extensiones totales lab' de lab que hacen que el digrafo (G, lab') sea de actualización.

Dado un digrafo etiquetado (G, lab) , se define el *digrafo reverso* (G_R, lab_R) , donde:

$$V(G_R) = V(G)$$

$$A(G_R) = \{(u, v) \mid ((u, v) \in A(G) \wedge \text{lab}(u, v) = \oplus) \vee ((v, u) \in A(G) \wedge \text{lab}(v, u) = \ominus)\}$$

$$\text{lab}_R(u, v) = \begin{cases} \ominus & \text{si } \exists (v, u) \in A(G), \text{lab}(v, u) = \ominus \\ \oplus & \text{si no} \end{cases}$$

A continuación, en la Figura 2.2b se presenta un ejemplo de digrafo reverso.

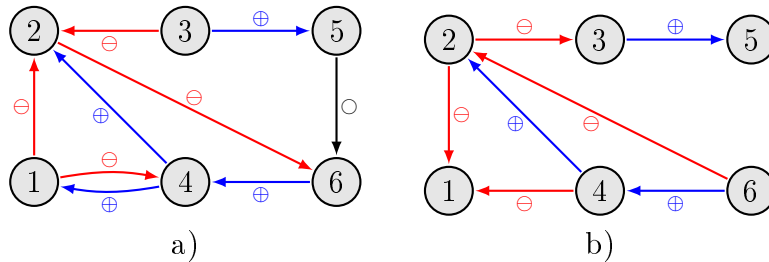


Figura 2.2: a) Un digrafo etiquetado (G, lab) . b) El digrafo reverso asociado (G_R, lab_R) .

Dado un digrafo etiquetado (G, lab) y G' un subdigrafo de G , se define la *restricción de lab al subgrafo G'* como $\text{lab}|_{A(G')} : A(G') \rightarrow \{\oplus, \ominus, \circ\}$, tal que $\text{lab}|_{A(G')}(a) = \text{lab}(a), \forall a \in A(G')$.

En este trabajo se utilizan los conceptos de *camino reverso* y *camino reverso negativo*. Estos son útiles para verificar la condición de digrafo de actualización.

Dado un digrafo etiquetado (G, lab) , se dirá que hay un *camino reverso* desde el nodo v_1 al v_n en G , si existe una secuencia de nodos (v_1, v_2, \dots, v_n) donde se cumple que:

$$\forall i \in \{1, \dots, n-1\}, [(v_i, v_{i+1}) \in A(G) \wedge \text{lab}(v_i, v_{i+1}) = \oplus] \vee [(v_{i+1}, v_i) \in A(G) \wedge \text{lab}(v_{i+1}, v_i) = \ominus],$$

es decir, hay un camino en el digrafo reverso.

Existirá un *camino reverso negativo* de v_1 a v_n , si hay un camino reverso (v_1, v_2, \dots, v_n) tal que

$$\exists i \in \{1, \dots, n-1\} : (v_{i+1}, v_i) \in A(G) \wedge \text{lab}(v_{i+1}, v_i) = \ominus,$$

o equivalentemente hay un camino en el digrafo reverso donde uno de sus arcos es negativo.

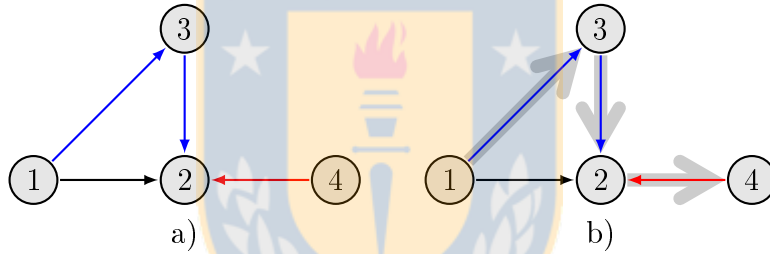


Figura 2.3: a) Digrafo etiquetado. b) Marcado un camino reverso negativo que incluye un camino reverso.

En la Figura 2.3 se ve un camino reverso de 1 a 2, dado por la secuencia $(1,3,2)$. Si se sigue el camino marcado por las flechas grises, se encuentra un camino reverso de 1 a 4. Este es también un camino reverso negativo, ya que el arco $(4,2)$ participa de la secuencia.

Con lo anterior quedan definidas las *relaciones binarias* $\text{CR} \subseteq V(G) \times V(G)$ y $\text{CRN} \subseteq V(G) \times V(G)$, de camino reverso y camino reverso negativo respectivamente. Para simplificar la notación se denota:

$$\begin{aligned} \text{CR}(i, j) &\iff (i, j) \in \text{CR}, \text{ y} \\ \text{CRN}(i, j) &\iff (i, j) \in \text{CRN} \end{aligned}$$

Observación 2.1. Se pueden enumerar las siguientes propiedades

1. La relación de camino reverso es transitiva, es decir:

$$\forall i, j, k \in V(G) : \text{CR}(i, j) \wedge \text{CR}(j, k) \implies \text{CR}(i, k)$$

2. La relación de camino reverso negativo es transitiva, es decir:

$$\forall i, j, k \in V(G) : \text{CRN}(i, j) \wedge \text{CRN}(j, k) \implies \text{CRN}(i, k)$$

3. Además es posible afirmar que se dan las siguientes propiedades:

$$\forall i, j, k \in V(G) : \text{CR}(i, j) \wedge \text{CRN}(j, k) \implies \text{CRN}(i, k)$$

$$\forall i, j, k \in V(G) : \text{CRN}(i, j) \wedge \text{CR}(j, k) \implies \text{CRN}(i, k)$$

Es fácil ver que la última propiedad se cumple, ya que en ambos casos se da la situación de que existe una secuencia de nodos de i a j y de j a k con las condiciones de camino reverso, por lo que es evidente que hay un camino reverso de i a k . Para que el camino sea reverso negativo se necesita que exista un arco negativo, lo cual sucede en ambos casos pues al menos una de las subsecuencias tiene el arco negativo.

2.2 Trabajos previos

A continuación se presentan resultados importantes conocidos en la literatura que son utilizados en esta tesis.

Los siguientes teoremas son resultados probados en [19].

Teorema 2.1 (Caracterización de un digrafo de actualización). *Un digrafo totalmente etiquetado (G, lab) es un digrafo de actualización si y sólo si su digrafo reverso (G_R, lab_R) no contiene ciclos prohibidos.*

Es importante notar que un *ciclo prohibido* es equivalente a un camino reverso negativo (v_1, v_2, \dots, v_n) donde $v_1 = v_n$. En la Figura 2.4a se muestra un digrafo que no es de actualización, mientras que en la Figura 2.4b aparece sombreado su ciclo prohibido.

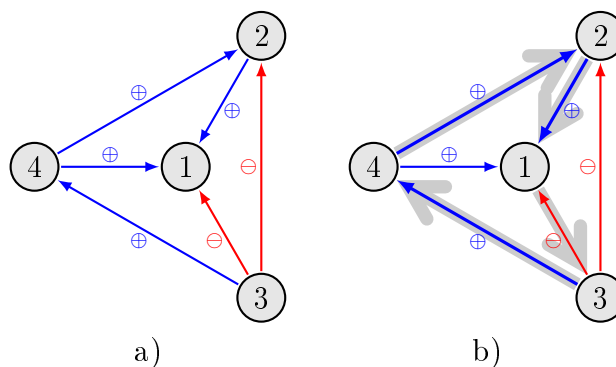


Figura 2.4: a) Un digrafo etiquetado (G, lab) que no es de actualización. b) Indicado con flechas grises el ciclo prohibido de secuencia $(4, 2, 1, 3, 4)$.

Teorema 2.2 (Extensión). *Sea G un digrafo y G' un subdigrafo de G . Si (G', lab') es un digrafo de actualización, entonces existe una función de etiquetado lab sobre $A(G)$ tal que (G, lab) es un digrafo de actualización y $\text{lab}|_{A(G')} = \text{lab}'$.*

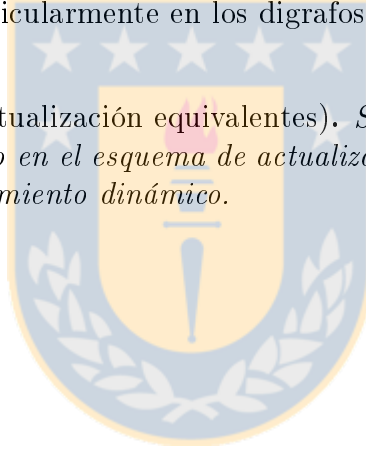
De este teorema es posible deducir que un digrafo parcialmente etiquetado que no contenga algún ciclo prohibido, puede ser completado de al menos una forma y de manera que el digrafo etiquetado resultante sea de actualización.

A partir de los Teoremas 2.1 y 2.2, se deduce el siguiente corolario.

Corolario 2.3. *El problema de existencia de una extensión en un digrafo parcialmente etiquetado es polinomial.*

El siguiente teorema fue probado en [23] e indica que dos redes que difieren solamente en el esquema de actualización tienen el mismo comportamiento dinámico (son equivalentes) si se representan mediante el mismo digrafo etiquetado. Esta es la razón por la cual el estudio realizado en esta tesis se centra particularmente en los digrafos etiquetados en vez de los esquemas de actualización.

Teorema 2.4 (Esquemas de actualización equivalentes). *Sean $N_1 = (F, s_1)$ y $N_2 = (F, s_2)$ dos redes Booleanas que difieren solo en el esquema de actualización. Si $G_{s_1}^F = G_{s_2}^F$, entonces ambas redes tienen el mismo comportamiento dinámico.*



En este capítulo se presenta el problema central a resolver en esta tesis y se estudia su complejidad asociada.

Definición 3.1 (Problema de extensión del digrafo de actualización (UDEP)). *Dado un digrafo (G, lab) etiquetado, encontrar el conjunto $\mathcal{S}(G, \text{lab})$, es decir, aquel que contiene todas las extensiones totales lab' de lab tales que (G, lab') es digrafo de actualización.*

Para abreviar el nombre del problema se usará UDEP, que viene del inglés “Update Digraph Extension Problem”.

Un primer paso para resolver el problema es conocer su complejidad. Para esto estudiamos el problema de conteo asociado, es decir, ¿cuántas extensiones tiene (G, lab) ? y probamos que el problema de conteo es $\#P$ -completo.

3.1 Complejidad

En esta sección vemos que el problema de conteo asociado a UDEP pertenece a la clase de problemas $\#P$ -completo [18]. Con esto también se entiende que el problema de encontrar el conjunto de extensiones del digrafo de actualización también es difícil. Para probar este resultado, se hace una reducción del problema de las orientaciones acíclicas en un grafo. Este consiste en dado un grafo $G = (V, E)$, se debe encontrar el número de orientaciones de G que no generan ciclo.

En términos generales la reducción consiste en dado un grafo G construir en tiempo polinomial un digrafo G' que sea una orientación acíclica de G . De esta forma G' tiene la particularidad de que los digrafos reversos asociados a uno de actualización, obtenido a partir del digrafo par-

cialmente etiquetado (G', lab_\circ) con $\text{lab}_\circ(e) = \circ, \forall e \in A(G')$, están en correspondencia uno a uno con las orientaciones acíclicas en G como se muestra en el siguiente teorema:

Teorema 3.1. *El problema de conteo asociado a UDEP es #P-completo*

Demostación. Sea G un grafo con $V(G) = \{1, \dots, n\}$, se define G' (una orientación acíclica de G) con $V(G') = V(G)$ y $A(G') = \{(u, v) | uv \in E(G) \wedge u > v\}$ (ver Figura 3.1). Obviamente la construcción de G' es polinomial.



Figura 3.1: A la izquierda se presenta un grafo G y a la derecha su digrafo G' asociado, una orientación acíclica de él.

Se denotan los conjuntos

$$O(G) = \{G_0 | G_0 \text{ orientación acíclica de } G\} \text{ y}$$

$$DA(G') = \{\text{lab} | (G', \text{lab}) \text{ es digrafo de actualización totalmente etiquetado}\}.$$

Luego, se define la siguiente función:

$$\phi : O(G) \rightarrow DA(G')$$

$$G_0 \rightarrow \phi(G_0) = \text{lab}_{G_0}$$

donde

$$\text{lab}_{G_0}(u, v) = \begin{cases} \oplus, & \text{si } (u, v) \in A(G_0) \\ \ominus, & \text{si } (v, u) \in A(G_0) \end{cases}$$

Notar que el digrafo reverso de (G', lab_{G_0}) corresponde a G_0 . Ver Figura 3.2.



Figura 3.2: A la izquierda se presenta un digrafo G_0 , una orientación acíclica de G , y la derecha el digrafo de actualización (G', lab_{G_0}) .

Además, (G', lab_{G_0}) es de actualización. En efecto, si (G', lab_{G_0}) tiene ciclo prohibido entonces G_0 tiene ciclo. Luego, ϕ está bien definido.

Por otro lado, ϕ es obviamente inyectiva. Efectivamente:

Sea $G_1 = (V, A_1)$ y $G_2 = (V, A_2)$ dos orientaciones acíclicas de G :

$$\begin{aligned}
 \phi(G_1) = \phi(G_2) &\implies \text{lab}_{G_1} = \text{lab}_{G_2} \\
 &\implies \forall (u, v) \in A(G') : \text{lab}_{G_1}(u, v) = \text{lab}_{G_2}(u, v) \\
 &\implies \forall (u, v) \in A(G') : ((u, v) \in A_1 \wedge (u, v) \in A_2) \\
 &\qquad \qquad \qquad \vee ((v, u) \in A_1 \wedge (v, u) \in A_2) \\
 &\implies A_1 = A_2 \\
 &\implies G_1 = G_2
 \end{aligned}$$

También se tiene que ϕ es sobreyectiva:

Para todo lab tal que (G', lab) es digrafo de actualización, se tiene que G'_R (el digrafo reverso de G') es una orientación acíclica de G . Efectivamente no tiene ciclos, ya que si tuviera uno entonces G' tendría un ciclo prohibido o un ciclo formado solo por arcos positivos. Lo primero es imposible, pues es digrafo de actualización y lo segundo tampoco puede suceder porque en ese caso G' tendría un ciclo.

Por lo que ϕ también es sobreyectiva y por ende biyectiva. ■

Observar que el resultado anterior indica que conocer el número total de extensiones de un digrafo parcialmente etiquetado es un problema difícil. Sin embargo, se sabe que el problema de existencia es polinomial [19].

Proposición 3.2. *Sea (G, lab) un digrafo de actualización parcialmente etiquetado, existe una única extensión para él si y solo si $\forall (i, j) \in A(G)$ con $\text{lab}(i, j) = \circ$: existe un camino reverso de i a j o uno reverso negativo de j a i .*

La proposición anterior entrega una caracterización para los digrafos que tienen una solución única. Por las condiciones encontradas se deduce que tanto el problema de saber si existe una única solución como el de encontrarla son polinomiales como se verá más adelante.



En este capítulo se presentan, explican y estudian los algoritmos utilizados para resolver UDEP. La primera idea es la llamada fuerza bruta, que correspondería a generar los etiquetados y comprobar si cumplen con la condición de digrafo de actualización.

4.1 Verifica

Lo primero que se realizará para abordar el problema es verificar si el digrafo etiquetado es de actualización. Este algoritmo es necesario incluso para utilizar la fuerza bruta. Para comprobar si un digrafo cumple o no con la propiedad se debe ver la existencia de ciclos prohibidos, tal como que se hace en el siguiente algoritmo.

Algoritmo 1 Verificar

Entrada: Un digrafo $G = (V, A)$ y una función de etiquetado $\text{lab} : A \rightarrow \{\oplus, \ominus, \circ\}$.

Salida: Verdadero si es de actualización. En caso contrario falso.

```
1: for all  $u \in V(G)$  do  
2:   if ( hay camino reverso negativo de  $u$  a  $u$  ) then  
3:     return false  
4: return true
```

Para realizar lo anterior queda clara la necesidad de conocer los caminos reversos y reversos negativos en el digrafo, ya que estos nos permiten fácilmente reconocer si existe algún ciclo prohibido.

4.1.1 Caminos reversos

Este algoritmo sirve para calcular si existe un camino reverso o reverso negativo entre cada par de nodos en un digrafo etiquetado. Está construido como una adaptación del algoritmo Floyd-Warshall que sirve para encontrar distancias mínimas entre cada par de vértices en un digrafo.

Primero, se define la siguiente función:

$$\forall(u, v) \in A(G) : d(u, v) = \begin{cases} -1 & \text{si CRN}(u, v) \\ 1 & \text{si } \neg \text{CRN}(u, v) \wedge \text{CR}(u, v) \\ \infty & \text{si } \neg \text{CR}(u, v) \end{cases}$$

y también el operador binario conmutativo \odot , que aparece definido en la Tabla 4.1 que se muestra a continuación.

a	b	$a \odot b$
∞	∞	∞
∞	1	∞
∞	-1	∞
1	1	1
1	-1	-1
-1	-1	-1

Tabla 4.1: Definición del operador \odot .

Algoritmo 2 CaminosReversos

Entrada: Un digrafo etiquetado (G, lab)

Salida: La matriz M con todos los caminos reversos y reversos negativos en (G, lab)

```

1: Sea  $M$  la matriz de  $|V(G)| \times |V(G)|$ 
2: for all  $i, j \in \{1, \dots, |V(G)|\}$  do
3:    $M(i, j) \leftarrow \infty$ 
4: for all  $(u, v) \in A(G)$  do
5:   if  $\text{lab}(u, v) = \oplus$  then
6:      $M(u, v) \leftarrow 1$ 
7: for all  $(u, v) \in A(G)$  do
8:   if  $\text{lab}(u, v) = \ominus$  then
9:      $M(v, u) \leftarrow -1$ 
10: for  $k = 1$  to  $|V(G)|$  do
11:   for  $i = 1$  to  $|V(G)|$  do
12:     for  $j = 1$  to  $|V(G)|$  do
13:       if  $M(i, k) \odot M(k, j) < M(i, j)$  then
14:          $M(i, j) \leftarrow M(i, k) \odot M(k, j)$ 
15: return  $M$ 

```

A continuación se puede apreciar en la Figura 4.1 un ejemplo de la aplicación del Algoritmo 2.

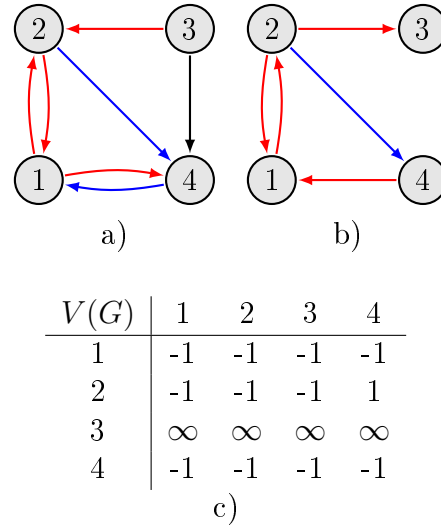


Figura 4.1: a) Un digrafo etiquetado de entrada. b) El digrafo reverso asociado a la entrada. c) Matriz de salida al aplicar el algoritmo CaminosReversos.

Queda claro que el orden del algoritmo presentado es $O(|V(G)|^3)$ por los ciclos “for” anidados, es decir, es polinomial. Este algoritmo sirve también para demostrar lo mencionado previamente respecto a la polinomialidad de los problemas de saber si un digrafo parcialmente etiquetado tiene una única extensión y de encontrarla.

La polinomialidad mencionada para CaminosReversos también vale para Verificar, ya que lo más complejo que tiene es buscar si hay ciclos prohibidos. Considerando este último algoritmo, al ingresar como entrada el digrafo etiquetado presentado en la Figura 4.1a se entrega como respuesta “falso”, ya que hay camino reverso negativo de 1 a 1.

4.2 Forzar

Es importante buscar características que permitan acortar caminos en la construcción de soluciones. Gracias a la caracterización de un digrafo de actualización y al Teorema de Extensión es posible reconocer ciertos arcos no etiquetados que solo podrían etiquetarse con uno de los dos signos (\oplus o \ominus).

Dado el digrafo de actualización $G = (V, A)$, se presentan situaciones en las que el arco sin etiqueta $(i, j) \in A(G)$ se puede etiquetar de una sola manera para que la función de etiqueta siga correspondiendo a un esquema de actualización.

- Si existe un camino reverso negativo de j a i , entonces el arco (i, j) debe ser etiquetado con \ominus .

- Si existe un camino reverso de i a j , entonces el arco (i, j) debe ser etiquetado con \oplus .

En ambos casos se dirá que el arco está *forzado*. La idea básica consiste en verificar que el arco en cuestión no forme un ciclo prohibido con alguna de las etiquetas posibles.

Ejemplo 4.1. En la Figura 4.2b se ve que existe un camino reverso negativo de 3 a 2 por la secuencia (3,1,2); mientras que en c) se ve que existe un camino reverso de 2 a 4, por esto es que los arcos (2,3) y (2,4) deben ser etiquetados negativo y positivo, respectivamente tal como se muestra en d).

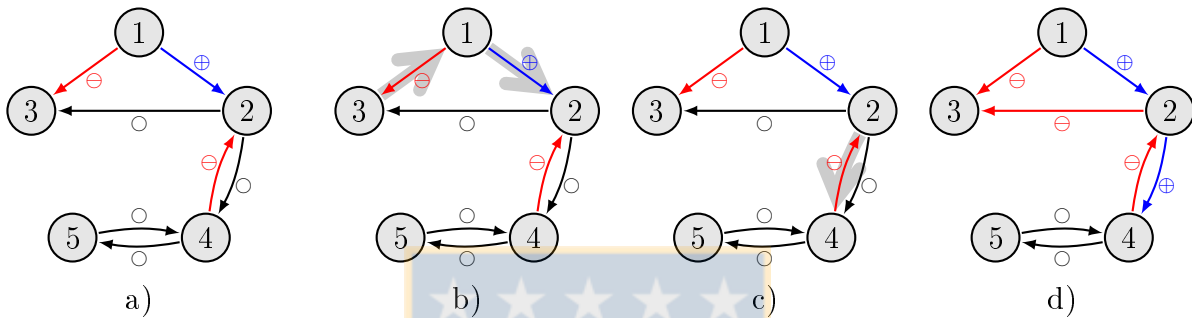


Figura 4.2: a) Un digrafo etiquetado. b) Se destaca camino reverso negativo de 3 a 2. c) Se remarca camino reverso de 2 a 4. c) Se fuerzan los arcos (2,3) y (2,4) negativo y positivo respectivamente.

Lo anterior se formaliza en la siguiente proposición.

Proposición 4.1 (Arco forzado). *Sea (G, lab) digrafo de actualización, $(i, j) \in A(G)$ y $\text{lab}(i, j) = \circ$, entonces:*

1. $\forall (i, j) \in A(G) : \text{CRN}(j, i) \iff (G, \text{lab}^{(i,j)=\oplus})$ no es digrafo de actualización; entonces (i, j) se dice forzado negativo.
2. $\forall i, j \in V(G) : \text{CR}(i, j) \iff (G, \text{lab}^{(i,j)=\ominus})$ no es digrafo de actualización; entonces (i, j) se dice forzado positivo.

Demostración.

Se probará solo el primer caso, el segundo es análogo.

(\implies) Si se toma en cuenta que existe un camino reverso negativo de j a i , efectivamente al etiquetar el arco (i, j) con \oplus , habrá un camino reverso negativo (ciclo prohibido) de j a j . Por lo que será cierto que $(G, \text{lab}^{(i,j)=\oplus})$ no es digrafo de actualización.

(\impliedby) Considerando que $(G, \text{lab}^{(i,j)=\oplus})$ no es digrafo de actualización y que (G, lab) si lo es, entonces se forma un ciclo prohibido al etiquetar (i, j) con \oplus . Por lo tanto existe un camino reverso negativo de j a i en (G, lab) . ■

Esto tiene que ver con que al etiquetar el arco, positivo en el primer caso y negativo en el segundo, se genera un ciclo prohibido. Como lo que se busca es encontrar los digrafos de actualización totalmente etiquetados se puede forzar la etiqueta correspondiente.

4.2.1 Propiedades

Se demuestra que el orden en que se fuercen los arcos es irrelevante en el etiquetado obtenido, ya que los arcos forzados no agregan información adicional en cuanto a los caminos reversos ni reversos negativos.

Proposición 4.2. *Sea (G, lab) digrafo de actualización y $a = (k, l) \in A$ un arco forzado, entonces:*

$$\forall i, j \in V(G) : \text{CRN}(i, j) \text{ en } (G, \text{lab}) \iff \text{CRN}(i, j) \text{ en } (G, \text{lab}^{a=\oplus})$$

$$\forall i, j \in V(G) : \text{CRN}(i, j) \text{ en } (G, \text{lab}) \iff \text{CRN}(i, j) \text{ en } (G, \text{lab}^{a=\ominus})$$

Demostración.

Se probará el primer caso, que corresponde al arco forzado positivo. Para el forzado negativo la demostración es análoga.

(\implies) Es verdadero, porque $(G, \text{lab}^{(i,j)=\oplus})$ es extensión de (G, lab) . Como en una extensión están al menos los mismos arcos etiquetados, es fácil entender que un camino reverso negativo en el original, también se encontrará en su extensión.

(\impliedby) Tomando un camino reverso negativo entre dos nodos cualquiera en $(G, \text{lab}^{a=\oplus})$, se hará ver que necesariamente existe un camino reverso negativo en (G, lab) . La única diferencia entre los dos digrafos está en arco a que es forzado positivo. Esto implica que existía un camino reverso de k a l en (G, lab) , que es la misma información obtenida por el etiquetado de a . ■

Como $\text{CRN}(i, j) \implies \text{CR}(i, j)$ la proposición anterior también vale para caminos reversos, es decir, dado un digrafo de actualización, cualquier extensión de ella construida a partir de arcos forzados mantiene la misma información en cuanto a caminos reversos y reversos negativos.

4.2.2 Extensión maximal

A partir de la idea anterior, surge la siguiente definición:

Definición 4.1. *Sea (G, lab) un digrafo de actualización. Entonces se dice que $\widetilde{\text{lab}}$ es una **extensión maximal** de lab si es una extensión de lab y $\forall a \notin \text{Sup}(\text{lab})$:*

1. Si a es forzado positivo, entonces $\widetilde{\text{lab}}(a) = \oplus$.
2. Si a es forzado negativo, entonces $\widetilde{\text{lab}}(a) = \ominus$.
3. En otro caso, $\widetilde{\text{lab}}(a) = \circ$

Notar que dado un digrafo de actualización (G, lab) y $\widetilde{\text{lab}}$ su extensión maximal, se cumple que:

$$\text{Sup}(\widetilde{\text{lab}}) = \text{Sup}(\text{lab}) \cup \{a \in A(G) : a \text{ es un arco forzado}\},$$

es decir, es la única extensión que etiqueta sus arcos forzados y mantiene la condición de digrafo de actualización.

Se presenta el Algoritmo 3, llamado **Forzar**, que ayuda a acortar el espacio de búsqueda utilizando lo observado en esta sección, ya que etiqueta correctamente los arcos forzados de un digrafo etiquetado; es decir, entrega la extensión maximal.

Algoritmo 3 Forzar

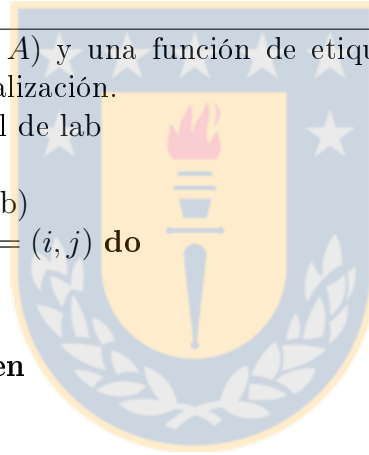
Entrada: Un digrafo $G = (V, A)$ y una función de etiqueta $\text{lab} : A \rightarrow \{\oplus, \ominus, \circ\}$, tal que (G, lab) sea digrafo de actualización.

Salida: Una extensión maximal de lab

```

1:  $\widetilde{\text{lab}} \leftarrow \text{lab}$ 
2:  $M \leftarrow \text{CaminosReversos}(G, \text{lab})$ 
3: for all  $a \in \text{Sup}(\text{lab})$ , con  $a = (i, j)$  do
4:   if  $M(i, j) \neq \infty$  then
5:      $\widetilde{\text{lab}} \leftarrow \widetilde{\text{lab}}^{a=\oplus}$ 
6:   else if  $M(j, i) = -1$  then
7:      $\widetilde{\text{lab}} \leftarrow \widetilde{\text{lab}}^{a=\ominus}$ 
8: return  $\widetilde{\text{lab}}$ 

```



Ejemplo 4.2. El digrafo que aparece en la Figura 4.2d es la extensión maximal del que aparece en la Figura 4.2a. Es la salida que entrega el Algoritmo 3. Este también revisaría los arcos $(5, 4)$ y $(4, 5)$, sin embargo no hay caminos reversos ni caminos reversos negativos entre ellos, por lo tanto no son arcos forzados.

Con los algoritmos mostrados hasta ahora ya es posible construir un algoritmo inicial que resuelva UDEP y sea mejor que la fuerza bruta. La idea sería verificar inicialmente la condición de digrafo de actualización, luego forzar y en caso de que no esté totalmente etiquetado, etiquetar alguno de los arcos sin etiqueta primero con un signo y después con el opuesto (recursivamente); y repetir así el proceso de forzar y etiquetar recursivamente. Así se van obteniendo los etiquetados esperados con la ventaja de no explorar soluciones no válidas y solo verificando la condición de actualización al digrafo de entrada.

El Algoritmo 4 permite realizar todo el proceso de etiquetado que resuelve UDEP. Falta un algoritmo que haga la Verificación y llame a `EtiquetarSimple`. Así estaríamos en presencia del algoritmo mencionado en el párrafo anterior.

Algoritmo 4 EtiquetarSimple

Entrada: Un digrafo $G = (V, A)$ y una función de etiqueta $\text{lab} : A \rightarrow \{\oplus, \ominus, \circ\}$, tal que (G, lab) sea digrafo de actualización.

Salida: El conjunto $\mathcal{S}(G, \text{lab})$ representado por S y $r := |\mathcal{S}(G, \text{lab})|$

```

1:  $S \leftarrow \emptyset$ 
2:  $r \leftarrow 0$ 
3:  $\text{lab} \leftarrow \text{Forzar}(G, \text{lab})$ 
4: if  $\text{Sup}(\text{lab}) = A(G)$  then
5:   return  $\{\text{lab}, 1\}$ 
6: else
7:   Sea  $a \in A(G) \setminus \text{Sup}(\text{lab})$ 
8:    $\{S_1, r_1\} \leftarrow \text{EtiquetarSimple}(G, \text{lab}^{a=\oplus})$ 
9:    $\{S_2, r_2\} \leftarrow \text{EtiquetarSimple}(G, \text{lab}^{a=\ominus})$ 
10:   $r \leftarrow r_1 + r_2$ 
11:   $S \leftarrow S_1 \cup S_2$ 
12: return  $\{S, r\}$ 

```

4.3 Reducción del digrafo

Como ya se vio en el capítulo anterior, UDEP pertenece a una clase en la cual no se han encontrado algoritmos polinomiales que los resuelvan. Por esto es importante realizar simplificaciones del mismo, para así reducir el tamaño de la entrada, y con esto lo que se demore en responder el algoritmo. Lo primero que se hace en esta dirección es definir un digrafo reducido donde cada componente fuertemente conexa positiva es reemplazada por un único nodo, como se explica en Teorema 4.4. Las *componentes fuertemente conexas positivas* de un digrafo etiquetado, son las CFC del digrafo inducido por los arcos positivos del mismo. Para poder definir correctamente el digrafo reducido es necesario el siguiente resultado:

Proposición 4.3. *Sea (G, lab) un digrafo de actualización, G_1 y G_2 componentes fuertemente conexas positivas de G , y $\widetilde{\text{lab}}$ una extensión tal que $(G, \widetilde{\text{lab}})$ es digrafo de actualización totalmente etiquetado. Entonces $\forall (u, v), (x, y) \in A(G) \cap (V(G_1) \times V(G_2)) : \widetilde{\text{lab}}(u, v) = \widetilde{\text{lab}}(x, y)$.*

Demostración. Sea $(G, \widetilde{\text{lab}})$ un digrafo de actualización totalmente etiquetado, con G_1 y G_2 dos componentes fuertemente conexas positivas de G ; y $(u, v), (x, y) \in A(G) \cap (V(G_1) \times V(G_2))$. Supongamos que $\widetilde{\text{lab}}(u, v) = \oplus$ y $\widetilde{\text{lab}}(x, y) = \ominus$. Con esto, se tiene que $\text{CR}(u, v)$ y $\text{CRN}(y, x)$. Además, $\text{CR}(v, y)$ y $\text{CR}(x, u)$, porque $u, x \in V(G_1)$ y $v, y \in V(G_2)$. Luego, se deduce por las propiedades de ambas relaciones que $\text{CRN}(u, u)$, es decir, hay un ciclo prohibido. Esto representa una contradicción con el hecho de que $(G, \widetilde{\text{lab}})$ es digrafo de actualización. Por lo tanto, se debe cumplir que $\widetilde{\text{lab}}(u, v) = \widetilde{\text{lab}}(x, y)$. ■

De la proposición anterior se entiende, que es posible etiquetar arcos previamente (arcos forzados). Esto evita que el digrafo reducido sea un multidigrafo o que haya problemas en la definición de su función de etiqueta.

Definición 4.2. Sea (G, lab) un digrafo de actualización y sus componentes fuertemente conexas positivas $\{G_1, \dots, G_k\}$. Se define el digrafo etiquetado reducido $R(G, \text{lab})$ asociado a (G, lab) por $R(G, \text{lab}) = (G_{\text{red}} = (V_{\text{red}}, A_{\text{red}}), \text{lab}_{\text{red}})$, donde $V_{\text{red}} = \{v_1, \dots, v_k\}$ y $A_{\text{red}} = \{(v_i, v_j) | \exists (u, v) \in A(G) \cap (V(G_i) \times V(G_j))\}$. Además, $\text{lab}_{\text{red}}(v_i, v_j) = \text{lab}(u, v)$, si existe $(u, v) \in (V(G_i) \times V(G_j)) \cap \text{Sup}(\text{lab})$ y $\text{lab}_{\text{red}}(v_i, v_j) = \circ$ en caso contrario. Un digrafo etiquetado (G, lab) se dice reducido si $(G, \text{lab}) = R(G, \text{lab})$.

En la Figura 4.3 aparece un ejemplo de una reducción de digrafo etiquetado (G, lab) a $R(G, \text{lab})$.

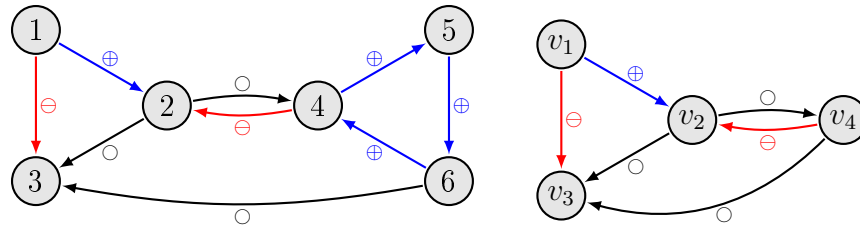


Figura 4.3: A la izquierda se aprecia el digrafo etiquetado (G, lab) con los nodos de sus cuatro componentes fuertemente conexas positivas $\{G_1, G_2, G_3, G_4\}$ con $V(G_1) = \{1\}$, $V(G_2) = \{2\}$, $V(G_3) = \{3\}$ y $V(G_4) = \{4, 5, 6\}$ y a la derecha su digrafo reducido $R(G, \text{lab}) = (G_{\text{red}}, \text{lab}_{\text{red}})$, donde v_i representa a la CFCEP G_i .

Observar que si (G, lab) es conexo, entonces obviamente $R(G, \text{lab})$ es también conexo. Además como (G, lab) es digrafo de actualización, entonces $R(G, \text{lab})$ también lo es.

Teorema 4.4. El problema de extensión del digrafo de actualización para (G, lab) , está en biyección con el UDEP para $R(G, \text{lab})$.

Demostración. Sea la función

$$\phi : \mathcal{S}(G, \text{lab}) \rightarrow \mathcal{S}(R(G, \text{lab})),$$

definida por: $\forall \text{lab}' \in \mathcal{S}(G, \text{lab}), \phi(\text{lab}') = \text{lab}''$, tal que $\forall (v_i, v_j) \in A_{\text{red}}, \text{lab}''(v_i, v_j) = \text{lab}'(u, v)$; donde $u \in V(G_i)$ y $v \in V(G_j)$.

Se prueba que $(G_{\text{red}}, \text{lab}'')$ es digrafo de actualización.

Supongamos que existe un ciclo prohibido en $(G_{\text{red}}, \text{lab}'')$, entonces se puede ver que existe una secuencia (u_1, \dots, u_l) tal que

$$\forall i \in \{1, \dots, l-1\}, [(u_i, u_{i+1}) \in A(G_{\text{red}}) \wedge \text{lab}''(u_i, u_{i+1}) = \oplus] \vee [(u_{i+1}, u_i) \in A(G_{\text{red}}) \wedge \text{lab}''(u_{i+1}, u_i) = \ominus].$$

Sin pérdida de generalidad, diremos que u_i es el representante de la componente fuertemente conexa positiva G_i . Luego, por la definición de $A(G_{\text{red}})$ se sabe que dado $(u_j, u_k), (u_k, u_h) \in$

$A(G_{red})$ existe $(v_1, v_2), (v_3, v_4) \in A(G)$ con $v_2, v_3 \in G_k, v_1 \in G_j, v_4 \in G_h$, por lo que se entiende que dado un camino reverso o reverso negativo en G_{red} , existe uno en G . Luego, si hay ciclo prohibido en (G_{red}, lab'') también habrá uno en (G, lab') . Por lo tanto ϕ está bien definida.

Además ϕ es inyectiva. En efecto:

Dados $lab_1, lab_2 \in \mathcal{S}(G, lab)$, $\phi(lab_1) = lab''_1$, $\phi(lab_2) = lab''_2$ y $\{G_1, \dots, G_k\}$ las componentes fuertemente conexas positivas de (G, lab) entonces:

$$\begin{aligned} lab_1 \neq lab_2 &\Rightarrow \exists (u, v) \in A(G), lab_1(u, v) \neq lab''_1(u, v), \text{ con } u \in V(G_i), v \in V(G_j), \text{ y } i \neq j \\ &\Rightarrow lab''_1(w_i, w_j) \neq lab''_2(w_i, w_j) \end{aligned}$$

También se tiene que ϕ es sobreyectiva:

Dado $lab'' \in \mathcal{S}(R(G, lab))$, se define la función lab' como sigue:

$$\begin{aligned} \forall (u, v) \in A(G); \forall i, j \in \{1, \dots, k\}, i \neq j, \text{ con } u \in G_i, v \in G_j : lab'(u, v) = lab''(u, v) \text{ y} \\ \forall (u, v) \in A(G); \forall i \in \{1, \dots, k\}, \text{ con } u, v \in G_i : lab'(u, v) = \oplus. \end{aligned}$$

Es fácil ver que $lab' \in \mathcal{S}(G, lab)$ y $\phi(lab') = lab''$. ■

La reducción mencionada en esta sección puede realizarse mediante el Algoritmo 5, que se detalla a continuación.

Algoritmo 5 Reducir

Entrada: Un digrafo $G = (V, A)$ y una función de etiquetado lab , tal que (G, lab) es de actualización

Salida: El digrafo reducido de (G_{red}, lab_{red}) .

```

1:  $\{G_1, \dots, G_k\} \leftarrow \text{CFC}^+(G, lab)$ 
2:  $V(G_{red}) \leftarrow \{w_1, \dots, w_k\}$ 
3:  $A(G_{red}) \leftarrow \emptyset$ 
4: for  $i = 1$  to  $k$  do
5:   for  $j = 1$  to  $k$  do
6:     if  $\exists (u, v) \in A(G)$  con  $u \in V(G_i)$  y  $v \in V(G_j)$  then
7:        $A(G_{red}) \leftarrow A(G_{red}) \cup (w_i, w_j)$ 
8:   for all  $(w_i, w_j) \in A(G_{red})$ 
9:     if  $\exists u \in V(G_i), v \in V(G_j)$  y  $lab(u, v) = \ominus$  then
10:       $lab_{red}(u, v) \leftarrow \ominus$ 
11:     else if  $\exists u \in V(G_i), v \in V(G_j)$  y  $lab(u, v) = \oplus$  then
12:       $lab_{red}(u, v) \leftarrow \oplus$ 
13:     else
14:       $lab_{red}(u, v) \leftarrow \circ$ 
15: return  $(G_{red}, lab_{red})$ 

```

En el algoritmo se aplica la rutina CFC^+ , que es aquella que entrega las componentes fuertemente conexas positivas del digrafo de entrada (G, lab) . Esto es muy simple de hacer utilizando

el algoritmo de Tarjan, y considerando solamente los arcos positivos del digrafo etiquetado. Por lo tanto, el algoritmo presentado es $O(|V(G)| + |A(G)|)$.

Un tema importante a considerar es que verificar la condición de digrafo de actualización sobre uno reducido puede ser efectuado usando el algoritmo de Tarjan sobre el digrafo reverso que omite los arcos no etiquetados, ya que en este caso consistiría en encontrar una componente fuertemente conexa (de más de un nodo) sobre el digrafo mencionado. Esto reduciría la complejidad de verificar.

4.4 Dividir para conquistar

Otra de las técnicas utilizadas para reducir el tiempo total en la resolución del problema consiste en realizar ciertas divisiones al digrafo con el fin de separar en subproblemas más pequeños. Para esto es necesario introducir la siguiente definición:

Definición 4.3. Dado un digrafo G , $\{A_1, \dots, A_k\}$ una partición de $A(G)$, y $L_i \subseteq \{\text{lab} \mid \text{lab} : A_i \rightarrow \{\oplus, \ominus, \circ\}\}$ es función de etiqueta de G ; se define

$$L_1 \otimes \dots \otimes L_k = \{\text{lab} : A \rightarrow \{\oplus, \ominus, \circ\} \mid \forall i \in \{1, \dots, k\}, \exists \text{lab}_i \in L_i, \text{ donde } \forall a \in A_i, \text{lab}(a) = \text{lab}_i(a)\}$$

Notar que por definición se tiene que:

$$\begin{aligned} \forall L, L' : L \otimes L' &= L' \otimes L \\ \forall L : \emptyset \otimes L &= L \end{aligned}$$

El resultado que se presenta a continuación es directo de la definición:

Proposición 4.5. Sea (G, lab) un digrafo tal que sus componentes conexas son G_1, \dots, G_k , entonces

$$\mathcal{S}(G, \text{lab}) = \mathcal{S}(G_1, \text{lab}|_{A(G_1)}) \otimes \dots \otimes \mathcal{S}(G_k, \text{lab}|_{A(G_k)})$$

4.4.1 División por puentes

Una manera de dividir el problema es separando por puentes sobre el grafo fundamental. Esto es posible debido a que no es posible encontrar un ciclo prohibido que pase por un puente, a menos que sea un ciclo de largo dos.

Proposición 4.6. Sea (G, lab) un digrafo conexo, G_U el grafo fundamental de G y $uv \in E(G_U)$ un puente que separa a G en G_1 y G_2 , entonces

$$\mathcal{S}(G, \text{lab}) = \mathcal{S}(G_1, \text{lab}|_{A(G_1)}) \otimes \mathcal{S}(G_2, \text{lab}|_{A(G_2)}) \otimes \mathcal{S}(G[\{u, v\}], \text{lab}|_{A(G[\{u, v\}]})$$

Demostración. Dado $\text{lab}' \in \mathcal{S}(G, \text{lab})$, entonces es claro que $\text{lab}' \in \mathcal{S}(G_1, \text{lab}|_{A(G_1)}) \otimes \mathcal{S}(G_2, \text{lab}|_{A(G_2)}) \otimes \mathcal{S}(G[\{u, v\}], \text{lab}|_{A(G[\{u, v\}]})$.

Dado $\text{lab}' \in \mathcal{S}(G_1, \text{lab}|_{A(G_1)}) \otimes \mathcal{S}(G_2, \text{lab}|_{A(G_2)}) \otimes \mathcal{S}(G[\{u, v\}], \text{lab}|_{A(G[\{u, v\}]})$, supongamos que $u \in G_1$ y $v \in G_2$, y que $\text{lab}' \notin \mathcal{S}(G, \text{lab})$, esto implica que lab' es una extensión total que contiene un ciclo prohibido. Sin pérdida de generalidad, digamos que el ciclo comienza en el nodo u_1 que es parte de la componente G_1 , como (G_1, lab_1) es de actualización no existe tal ciclo dentro del subgrafo. Esto implicaría que (u, v) debe ser parte de la secuencia, sin embargo, no podría haber ciclo prohibido dentro de $G[\{u, v\}]$, por lo que el ciclo supuesto debiese pasar por otros nodos de G_2 , lo cual es imposible. Por lo tanto, $\text{lab} \in \mathcal{S}(G, \text{lab})$. ■

En la Figura 4.4 se puede apreciar una división por puentes.

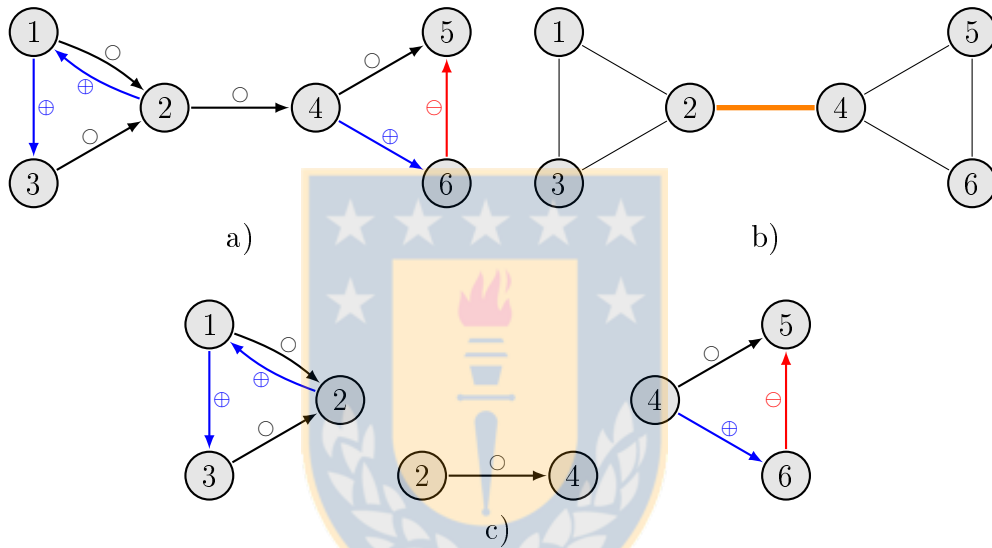


Figura 4.4: a) Un digrafo etiquetado (G, lab) . b) El grafo fundamental de G con un puente destacado con naranja. c) Los tres subdigrafos resultantes luego de la división por puentes.

4.4.2 Dividir por componentes fuertemente conexas

Otra división posible consiste en separar las distintas componentes fuertemente conexas en el digrafo reverso extendido, que es similar al digrafo reverso con la diferencia que este no tiene etiquetas y que los arcos sin etiqueta de original aparecen en ambas direcciones.

Proposición 4.7. Sea (G, lab) un digrafo de actualización con componentes fuertemente conexas (no necesariamente positivas) G_1, \dots, G_k en el digrafo reverso extendido, se define el conjunto de arcos:

$$A_T = \bigcup_{i,j} A(G) \cap (V(G_i) \times V(G_j)),$$

el cual está compuesto por aquellos arcos que unen las diversas componentes fuertemente conexas en el digrafo reverso extendido de G . Entonces se verifica que:

$$\mathcal{S}(G, \text{lab}) = \mathcal{S}(\tilde{G}_1, \text{lab}|_{A(G_1)}) \otimes \dots \otimes \mathcal{S}(\tilde{G}_k, \text{lab}|_{A(G_k)}) \otimes \{\text{lab}|_{A_T}\}$$

donde $\forall i \in \{1, \dots, k\}$, $\tilde{G}_i = G[V(G_i)]$

Demostración. Dado $\text{lab}' \in \mathcal{S}(G, \text{lab})$ si no existe ciclo prohibido en G , tampoco existirá en alguna de sus componentes fuertemente conexas del digrafo reverse extendido, por lo que $\text{lab}' \in \mathcal{S}(G, \text{lab}) \implies \text{lab}' \in \mathcal{S}(\tilde{G}_1, \text{lab}|_{A(G_1)}) \otimes \dots \otimes \mathcal{S}(\tilde{G}_k, \text{lab}|_{A(G_k)}) \otimes \{\text{lab}|_{A_T}\}$. Además, $\text{lab}' \in \mathcal{S}(\tilde{G}_1, \text{lab}|_{A(G_1)}) \otimes \dots \otimes \mathcal{S}(\tilde{G}_k, \text{lab}|_{A(G_k)}) \implies \text{lab}' \in \mathcal{S}(G, \text{lab})$. Esta implicancia también es fácil ver; ya que, dado que son componentes fuertemente conexas, no hay caminos de ida y vuelta entre ellas. ■

Notar que los arcos en A_T son etiquetados.

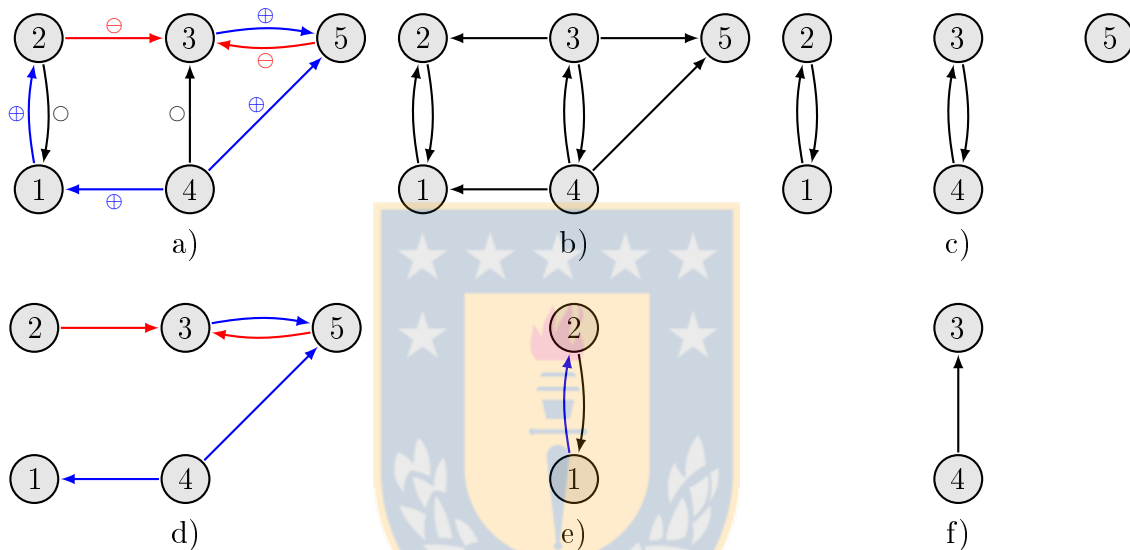


Figura 4.5: a) Digrafo etiquetado a ser dividido por CFC. b) Digrafo reverse extendido asociado. c) Los digrafos inducidos por la CFC del digrafo reverse extendido. d) Los arcos entre componentes A_T . e) El subproblema inducido por la componente $\{1,2\}$. f) El subproblema inducido por la componente $\{3,4\}$.

4.5 Algoritmo principal

En esta sección se presenta la versión final del algoritmo que resuelve el problema planteado.

El algoritmo `EtiquetadosUpdate` verifica que el digrafo etiquetado inicial, que ya fue reducido, no contenga un ciclo prohibido para luego Forzar y llamar a la función `Etiquetar`.

Algoritmo 6 EtiquetadosUpdate

Entrada: Un digrafo $G = (V, A)$ y la función de etiquetado $\text{lab} : A \rightarrow \{\oplus, \ominus, \circ\}$

Salida: El conjunto $\mathcal{S}(G, \text{lab})$ denotado por S y su cardinal $r := |S|$

```

1:  $(G_{red}, \text{lab}_{red}) \leftarrow \text{Reducir}(G, \text{lab})$ 
2: if Verifica( $G, \text{lab}$ ) = false then
3:   return  $(\emptyset, 0)$ 
4: else
5:    $\widetilde{\text{lab}} \leftarrow \text{Forzar}(G_{red}, \text{lab}_{red})$ 
6:   return Etiquetar( $G_{red}, \widetilde{\text{lab}}$ )

```

En Etiquetar, se agregan las divisiones del problema presentadas en la sección anterior. Primero se divide por puentes y se trata de dividir cada una de esas divisiones por componentes fuertemente conexas positivas. Estas componentes obtenidas se trabajan de forma individual, se ve si quedan arcos por etiquetar, de ser así se realiza el etiquetado recursivo de uno de sus arcos sin etiqueta, asignándole etiqueta y forzando para volver a llamar a Etiquetar. Cuando se hace el regreso recursivo se cambia el signo del arco a negativo para que de esta manera se exploren todas las posibilidades válidas.



Algoritmo 7 Etiquetar**Entrada:** Un digrafo $G = (V, A)$ y la función de etiquetado $\text{lab} : A \rightarrow \{\oplus, \ominus, \circ\}$ **Salida:** El conjunto $\mathcal{S}(G, \text{lab})$ denotado por S y su cardinal $r := |S|$

```

1:  $S \leftarrow \emptyset$ 
2:  $r \leftarrow 1$ 
3:  $\{(G_1, \text{lab}_{G_1}), \dots, (G_k, \text{lab}_{G_k})\} \leftarrow \text{Puentes}(G_U)$ 
4: for  $i = 1$  to  $k$  do
5:   if  $\text{Sup}(\text{lab}_{G_i}) = A(G_i)$  then
6:      $S \leftarrow S \otimes \{\text{lab}_{G_i}\}$ 
7:   else
8:      $\{(H_1, \text{lab}_{H_1}), \dots, (H_p, \text{lab}_{H_p})\} \leftarrow \text{CFC}(G_i, \text{lab}_{G_i})$ 
9:     if  $p = 1$  then
10:      Sea  $a \in A(H_1)$  tal que  $\text{lab}(a) = \circ$ 
11:       $\text{lab}^+ \leftarrow \text{Forzar}(H_1, \text{lab}_{H_1}^{a=\oplus})$ 
12:      if  $\text{Sup}(\text{lab}^+) \neq A(G)$  then
13:         $(\hat{S}, \hat{r}) \leftarrow \text{Etiquetar}(H_1, \text{lab}^+)$ 
14:      else
15:         $(\hat{S}, \hat{r}) \leftarrow (\{\text{lab}^+\}, 1)$ 
16:       $\text{lab}^- \leftarrow \text{Forzar}(H_1, \text{lab}_{H_1}^{a=\ominus})$ 
17:      if  $\text{Sup}(\text{lab}^-) \neq A(G)$  then
18:         $(\tilde{S}, \tilde{r}) \leftarrow \text{Etiquetar}(H_1, \text{lab}^-)$ 
19:      else
20:         $(\tilde{S}, \tilde{r}) \leftarrow (\{\text{lab}^-\}, 1)$ 
21:       $r \leftarrow r \cdot (\hat{r} + \tilde{r})$ 
22:       $S \leftarrow S \otimes (\hat{S} \cup \tilde{S})$ 
23:     else
24:       for  $j = 1$  to  $p$  do
25:         if  $\text{Sup}(\text{lab}_{H_j}) = A(H_j)$  then
26:            $S \leftarrow S \otimes \{\text{lab}_{H_j}\}$ 
27:         else
28:            $(\tilde{S}, \tilde{r}) \leftarrow \text{Etiquetar}(H_j, \text{lab}_{H_j})$ 
29:            $S \leftarrow S \otimes \tilde{S}$ 
30:            $r \leftarrow r \cdot \tilde{r}$ 
31: return  $(S, r)$ 

```

En cuanto a la complejidad del Algoritmo 7 podemos ver que los algoritmos aplicados son polinomiales. Está primero Puentes, que es $O(|V| + |A|)$, luego a las componentes resultantes se les aplica CFC, que tiene la misma complejidad. A continuación de eso se aplica Forzar que es el algoritmo más complejo, pues es $O(|V|^3)$. Posteriormente se llama recursivamente a Etiquetar, esto está relacionado a la cantidad de etiquetados que debe generar y que es $O(2^{|A|})$, el cual corresponde al número máximo de funciones de etiquetas diferentes de un digrafo y que es alcanzado cuando el grafo fundamental no tiene ciclos, es decir, todas las combinaciones de etiquetas generan digrafos update. A pesar de esto, estos se obtienen de una manera bastante mejorada respecto de la fuerza bruta principalmente por la disminución del espacio de búsqueda

provista por Forzar, por otro lado para reducir tiempos se utiliza la idea de dividir para conquistar aplicada en Puentes y CFC, que permiten reducir el tamaño de la matriz de caminos reversos calculada constantemente.

Se ingresa la entrada mostrada en Figura 4.6a que representa al digrafo etiquetado (G, lab) de la Figura 4.6b.

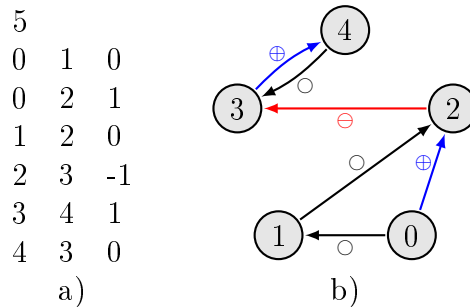


Figura 4.6: a) Archivo de entrada. b) Digrafo etiquetado (G, lab) asociado a la entrada.

Luego, al terminar la ejecución. La implementación del programa entrega la salida mostrada en la Figura 4.16. Esta representa al conjunto $\mathcal{S}(G, \text{lab})$ que contiene los seis digrafos de la Figura 4.15.

Los primeros arcos en el archivo de salida corresponden a los etiquetados inicialmente. Representan a la partición de arcos que aparece en la Figura 4.7. Luego, aparecen arcos agrupados entre llaves generados por la división por puentes que se hace con el digrafo inicial. En la Figura 4.8 se muestran los puentes del digrafo fundamental de G .

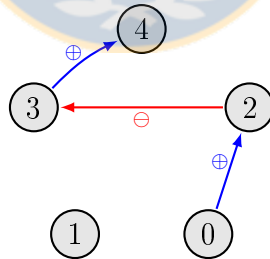


Figura 4.7: Arcos etiquetados inicialmente en (G, lab) .

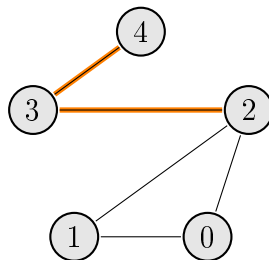


Figura 4.8: Grafo fundamental asociado a la entrada con los puentes $\{2, 3\}$ y $\{3, 4\}$.

Con la división mencionada anteriormente se separa el problema en la resolución de tres subdigrafos que son los mostrados en la Figura 4.9. En el archivo de salida se pueden apreciar las soluciones a estas divisiones separadas por las “X”.

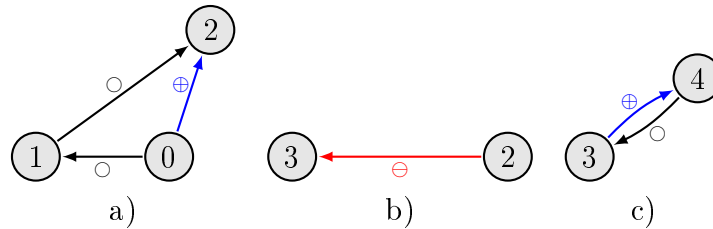


Figura 4.9: Los tres subdigrafos generados al realizar la división por puentes en (G, lab) .

Dentro de las soluciones de los subdigrafos no aparecen obviamente los arcos etiquetados, porque como ya se dijo están al comienzo. Así, entonces se puede ver que en el primer grupo aparecen los resultados del subdigrafo de la Figura 4.9c, que son especificados en la Figura 4.10. Luego, viene una solución vacía que corresponde al subdigrafo de la Figura 4.9b que ya está totalmente etiquetado. Y finalmente, las correspondientes a la Figura 4.9a, estas se muestran con detalle en la Figura 4.11. En este subdigrafo se comienza etiquetando positivo el arco $(0, 1)$, al hacer esto se produce una división por componentes fuertemente conexas tal como se muestra en la Figura 4.12. Por lo que se resuelve el subproblema como se aprecia en la Figura 4.13. Luego, al regresar en la recursión se etiqueta negativo el arco $(0, 1)$ y se fuerza el arco restante tal como se explica en la Figura 4.14.

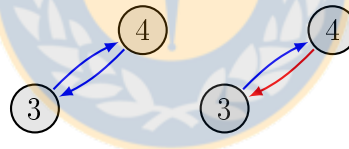


Figura 4.10: Soluciones del subdigrafo inducido por los nodos 3 y 4.

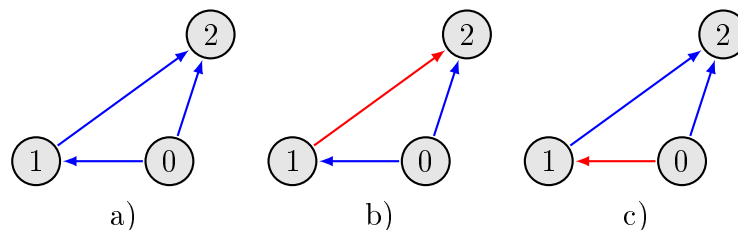


Figura 4.11: Soluciones del subdigrafo inducido por los arcos 0, 1 y 2.

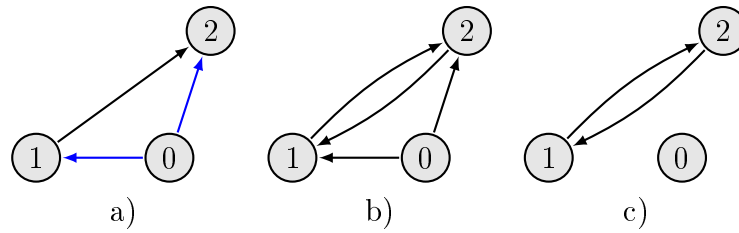


Figura 4.12: a) Subdigrafo luego de etiquetar positivo el arco $(0, 1)$ positivo. b) Digrafo reverso extendido asociado. c) Los digrafos inducidos por las componentes fuertemente conexas $\{0\}$ y $\{1, 2\}$.

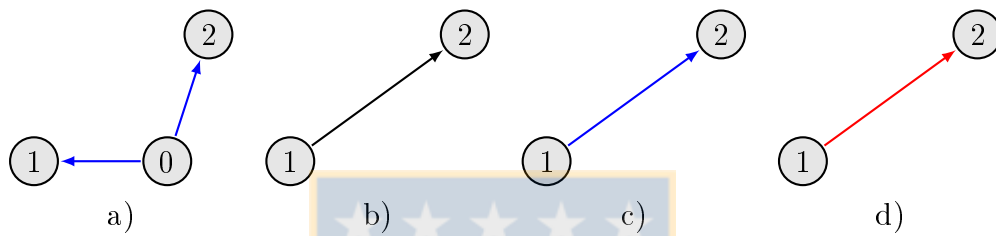


Figura 4.13: a) Arcos entre componentes A_T . b) Subproblema a resolver. c) Solución al etiquetar positivo. d) Solución al etiquetar negativo.

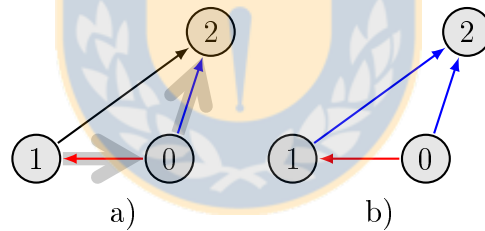


Figura 4.14: a) Hay camino reverso de 0 a 2. b) El arco $(0, 2)$ es forzado positivo.

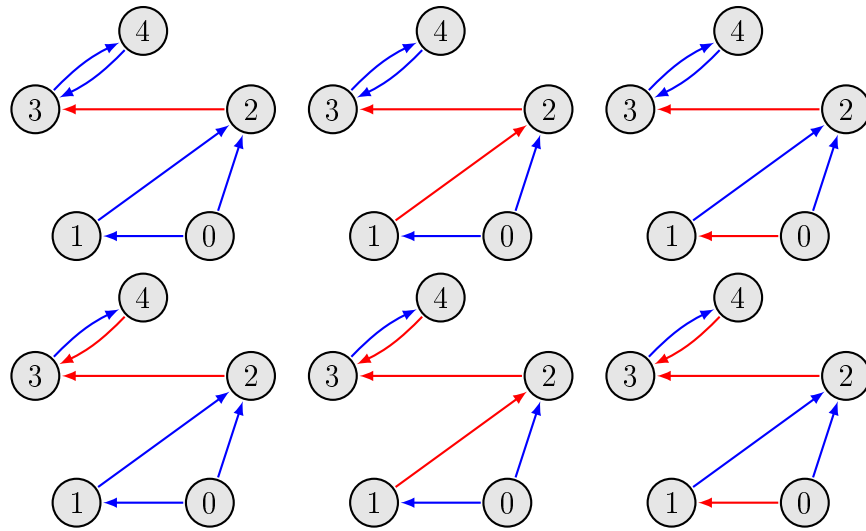


Figura 4.15: Los seis elementos del conjunto $\mathcal{S}(G, \text{lab})$.



```

{
  (0, 2, +); (2, 3, -); (3, 4, +)
}
X
{
  {
    {(4, 3, +)} U {(4, 3, -)}
  }
  X
  {}
  X
  {
    {
      {
        {(1, 2, +)} U {(1, 2, -)}
      }
      }
      X
      {(0, 1, +)}
    }
    }
    U
    {(0, 1, -); (1, 2, +)}
  }
}

```


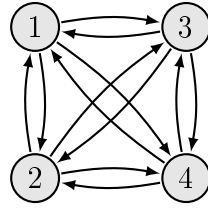


Figura 4.16: Archivo de salida entregado por la implementación de `EtiquetadosUpdate`.

4.6 Ejemplos de casos extremos

4.6.1 Digrafos Completos

Para tener una idea de la situación más compleja que se presenta en este problema para un digrafo G con $|V(G)| = n$, se muestra el comportamiento del algoritmo teniendo como entrada digrafos completos para ciertos valores de n . Se sabe que la cantidad de digrafos de actualización completamente etiquetados está dada por los números de Fubinni, y que justamente corresponde al mayor número posible para una cantidad de nodos dada. En la Figura 4.17 se aprecia el digrafo completo para $n = 4$.

Figura 4.17: Ejemplo del digrafo completo de $n = 4$ (K_4).

Así al probar con el algoritmo propuesto se obtiene la siguiente tabla resumen:

Nodos	Arcos	$ \mathcal{S}(G, \text{lab}) $	Tiempo CE EtiquetadosUpdate	Tiempo SE EtiquetadosUpdate	Tiempo CE EtiquetadoSimple
3	6	13	0,02	0,01	0,02
4	12	75	0,04	0,03	0,05
5	20	541	0,13	0,08	0,18
6	30	4 683	0,44	0,37	1,48
7	42	47 293	0,79	0,58	140,76
8	56	545 835	1,90	1,25	> 22 200,00
9	72	7 087 261	8,49	5,82	–
10	90	102 247 563	66,14	38,36	–
11	110	1 622 632 573	507,61	321,99	–
12	132	28 091 567 595	–	2 932,90	–

Tabla 4.2: Resultados obtenidos al aplicar el algoritmo sobre los digrafos completos con distinta cantidad de nodos.

Nota: el Tiempo CE corresponde a la versión del programa con escritura, mientras que el Tiempo SE omite este aspecto. El digrafo completo de 12 nodos no fue evaluado porque el archivo de texto se asume demasiado grande. El de 11 nodos pesa 11,5 GB. El Tiempo `EtiquetadoSimple` es el obtenido con el programa que usa ese algoritmo para etiquetar, es decir, aquel que no usa divisiones. Para este último se aprecia que a medida que aumenta la cantidad de nodos se van notando mayores diferencias en el tiempo de ejecución, también se notan diferencias a la hora de almacenar la información pues para el caso $n = 8$ ya se contaba, pese a que no se terminó de calcular por completo, con un archivo de 538 000 KB; mientras que para el algoritmo `EtiquetadosUpdate`, para el mismo digrafo se tiene un archivo de salida de 14 000 KB. Todos los tiempos se encuentran en segundos.

4.6.2 Cadenas bidireccionales no circulares

Otros digrafos interesante a considerar son las cadenas bidireccionales no circulares como la mostrada en la Figura 4.18. Lo importante de ellas es que se dividen enormemente gracias a los puentes, mostrando las grandes ventajas que puede traer la división en comparación con el

algoritmo que utiliza el `EtiquetarSimple`. En la Tabla 4.3 mostramos los resultados obtenidos al realizar pruebas con ambos algoritmos.

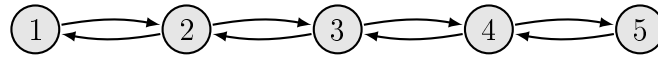


Figura 4.18: Ejemplo del digrafo cadena bidireccional $n = 5$.

Nodos	Arcos	$ \mathcal{S}(G, \text{lab}) $	Tiempo CE	
			EtiquetadoSimple	EtiquetadosUpdate
10	18	19 683	4,43	0,02
11	20	59 049	36,42	0,02
12	22	177 147	313,68	0,02
13	24	531 441	2 905,26	0,02
50	98	3^{49}	–	0,09
100	198	3^{99}	–	0,10
200	398	3^{199}	–	0,17

Tabla 4.3: Resultados obtenidos al aplicar el algoritmo sobre cadenas bidireccionales no circulares con distinta cantidad de nodos.

Nota: nuevamente el Tiempo CE corresponde al obtenido al utilizar el algoritmo `EtiquetadosUpdate`, mientras que el otro los obtenidos por `EtiquetadosSimple`. Todos los tiempos se encuentran en segundos. Se aprecian claramente las ventajas en este caso extremo, la división por puentes permite siempre trabajar con componentes de dos nodos solamente en `EtiquetadosUpdate`. En cambio con el otro algoritmo, al no obtener divisiones, para ejecutar forzar es necesario calcular los caminos reversos y reversos negativos entre cada par de nodos. También se aprecia nuevamente la compresión de datos, ya que para el mayor caso que se calculó el Tiempo 1, el archivo era de 197 000 KB mientras que para el mismo digrafo con el otro algoritmo el archivo de salida obtenido es de 2 KB.

4.7 Efectos en la salida frente a cambios en la entrada del algoritmo

En esta sección se evalúa cómo se ve afectada la respuesta entregada por el algoritmo en caso de que la entrada hubiese sido ligeramente modificada.

Siempre que hayan modificaciones en la entrada es importante analizar cuando se producen divisiones iniciales. Al realizar cambios en el etiquetado, es posible que algunas divisiones se mantengan. En esos casos se debe conservar la respuesta parcial entregada, puesto a que la ventaja de la división es la independencia entre las distintas componentes, es decir, las demás componentes no se ven afectadas por el resultado de otra. Para las divisiones que no se mantienen

habrá que calcular de nuevo la solución parcial asociada. En todo lo mencionado aquí estamos excluyendo a aquella división donde está el arco modificado.

Ahora pasamos a estudiar más en detalle algunas modificaciones específicas a la entrada:

4.7.1 Agregar una etiqueta a un arco sin etiqueta

Dado un digrafo etiquetado (G, lab) y $\mathcal{S}(G, \text{lab})$ el output entregado por el algoritmo `etiquetadosUpdate`, se plantea estudiar cómo varía dicho conjunto al modificar la función `lab` en un arco e tal que $\text{lab}(e) = \circ$.

Si el arco e estaba forzado en un comienzo, entonces hay dos opciones:

- Si era forzado con la misma etiqueta agregada, entonces se tiene el mismo conjunto solución.
- Si era forzado con la etiqueta opuesta, entonces el conjunto solución es vacío; pues el digrafo no sería de actualización.

En caso de no darse lo anterior, lo más simple es revisar cada una de las soluciones y verificar si en ellas el arco en cuestión tiene la etiqueta específica, en caso contrario se rechaza. Con el algoritmo generado las soluciones se entregan en grupos de acuerdo a las divisiones realizadas, así entonces lo que corresponde será ver en cada subconjunto si aparece la solución con el signo equivocado. De ser así, ver si está unido a otro subconjunto o está cruzado con otro, en el primer caso se elimina el subconjunto solución, y en el segundo se hace lo mismo con el subconjunto y todos aquellos que aparezcan cruzados a él.

4.7.2 Agregar un nuevo arco etiquetado

Supongamos que se agrega el arco (i, j) con $\text{lab}(i, j) = \oplus$, entonces:

- En el mejor caso ya existía un camino reverso de i a j , por lo que el conjunto solución es el mismo y sólo se debe agregar el nuevo arco etiquetado. Se puede hacer lo mismo si el arco agregado no participa en ningún ciclo prohibido (independiente de las distintas combinaciones posibles de etiquetas).
- Otro caso simple sería cuando inicialmente hay un camino reverso negativo de j a i , en ese caso el conjunto solución sería el conjunto vacío.

Si no se diera ninguno de los casos anteriores, se puede obtener lo esperado revisando en cada solución si existe un camino reverso negativo de j a i , en cuyo caso se descarta la solución obtenida. Considerando las divisiones basta hacer esto solo en el caso que los nodos j e i estén en la misma componente.

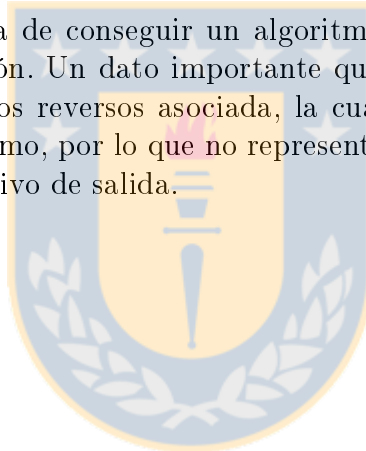
4.7.3 Cambiar una etiqueta \oplus por \ominus o viceversa

Consideremos el caso de cambiar $\text{lab}(i, j) = \oplus$ por $\text{lab}(i, j) = \ominus$:

- Buscar en la matriz inicial sin considerar el arco (i, j) si hay un camino reverso de i a j . En ese caso, la solución es el conjunto vacío.
- Si en la matriz inicial existiera un camino reverso negativo de j a i , entonces el conjunto solución inicial sería vacío. Por lo que habría que calcularlo desde cero.

El cambio en este caso es muy grande en general. Pese a que solo se cambio un signo, en el problema interesan mucho los caminos reversos y reversos negativos, y el cambio realizado suele realizar muchas modificaciones en ese aspecto. Esto implica que se agreguen y eliminen soluciones, lo que hace más compleja la situación que en casos anteriores. Por lo mismo parece normal buscar el conjunto solución desde cero.

Observación 4.1. Una manera de conseguir un algoritmo más robusto consiste en entregar más datos asociados a la solución. Un dato importante que se podría agregar a las soluciones parciales es la matriz de caminos reversos asociada, la cual está calculada en el momento de entregar la solución en el algoritmo, por lo que no representaría un gasto más que de tiempo de escritura y de espacio en el archivo de salida.



Aplicaciones

En este capítulo se habla de la implementación del algoritmo presentado en el Capítulo 4 y se aplica sobre dos redes que han sido estudiadas en la literatura, la red que representa el ciclo celular mamífero y la que hace lo mismo con el ciclo celular de la levadura de fisión. Luego de obtenidos los resultados, se analizan para indicar las principales ventajas de utilizar el algoritmo creado.

5.1 Implementación

El algoritmo fue programado en Java. El programa recibe como entrada un archivo de texto que en la primera línea contiene el número de nodos. Y en la siguiente los arcos con las respectivas etiquetas. Suponiendo que el digrafo es de n nodos, entonces los nodos en la entrada van de 0 a $n - 1$. Mientras que para las etiquetas se considera -1 para \ominus , 0 para \circ y 1 para \oplus . Así, la línea 0 1 0, indicaría que existe un arco de 0 a 1 que está sin etiqueta (\circ).

En la Figura 5.1a se muestra la entrada que se debe entregar a la implementación para resolver UDEP para el digrafo de la Figura 5.1b.

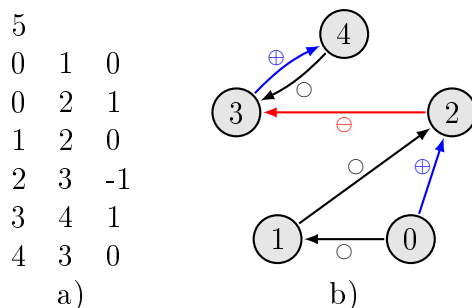


Figura 5.1: a) Archivo de entrada. b) Digrafo etiquetado (G_{lab}) asociado a la entrada.

La salida del programa, para un digrafo de entrada (G, lab) , es un archivo de texto que contiene el conjunto $\mathcal{S}(G, \text{lab})$. Estos se encuentran divididos de acuerdo a como los resolvió el algoritmo y codificados apropiadamente. Al final se entrega la cantidad total de etiquetados y el tiempo que demoró en ejecutarse el programa.

En cuanto a la estructura de datos, se utilizaron matrices de adyacencia para representar el digrafo. Aprovechando la orientación a objetos del lenguaje elegido, por cada nodo se generaron tres arreglos (matrices) para cada tipo de arco: positivos, negativos y no etiquetados. Para realizar los cambios de índices, que se dan principalmente por las divisiones, se usa un arreglo por nodo y se mantiene un entero que indica la profundidad de la recursión para identificar el valor actual que debe tomar el índice.

Otro tema importante es mencionar las características del computador utilizado para realizar los experimentos. Este cuenta con un procesador Intel Core i5-3317U CPU de 1.7GHz. Tiene 6 GB de memoria RAM, y se utilizó el sistema operativo Windows 8.1 de 64 bits.

5.2 Red del ciclo celular mamífero

Se estudia la red del ciclo celular mamífero, la cual fue presentada en [8] y fue ampliamente estudiada en [10], donde se entregan algunos resultados teóricos, pero su enfoque está basado esencialmente en simulaciones, usando todos los posibles esquemas de actualización.

A continuación se describe la red. La denotamos por G_M , tiene 10 nodos, que para simplificar la notación los denotamos como se muestra en la Tabla 5.1.

CycD $\equiv v_1$	p27 $\equiv v_6$
Rb $\equiv v_2$	Cdc20 $\equiv v_7$
E2F $\equiv v_3$	Cdh1 $\equiv v_8$
CycE $\equiv v_4$	UbcH10 $\equiv v_9$
CycA $\equiv v_5$	CycB $\equiv v_{10}$

Tabla 5.1: Notación para los nodos de la red del ciclo celular mamífero (G_M).

Las funciones de activación local están dadas en la Tabla 5.2 de acuerdo a lo descrito en [10]. El digrafo de interacción de G_M está dado en la Figura 5.2.

$$\begin{aligned}
f_{v_1}(x) &= x_{v_1} \\
f_{v_2}(x) &= (\neg x_{v_1} \wedge \neg x_{v_{10}}) \wedge ([\neg x_{v_4} \wedge \neg x_{v_5}] \vee x_{v_6}) \\
f_{v_3}(x) &= (\neg x_{v_2} \wedge \neg x_{v_5} \wedge \neg x_{v_{10}}) \vee (x_{v_6} \wedge \neg x_{v_2} \wedge \neg x_{v_{10}}) \\
f_{v_4}(x) &= x_{v_3} \wedge \neg x_{v_2} \\
f_{v_5}(x) &= (\neg x_{v_2} \wedge \neg x_{v_7} \wedge \neg(x_{v_8} \wedge x_{v_9})) \wedge (x_{v_3} \vee x_{v_5}) \\
f_{v_6}(x) &= (\neg x_{v_1} \wedge \neg x_{v_{10}}) \wedge ([\neg x_{v_4} \wedge \neg x_{v_5}] \vee [x_{v_6} \wedge \neg(x_{v_4} \wedge x_{v_5})]) \\
f_{v_7}(x) &= x_{v_{10}} \\
f_{v_8}(x) &= (\neg x_{v_5} \wedge \neg x_{v_{10}}) \vee x_{v_7} \vee (x_{v_6} \wedge \neg x_{v_{10}}) \\
f_{v_9}(x) &= \neg x_{v_8} \vee (x_{v_8} \wedge x_{v_9} \wedge [x_{v_7} \vee x_{v_5} \vee x_{v_{10}}]) \\
f_{v_{10}}(x) &= \neg x_{v_7} \wedge \neg x_{v_8}
\end{aligned}$$

Tabla 5.2: Funciones de activación local de la red del ciclo celular mamífero.

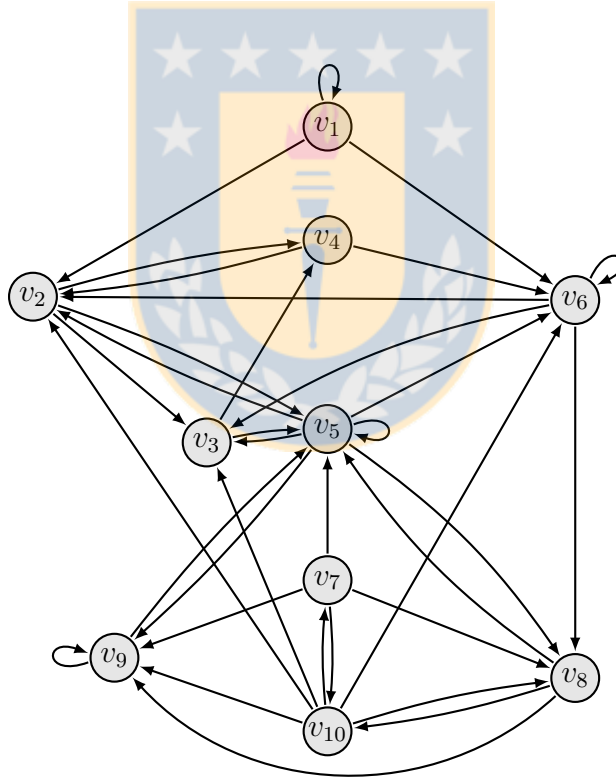


Figura 5.2: Digrafo de interacción de la red del ciclo celular mamífero.

Aplicamos nuestro algoritmo al digrafo sin etiquetas (G_M, lab_\circ) y encontramos todas sus extensiones totales válidas en 27,61 segundos. Existe un total de 466 712 elementos en $\mathcal{S}(G_M, \text{lab}_\circ)$.

Esta red actualizada de manera sincrónica, denotada por $N^p = (F, s^p)$, tiene un solo punto fijo y un ciclo límite de largo 7 como atractores, tal como se describe en Tabla 5.3.

Punto Fijo										
x^0	0	1	0	0	0	1	0	1	0	0

Ciclo límite										
$v \in V(G^F)$	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
x^0	1	0	1	0	0	0	0	1	1	0
x^1	1	0	1	1	0	0	0	1	0	0
x^2	1	0	1	1	1	0	0	1	0	0
x^3	1	0	0	1	1	0	0	0	0	0
x^4	1	0	0	0	1	0	0	0	1	1
x^5	1	0	0	0	1	0	1	0	1	1
x^6	1	0	0	0	0	0	1	1	1	0
x^7	1	0	1	0	0	0	0	1	1	0

Tabla 5.3: Atractores de la red del ciclo celular mamífero sincrónicamente actualizada.

En nuestro estudio nos interesa encontrar todas las funciones de etiqueta que mantienen un ciclo límite. Esto nos entrega ciertas restricciones sobre el etiquetado de ciertos arcos, así podemos etiquetar de manera fácil algunos de ellos para reducir el espacio de búsqueda. En el Ejemplo 5.1 se explica cómo se encuentran condiciones necesarias para mantener el ciclo y que induce a etiquetar algunos arcos.

Ejemplo 5.1. En las funciones de activación local f_{v_7} sabemos que cuando x_{v_7} es 1, $x_{v_{10}}$ es 1. En la sexta fila de la Tabla 5.3 vemos que x_{v_7} es 1 pero $x_{v_{10}}$ es 0, así que si $\text{lab}_M(10, 7) = \ominus$ el ciclo límite no sería posible, porque significaría que $s(v_{10}) < s(v_7)$ ($x_{v_{10}}$ es actualizado antes que x_{v_7}).

Análogamente podemos chequear $f_{v_{10}}$. En ese caso la función depende de v_7 y v_8 , específicamente cuando $x_{v_{10}}$ es 1, tanto x_{v_7} como x_{v_8} deben tener valor 0. Viendo la sexta fila, se concluye que $\text{lab}_M(7, 10) = \oplus$; sin embargo, usando la misma idea no es posible concluir la etiqueta de $(8, 10)$.

Finalmente es fácil ver que se debe etiquetar $(5,3)$; $(3,4)$; $(7,5)$; $(10,7)$; $(5,8)$; $(7,8)$; $(8,9)$ y $(7,10)$ como arcos positivos si buscamos funciones de etiqueta que preserven el ciclo mencionado. A este etiquetado le llamaremos lab_M .

El digrafo etiquetado obtenido con la información anterior es mostrado en la Figura 5.3.

Utilizando el algoritmo sobre el digrafo parcialmente etiquetado, se encuentran los 1440 elementos de $\mathcal{S}(G_M, \text{lab}_M)$ en 0,40 segundos. En [10] se estudiaron los diferentes digrafos de actualización que preservan el ciclo, pero se hizo tomando cada miembro de $\mathcal{S}(G_M, \text{lab}_\circ)$ como posible candidato, es decir, cada uno de los 466 712 elementos.

Considerando la simpleza para obtener lab_M a partir de lab_\circ y la enorme reducción tanto en tiempo de ejecución como en cantidad total de soluciones, podemos darnos cuenta de las ventajas obtenidas al usar el algoritmo.

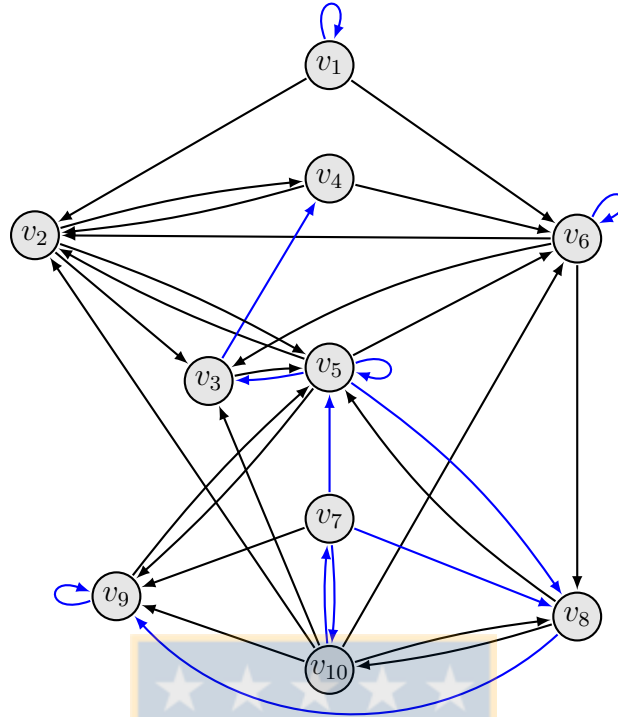


Figura 5.3: Red del ciclo celular mamífero parcialmente etiquetado (G_M, lab_M) .

Luego de esto se pasó a la etapa de verificar cuáles son los digrafos etiquetados que efectivamente preservan el ciclo entre los candidatos. Para esto se recuperaron las soluciones y por cada una de ellas se generó un esquema representante. Con este se iteró sobre la red para ver si se generaba el ciclo. Se comprobó que solo 216 de los 1 440 candidatos preservan el ciclo. Sin duda una cantidad muy pequeña en comparación de los 466 712 digrafos posibles que se generan en un comienzo con el digrafo sin etiquetas.

5.3 Red del ciclo celular de la levadura de fisión

En esta sección se estudia la red del ciclo celular de la levadura de fisión, la cual fue presentada en [5] y fue ampliamente estudiada en [11], donde se entregaron resultados teóricos y mediante simulaciones. Como se hizo con la red del ciclo celular mamífero, se muestran los resultados obtenidos al usar nuestro algoritmo para encontrar todas las extensiones totales que hacen que la red sea de actualización, para luego hacer lo mismo con aquellos que preservan el ciclo.

Ahora se describe la red del ciclo celular de la levadura de fisión (G_F) . Tiene 10 nodos, que para simplificar la notación serán denotados como se muestra en la Tabla 5.4.

Start $\equiv v_1$	Slp1 $\equiv v_6$
Sk $\equiv v_2$	Cdc2/Cdc13* $\equiv v_7$
Cdc2/Cdc13 $\equiv v_3$	Weel/Mik1 $\equiv v_8$
Ste9 $\equiv v_4$	Cdc25 $\equiv v_9$
Rum1 $\equiv v_5$	PP $\equiv v_{10}$

Tabla 5.4: Notación para los nodos de red del ciclo celular de la levadura de fisión.

Las funciones de activación local son dadas en la Tabla 5.5 y el digrafo de interacción se muestra en la Figura 5.4

$$\begin{aligned}
f_{v_1}(x) &= 0 \\
f_{v_2}(x) &= x_{v_1} \\
f_{v_3}(x) &= \neg x_{v_4} \wedge \neg x_{v_5} \wedge \neg x_{v_6} \\
f_{v_4}(x) &= (\neg x_{v_2} \wedge \neg x_{v_3} \wedge x_{v_4} \wedge \neg x_{v_7}) \vee (\neg x_{v_2} \wedge \neg x_{v_3} \wedge x_{v_4} \wedge x_{v_{10}}) \\
&\vee (\neg x_{v_2} \wedge \neg x_{v_3} \wedge \neg x_{v_7} \wedge x_{v_{10}}) \vee (\neg x_{v_2} \wedge x_{v_4} \wedge \neg x_{v_7} \wedge x_{v_{10}}) \\
&\vee (\neg x_{v_3} \wedge x_{v_4} \wedge \neg x_{v_7} \wedge x_{v_{10}}) \\
f_{v_5}(x) &= (\neg x_{v_2} \wedge \neg x_{v_3} \wedge x_{v_5} \wedge \neg x_{v_7}) \vee (\neg x_{v_2} \wedge \neg x_{v_3} \wedge x_{v_5} \wedge x_{v_{10}}) \\
&\vee (\neg x_{v_2} \wedge \neg x_{v_3} \wedge \neg x_{v_7} \wedge x_{v_{10}}) \vee (\neg x_{v_2} \wedge x_{v_5} \wedge \neg x_{v_7} \wedge x_{v_{10}}) \\
&\vee (\neg x_{v_3} \wedge x_{v_5} \wedge \neg x_{v_7} \wedge x_{v_{10}}) \\
f_{v_6}(x) &= x_{v_7} \\
f_{v_7}(x) &= \neg x_{v_4} \wedge \neg x_{v_5} \wedge \neg x_{v_6} \wedge \neg x_{v_8} \wedge x_{v_9} \\
f_{v_8}(x) &= (\neg x_{v_3} \wedge x_{v_8}) \vee (\neg x_{v_3} \wedge x_{v_{10}}) \vee (x_{v_8} \wedge x_{v_{10}}) \\
f_{v_9}(x) &= (x_{v_3} \wedge x_{v_9}) \vee (x_{v_3} \wedge \neg x_{v_{10}}) \vee (x_{v_9} \wedge \neg x_{v_{10}}) \\
f_{v_{10}}(x) &= x_{v_6}
\end{aligned}$$

Tabla 5.5: Funciones lógicas de la red de ciclo celular de la levadura de fisión.

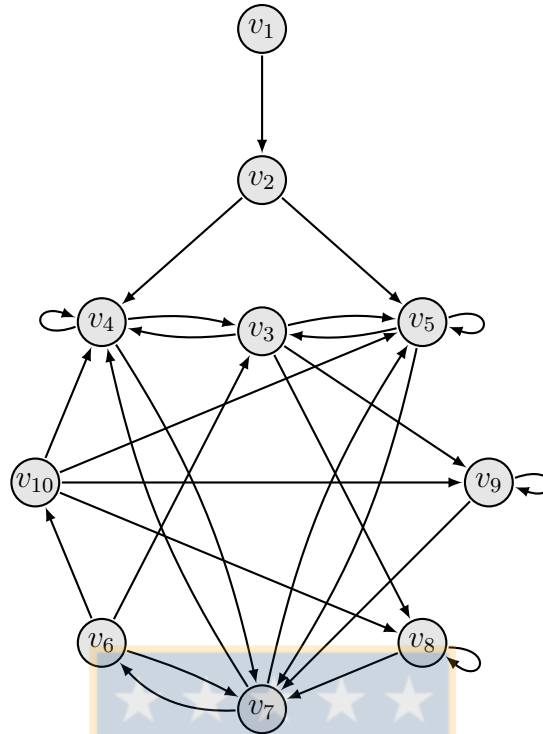


Figura 5.4: Digrafo de interacción de la red del ciclo celular de la levadura de fisión.

Al computar (G_F, lab_\circ) con nuestro algoritmo se encuentran las 80080 extensiones totales en 2,34 segundos. Esta red es más fácil de computar debido principalmente a su menor cantidad de arcos no etiquetados.

Ciclo límite										
$v \in V(G^F)$	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
x^0	0	0	0	0	0	0	0	0	1	1
x^1	0	0	1	1	1	0	1	1	0	0
x^2	0	0	0	0	0	1	0	0	1	0
x^3	0	0	0	0	0	0	0	0	1	1

Tabla 5.6: Ciclo límite de la red del ciclo celular de la levadura de fisión actualizada sincrónicamente.

Haciendo el análisis de vértices, explicado para obtener el etiquetado lab_M , se pueden etiquetar como positivos los siguientes arcos $(4,3)$; $(4,7)$; $(5,3)$; $(5,7)$; $(6,3)$; $(6,10)$; $(7,6)$; $(8,7)$ y $(9,7)$. Este etiquetado es llamado lab_F .

Ejemplo 5.2. Mirando en la Tabla 5.5 se ve que la función local f_{v_3} depende de x_{v_4} , x_{v_5} y x_{v_6} . Si x_{v_3} es 1, entonces x_{v_4} , x_{v_5} y x_{v_6} deben ser 0. Se observa que en la segunda fila de la Tabla 5.6 donde $x_{v_3} = 1$, $x_{v_4} = 1$, $x_{v_5} = 1$, entonces v_4 y v_5 no se pueden actualizar antes que v_3 .

Se ve en la cuarta fila que $x_{v_3} = 0$, y $x_{v_4} = x_{v_5} = 0$ en la tercera y cuarta fila, entonces para preservar el ciclo se necesita que $x_{v_6} = 1$. Por lo tanto $\text{lab}_F(6,3) = \oplus$.

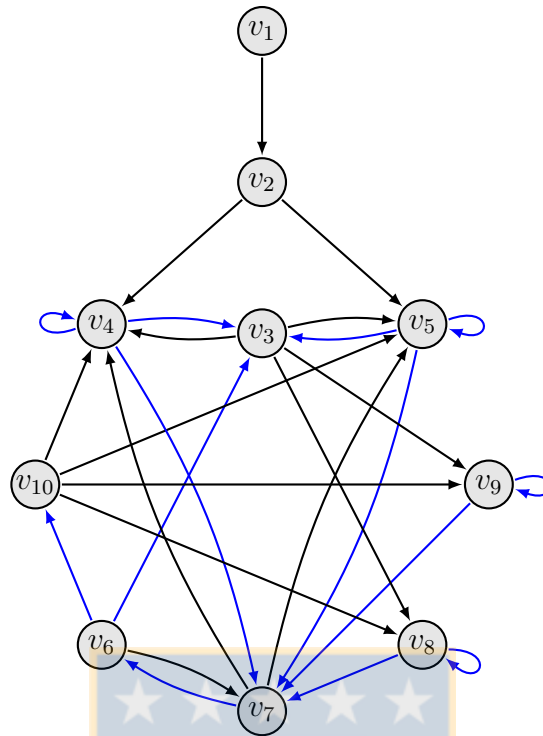


Figura 5.5: El digrafo parcialmente etiquetado de la red del ciclo celular de la levadura de fisión (G_F, lab_F) .

Al ejecutar la implementación del algoritmo con (G_F, lab_F) como input, se obtiene un tiempo de ejecución de 0,12 segundos para encontrar los 288 elementos de $\mathcal{S}(G_F, \text{lab}_F)$, lo que significa una gran reducción del total de candidatos. Y nuevamente, como se hizo la red del ciclo celular mamífero, se hace un análisis muy simple para obtener lab_F a partir de lab_\circ .

Finalmente, al analizar los digrafos etiquetados que efectivamente preservan el ciclo, de manera análoga a lo hecho para la red del ciclo celular mamífero, se encontró que solo 16 de los 288 cumplían con tal condición. Un número aún menor que en este caso en el que se pasó de 80 800 candidatos a 288 con el etiquetado parcial y luego a 16 etiquetados efectivos.

Red	Nodos	No etiquetados	Etiquetados	$ \mathcal{S}(G, \text{lab}) $	Tiempo [seg]
(G_M, lab_\circ)	10	31	0	466 712	27,61
(G_M, lab_M)	10	23	8	1 440	0,40
(G_F, lab_\circ)	10	22	0	80 080	2,34
(G_F, lab_F)	10	13	9	288	0,12

Tabla 5.7: Tabla resumen de resultados.

Recordar que el digrafo completo de 10 nodos (K_{10}), ya fue ejecutado con el programa y que arrojó que $|\mathcal{S}(C_{10}, \text{lab}_\circ)| = 102\,247\,563$ en 332,35 segundos.

Conclusiones

En esta tesis el interés principal fue estudiar el conjunto de extensiones totales que hacen que un digrafo parcialmente etiquetado sea de actualización. Esto con la idea de encontrar representantes para los esquemas de actualización que preservan alguna característica dinámica de la red Booleana, por ejemplo, un ciclo límite.

Dentro del algoritmo construido se puede discutir varios aspectos. Lo primero es analizar como evolucionó desde la primera versión más simple a la que se presenta en esta tesis. Considerando que el problema de conteo asociado es $\#P$ -completo como se demostró en el Capítulo 3, se sabe que UDEP es difícil de tratar computacionalmente y que no se conocen algoritmos polinomiales que lo resuelvan. Sin embargo, se buscó construir un algoritmo que mejorara al de fuerza bruta que consiste en generar todos los etiquetados y comprobar si son o no de actualización.

Para la fuerza bruta solo es necesario el algoritmo Verificar y uno que permita construir los etiquetados. La primera mejora que se realizó fue incorporar el concepto de forzar presentado en la Sección 4.2. Esto se hizo incluyendo el Forzar luego de Verificar y después de cada vez que se etiquetara un arco. De esta manera y a pesar de la simpleza de Forzar, que es $O(|V(G)|^3)$, se logró que disminuyeran los tiempos de ejecución mediante la reducción del espacio de búsqueda, pues así se descartan lo antes posible las soluciones no válidas y además no es necesario verificar que el etiquetado es de actualización en ningún otro momento.

Luego, se agregó la idea de reducir el digrafo. Esta operación permite disminuir los tiempos de respuesta cuando en el digrafo inicial existe una componente fuertemente conexa positiva. Se demostró que a partir de las soluciones del digrafo reducido es posible recuperar las del inicial. Este proceso también es polinomial, ya que está basado en el algoritmo de Tarjan para componentes fuertemente conexas, es decir, es $O(|V(G)| + |A(G)|)$. Finalmente, se decidió agregar la reducción solo una vez y al comienzo en el algoritmo EtiquetadosUpdate. Esto es debido a que la estructura de datos utilizada no es óptima con dicha operación, además de los cambios de índice hay que crear nuevos nodos, lo cual es costoso si se hiciera durante la recursividad. Como trabajo futuro respecto a este tema se puede buscar una estructura de datos más acorde para

implementar un algoritmo que utilice más veces la reducción. Esta rutina habría que aplicarla cada vez que se etiqueta un arco positivo (aunque sea por forzar). De esta manera se trabajaría siempre con digrafos reducidos. Lo cual llevaría a, cuando la reducción es efectiva, trabajar con menos nodos y arcos. Sería interesante comparar los resultados obtenidos con esta nueva variante.

La última idea que se agregó al algoritmo son las divisiones presentadas en la Sección 4.4. En esta se presenta la división por puentes y por componentes fuertemente conexas, ambas permiten separar el problema en subproblemas más pequeños que se resuelven de manera independiente. Los dos algoritmos creados son polinomiales, específicamente son $O(|V(G)| + |A(G)|)$, ya que están basados en el algoritmo de Tarjan para búsqueda en profundidad y se aplican sobre grafos que pueden ser generados también polinomialmente a partir del digrafo original. Se decidió hacer primero división por puentes y luego tomar una a una las componentes resultantes y aplicarles división por componentes fuertemente conexas, esto debido a que la primera división se hace sobre el grafo fundamental asociado y la segunda sobre el digrafo reverso extendido, y la cantidad de aristas siempre será menor que la de arcos.

Otra decisión que se tomó fue etiquetar un arco luego de intentar realizar el segundo tipo de división. Existía la posibilidad de tratar de volver a dividir. Sin embargo, al ser poco probable que se vuelva a dividir, se decidió descartar esa posibilidad. Se creyó que en la mayoría de los casos será más el tiempo que se perderá intentando dividir por segunda vez que lo que se ganará con las divisiones exitosas.

Hay que recordar que los algoritmos usados para forzar, reducir y dividir podrían dejar la entrada sin modificaciones; es decir, no ayudar para ciertos digrafos. Sería importante como trabajo futuro identificar cuándo las rutinas mencionadas tienen mayor probabilidad de aportar en la reducción del tiempo. De esta manera se podrían usar de manera más efectiva.

Como se vio en el Capítulo 5 la implementación hecha en Java también es mejorable en varios aspectos. Por ejemplo, si se considera mantener la representación del digrafo por matrices de adyacencias tal vez sería bueno implementar el algoritmo en MATLAB, ya que este lenguaje es muy bueno trabajando con estas. También sería apropiado, porque la rutina más costosa (más allá de lo exponencial de etiquetar) está dada por el forzar y la construcción de la matriz de caminos reversos. Además sería más simple manejar los subdigrafos generados por las divisiones. Otra opción sería encontrar una estructura de datos más apropiada. Para el caso de los algoritmos diseñados resulta más eficiente utilizar listas de adyacencia para representar el digrafo. Esta opción se descartó por la complejidad que tiene a la hora de modificar las listas durante el proceso recursivo.

Como trabajo futuro lo que se espera es poder continuar la reducción del espacio de búsqueda del conjunto solución en cuanto al conjunto de etiquetados que preservan alguna característica dinámica de la red. Se podrían buscar combinaciones prohibidas, es decir, combinaciones de etiquetas en ciertos arcos que al darse no mantendrían las características que se buscan. Con esta información adicional se tienen nuevas restricciones por verificar a medida que se va etiquetando. Considerando esto habría que crear un `Forzar2` y `Verificar2`. Tratando de mantener el resto del algoritmo presentado, la idea sería usar el `Verificar2` para comprobar que no se violen las nuevas restricciones al hacer `Forzar`, luego se hace el `Forzar2` y se comprueba la condición de digrafo

de actualización con `Verificar` y se repite el proceso hasta que no haya modificaciones. Después se etiqueta recursivamente como en el algoritmo creado en la tesis y se repite el proceso. Es importante hacer notar que en este caso no será posible utilizar las divisiones explotadas en el algoritmo `EtiquetadosUpdate` directamente. Esto se debe a que por la forma de resolver el problema sería imposible verificar el cumplimiento de las nuevas restricciones independiente de las divisiones, ya que en ocasiones ciertos miembros de los conjuntos de arcos quedarían en subproblemas distintos y por ende se resolverían por separado.

Recordando el problema de las orientaciones acíclicas usado para el cálculo de la complejidad en la Sección 3.1. Queda claro que con el algoritmo realizado se puede resolver también el mencionado. Sin embargo, habría que hacer una modificación para que al ingresar el grafo lo transformara al digrafo acíclico explicado en la reducción. El algoritmo de reducción estaría de más, ya que el digrafo generado inicialmente es acíclico. Lo demás se podría mantener y habría que ajustar también la salida para que se hiciera la entrega del digrafo reverso, que correspondería a una orientación acíclica.



Bibliografía

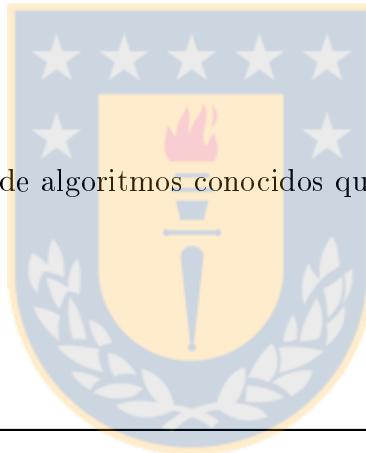
- [1] J Aracena, J Demongeot, E Fanchon, and M Montalva. On the number of different dynamics in Boolean networks with deterministic update schedules. *Mathematical Biosciences*, 2013.
- [2] Julio Aracena, Eric Goles, Andrés Moreira, and Lilian Salinas. On the robustness of update schedules in Boolean networks. *Biosystems*, 97(1):1–8, 2009.
- [3] 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.
- [4] 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.
- [5] M. Davidich and S. Bornholdt. Boolean model predicts cell cycle sequence of fission yeast. *PLoS ONE*, 3:e1672, 2008.
- [6] Jacques Demongeot, Adrien Elena, and Sylvain Sené. Robustness in regulatory networks: a multi-disciplinary approach. *Acta Biotheoretica*, 56(1-2):27–49, 2008.
- [7] Adrien Elena. *Robustesse des réseaux d'automates booléens à seuil aux modes d'itération*. PhD thesis, Université Joseph Fourier, Grenoble, Francia, 2009.
- [8] A. Fauré, A. Naldi, C. Chaouiya, and D. Thieffry. Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. *Bioinformatics*, 22:124–131, 2006.
- [9] 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.
- [10] G. Fogel, E. Goles, M. Montalva, and G. Ruz. Dynamical and topological robustness of the mammalian cell cycle network: A reverse engineering approach. *BioSystems*, 115:23–32, 2014.

- [11] E. Goles, M. Montalva, and G. Ruz. Deconstruction and dynamical robustness of regulatory networks: Application to the yeast cell cycle networks. *Bulletin of Mathematical Biology*, 75:939–966, 2013.
- [12] Eric Goles and Mathilde Noual. Disjunctive networks and update schedules. *Advances in Applied Mathematics*, 48(5):646–662, 2012.
- [13] Eric Goles and Lilian Salinas. Comparison between parallel and serial dynamics of Boolean networks. *Theoretical Computer Science*, 396(1):247–253, 2008.
- [14] Anders Å Hansson, Henning S Mortveit, and Christian M Reidys. On asynchronous cellular automata. *Advances in Complex Systems*, 8(04):521–538, 2005.
- [15] 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.
- [16] Stuart Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22(3):437–467, 1969.
- [17] Stuart Kauffman. *The origins of order: Self organization and selection in evolution*. Oxford University Press, 1993.
- [18] Nathan Linial. Hard enumeration problems in geometry and combinatorics. *SIAM Journal on Algebraic Discrete Methods*, 7(2):331–335, 1986.
- [19] Marco Antonio Montalva Medel. *Feedback set problems and dynamical behavior in regulatory networks*. PhD thesis, Universidad de Concepción, Concepción, Chile, 2011.
- [20] Henning S Mortveit and Christian M Reidys. Discrete, sequential dynamical systems. *Discrete Mathematics*, 226(1):281–295, 2001.
- [21] Sungju Park and Sheldon B Akers. An efficient method for finding a minimal feedback arc set in directed graphs. In *Circuits and Systems, 1992. ISCAS'92. Proceedings., 1992 IEEE International Symposium on*, volume 4, pages 1863–1866. IEEE, 1992.
- [22] François Robert. *Discrete iterations: a metric study*. Springer-Verlag New York, 1986.
- [23] Lilian Salinas. *Estudio de modelos discretos: estructura y dinámica*. PhD thesis, Universidad de Chile, Santiago, Chile, 2008.
- [24] Christoph Schmal, Tiago P Peixoto, and Barbara Drossel. Boolean networks with robust and reliable trajectories. *New Journal of Physics*, 12(11):113054, 2010.
- [25] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.
- [26] René Thomas. Boolean formalization of genetic control circuits. *Journal of Theoretical Biology*, 42(3):563–585, 1973.
- [27] René Thomas and Richard d’Ari. *Biological feedback*. CRC Press, 1990.

7.1 Algoritmos

Se presentan los pseudocódigos de algoritmos conocidos que han sido adaptados para ser utilizados de manera conveniente.

7.1.1 Floyd - Warshall



Algoritmo 8 FloydWarshall

Entrada: Un grafo G y la función w que entrega el peso de cada arco.

Salida: El peso mínimo entre cada par de vértices.

```

1: Sea  $M$  una matriz  $|V| \times |V|$  de distancias mínimas inicializada con  $\infty$ 
2: for all  $v \in V$  do
3:    $M[v][v] \leftarrow 0$ 
4: for all  $(u, v) \in A(G)$  do
5:    $M[u][v] \leftarrow w(u, v)$ 
6: for  $k = 1$  to  $|V|$  do
7:   for  $i = 1$  to  $|V|$  do
8:     for  $j = 1$  to  $|V|$  do
9:       if  $M[i][j] > M[i][k] + M[k][j]$  then  $M[i][j] \leftarrow M[i][k] + M[k][j]$ 
10: return  $M$ 

```

El algoritmo se realiza con una complejidad de $O(|V|^3)$. Este puede ser modificado fácilmente para considerar la existencia de caminos reversos y reversos negativos.

7.1.2 Tarjan

En [25] Robert Tarjan presentó una serie de algoritmos basados en la búsqueda en profundidad; que permiten, entre otras cosas, encontrar las componentes fuertemente conexas en un digrafo y los puentes en un grafo como se presenta a continuación:

Algoritmo 9 CFC

Entrada: Un digrafo $G = (V, A)$

Salida: Los conjuntos de nodos que inducen a sus componentes fuertemente conexas $\{V(G_1), \dots, V(G_n)\}$ que terminan contenidos en la variable global Sol

```

1: index  $\leftarrow$  0
2:  $i \leftarrow$  1
3:  $S \leftarrow \emptyset$ 
4:  $Sol \leftarrow \emptyset$ 
5: for all  $v \in V(G)$  do
6:   if  $pre[v]$  no está definido then
7:     FuertementeConexo( $v$ )
   return  $Sol$ 

```

Algoritmo 10 FuertementeConexo

Entrada: Un nodo $v \in V(G)$ de un digrafo G

Salida:

```

1:  $pre[v] \leftarrow$  index
2:  $low[v] \leftarrow$  index
3: index  $\leftarrow$  index + 1
4: Ingresar  $v$  a la pila  $S$ 
5: for all  $w$  sucesor de  $v$  do
6:   if  $pre[w]$  no está definido then
7:     FuertementeConexo( $w$ )
8:      $low[v] \leftarrow \min(low[v], low[w])$ 
9:   else if  $w \in S$  then
10:     $low[v] \leftarrow \min(low[v], pre[w])$ 
11: if  $low[v] = pre[v]$  then
12:   repeat
13:      $w \leftarrow$  Sacar de la pila  $S$ 
14:     Agregar  $w$  a  $V(G_i)$ 
15:   until  $v = w$ 
16:    $Sol \leftarrow Sol \cup \{V(G_i)\}$ 
17:    $i \leftarrow i + 1$ 

```

La complejidad de este algoritmo es de $O(|V| + |A|)$. Este servirá para realizar una reducción del grafo, verificar si es de actualización, dividir por puentes y componentes fuertemente conexas.

Algoritmo 11 Puentes**Entrada:** Un digrafo $G = (V, A)$ **Salida:** Los conjuntos de nodos que inducen a sus componentes fuertemente conexas y sus puentes $\{V(G_1), \dots, V(G_n)\}$ que terminan contenidos en la variable global Sol

```

1: index  $\leftarrow$  0
2:  $i \leftarrow$  1
3:  $S \leftarrow \emptyset$ 
4:  $Sol \leftarrow \emptyset$ 
5: for all  $v \in V(G)$  do
6:   if  $pre[v]$  no está definido then
7:     FuertementeConexoPuentes( $v$ )
return  $Sol$ 

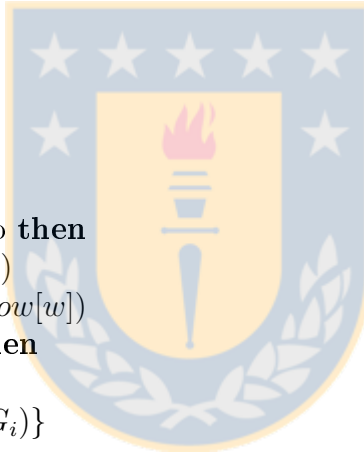
```

Algoritmo 12 FuertementeConexoPuentes**Entrada:** Un nodo $v \in V(G)$ de un grafo G **Salida:**

```

1:  $pre[v] \leftarrow$  index
2:  $low[v] \leftarrow$  index
3: index  $\leftarrow$  index + 1
4: Ingresar  $v$  a la pila  $S$ 
5: for all  $w$  sucesor de  $v$  do
6:   if  $pre[w]$  no está definido then
7:     FuertementeConexo( $w$ )
8:      $low[v] \leftarrow \min(low[v], low[w])$ 
9:     if  $low[w] = pre[w]$  then
10:       $V(G_i) = \{v, w\}$ 
11:       $Sol \leftarrow Sol \cup \{V(G_i)\}$ 
12:       $i \leftarrow i + 1$ 
13:   else if  $w \in S$  then
14:      $low[v] \leftarrow \min(low[v], pre[w])$ 
15: if  $low[v] = pre[v]$  then
16:   repeat
17:      $w \leftarrow$  Sacar de la pila  $S$ 
18:     Agregar  $w$  a  $V(G_i)$ 
19:   until  $v = w$ 
20:    $Sol \leftarrow Sol \cup \{V(G_i)\}$ 
21:    $i \leftarrow i + 1$ 

```



La diferencia en este algoritmo en comparación a CFC es que se agregan como componentes los nodos que forman puentes tal como se ve en la línea 10 de FuertementeConexoPuentes.