Identificación eficiente de componentes fuertemente conexas en la causalización de grandes modelos orientados a objetos

Denise Marzorati[1], Joaquín Fernández[1] y Ernesto Kofman[1]

FCEIA-UNR, Pellegrini 250, Rosario, 2000, Santa Fe, Argentina
CIFASIS-CONICET UNR, 27 de Febrero 210 bis, Rosario, 2000, Santa Fe, Argentina
marzorati,fernandez,kofman@cifasis-conicet.gov.ar

Abstract. Este artículo describe un algoritmo para encontrar componentes fuertemente conexas (CFC) de grafos dirigidos que presentan patrones repetitivos en su estructura. Dichos grafos pueden ser definidos de manera compacta utilizando Set-Based Graphs (SBG). Aprovechando esta representación, el costo computacional del algoritmo propuesto no varía al incrementar la cantidad de vértices y aristas presentes en el mismo patrón.
Su principal aplicación es la detección de lazos algebraicos durante la causalización de modelos orientados a objetos que cuentan con arreglos de variables y ecuaciones. De este modo el algorimo propuesto preserva y retorna un resultado compacto, lo cual es útil para generar código de simulación eficiente.
Además de presentar el algoritmo en cuestión, el artículo incluye resultados teóricos relativos a su complejidad computacional. Por último se analiza el rendimiento real del algoritmo en dos modelos con distintos tipos de lazos algebraicos.

Palabras clave: modelos a gran escala, componentes fuertemente conexas, Modelica, ecuaciones algebraico-diferenciales

Efficient detection of strongly connected components for the causalization of large object oriented models

Abstract. This article describes an algorithm that finds strongly connected components (SCC) in directed graphs with repetitive patterns in its structure. These graphs can be compactly defined using the Set-Based Graphs (SBG) representation. This way, the computational cost of the proposed algorithm is independent of the number of verticesand edges present in the same pattern.

The main application is the detection of algebraic loops during the causalization of object oriented models defined using arrays of variables and equations. This way the algorithm preserves and returns a compact result, which is convenient to generate efficient simulation code.
Besides the description of the proposed procedure, the article includes theoretical results regarding its time complexity. Finally, two examples of two models with different types of algebraic loops are presented at the end of this work.

Keywords: strongly connected components, causalization, Modelica, algebraic loops, Set-Based Graphs

1 Introducción

En numerosos casos la Teoría de Grafos constituye un herramienta eficaz para analizar la estructura de distintos sistemas, independientemente de sus características y componentes. Algunos de ellos requieren evaluar el grado de dependencia entre sus diversas partes. En tales contextos, las relaciones subyacentes entre componentes pueden representarse mediante un grafo, para luego obtener sus Componentes Fuertemente Conexas (CFC).

Un ejemplo de aplicación es la identificación de lazos algebraicos, es decir, grupos de variables que necesitan ser despejadas al mismo tiempo, como parte del proceso de conversión de un sistema de Ecuaciones Algebraico-Diferenciales (EDAs) en Ecuaciones Diferenciales Ordinarias (EDOs) para su simulación, tal como se explica en Cellier and Kofman, 2006. Esta traducción puede modelarse como un problema de grafos representando cada variable con un vértice, y agregando una arista de acuerdo al orden en que deben ser resueltas. La existencia de una componente fuertemente conexa indica un grupo de variables que deben resolverse conjuntamente (lazo algebraico).

Se han propuesto diversos algoritmos para abordar la detección de CFCs tales como Gabow, 2000; Sharir, 1981; Tarjan, 1972. Sin embargo, en el mejor de los casos, el costo computacional de estos algoritmos depende de la cantidad de vértices y aristas del grafo. Aunque estos resultados son adecuados para muchos casos, la causalización de un sistema con millones de ecuaciones sobrepasa la capacidad de las técnicas existentes.

Cabe destacar que la mayoría de los sistemas a gran escala se definen utilizando estructuras repetitivas. Esta característica puede ser aprovechada usando una representación compacta, como la propuesta por el enfoque de *Set Based Graph* (Grafos Basados en Conjuntos, SBG), definido en Zimmermann et al., 2019. Allí se propone agrupar vértices y aristas en conjuntos definidos de forma intensiva, lo que permite operar sobre ellos con costos computacionales independientes del tamaño de dichos conjuntos.

El presente trabajo introduce un nuevo algoritmo que opera con SBGs para identificar CFCs de manera compacta. Luego, se definen condiciones suficientes bajo las cuales el procedimiento calcula las CFCs de en un SBG con costo com-

putacional constante respecto del tamaño de los conjuntos de vértices, y de aristas.

2 Motivación

El sistema de EDAs correspondiente al circuito eléctrico de la Figura 1 puede ser expresado en el lenguaje Modelica (para mayores detalles ver Fritzson and Bunus, 2002; Fritzson and Engelson, 1998) escribiendo el código del Listado 1.1.

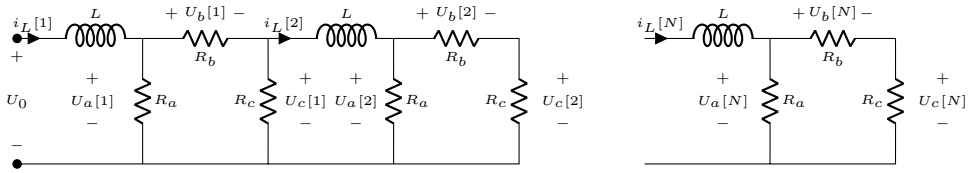


Fig. 1: Circuito Eléctrico.

La simulación eficiente de este sistema de EDAs requiere que el mismo sea *causalizado*. Primero, las ecuaciones deben ser ordenadas horizontalmente, emparejando variables con la ecuación de la cual serán despejadas, y verticalmente para que cada variable sea utilizada solo luego de conocer su valor. El Listado 1.2 muestra una de las opciones posibles de causalización para el modelo del Listado 1.1. De este modo, las derivadas de las variables de estado `der(iL[i])` pueden ser computadas como una función de las variables de estado `iL[i]`. Es decir, se ha transformado el modelo para definirlo en base a ODEs en vez de EDAs.

Es muy común que estos modelos presenten *lazos algebraicos*, es decir, grupos de variables que deben ser calculadas simultáneamente. Precisamente este es el caso del modelo del Listado 1.1, donde `Ua[i]`y `Uc[i]` conforman N lazos algebraicos que deben resolverse conjuntamente para obtener el conjunto de ecuaciones explícitas del Listado 1.2. Se debe notar que las ecuaciones de donde se despejan `Ua[i]` y `Uc[i]` de dicho modelo son el resultado de resolver N sistemas de dos ecuaciones y dos variables.

Para lograr la forma explícita de las ecuaciones del Listado 1.2, se deben identificar los lazos algebraicos en primer lugar. Así, un paso intermedio en la causalización del sistema es su transformación en el modelo del Listado 1.3 donde cada lazo es detectado e identificado (en este caso, con comentarios de la forma `Loop i`).

La detección de lazos algebraicos es equivalente a encontrar las CFC de un grafo definido en base al modelo, para lo cual existen algoritmos eficientes de la Teoría de Grafos. En este caso, considerando $N = 10$, el modelo cuenta con 30 ecuaciones, por lo que el grafo correspondiente tendrá 30 vértices y 39 aristas, sin representar desafío alguno para cualquier algoritmo de CFC tradicional.

```
model Circuit
  constant Integer N = 10;
  Real iL[N], Ua[N], Uc[N];
  parameter Real Ra=1, Rb=1, Rc=1, L=1,
       U0=1;
equation
  der(iL[1]) = U0 - Ua[1];
                        // Eq1
  for i in 2:N loop
    L*der(iL[i]) = Uc[i-1] - Ua[i];
                // Eq2
  end for;
  for i in 1:N-1 loop
    Ua[i] = Rb*iL[i+1] + Uc[i]*Rb/Rc;
              // Eq3
  end for;
  for i in 1:N-1 loop
    iL[i] = Ua[i]/Ra + Uc[i]/Rc + iL[i
         +1]; // Eq4
  end for;
  iL[N] = Ua[N]/Ra + Uc[N]/Rc;
                    // Eq5
  Ua[N] = Uc[N]*Rb/Rc;
                           // Eq6
end Circuit;
```

Listado 1.1: Ecuaciones del Circuito.

```
model CircuitB
  constant Integer N = 10;
  Real iL[N], Ua[N], Uc[N];
  parameter Real Ra=1, Rb=1, Rc=1, L=1,
       U0=1;
equation
  Uc[1]=Ra*Rc/(Ra+Rb) iL[1] - Rc * iL[2]
  Ua[1]=Ra*Rb/(Ra+Rb) * iL[1]
  der(iL[1]) = U0 - Ua[1];
  for i in 2:N-1 loop
    Uc[i]=Ra*Rc/(Ra+Rb) iL[i] - Rc * iL[
         i+1]
    Ua[i]=Ra*Rb/(Ra+Rb) * iL[i]
  end for
  for i in 1:N-1 loop
    der(iL[i]) = (Uc[i-1] - Ua[i])/L;
  end for;
  Uc[N]=Ra*Rc/(Ra+Rb) iL[N]
  Ua[N]=Ra*Rb/(Ra+Rb) * iL[N]
  der(iL[N]) = (Uc[N-1] - Ua[N])/L;
end CircuitB;
```

Listado 1.2: Ecuaciones del Circuito Causalizadas.

Sin embargo, como la complejidad computacional de dichos algoritmos depende del tamaño del grafo, al incrementar el valor de N, el costo se torna prohibitivo. Más aún, usando algoritmos tradicionales de CFC, el código resultante no mantendrá la forma compacta vista en el Listado 1.2, lo cual puede ser un problema para etapas posteriores del proceso de compilación, que deberán tratar con un modelo con $3N$ ecuaciones.

Este desafío sigue representado un obstáculo importante para los compiladores de Modelica a la hora de trabajar con modelos a gran escala. En la actualidad, herramientas populares tales como Dymola Brück et al., 2002, OpenModelica Fritzson et al., 2020, y Wolfram SystemModeler Rozhdestvensky et al., 2020 no son capaces de generar código de simulación de sistemas con una cantidad grande de estados.

La línea de trabajo desarrollada en Zimmermann et al., 2019 introduce el concepto de Grafos Basados en Conjuntos (Set–Based Graphs o SBG), donde se intenta desarrollar todo el proceso de causalización manteniendo la representación compacta. Sin embargo, la solución no es general, y en muchos casos los algoritmos propuestos terminan expandiendo los arreglos involucrados.

Este trabajo extiende esta línea de investigación definiendo un nuevo algoritmo general para hallar CFC en SBGs.

3 Conceptos Previos

Esta sección explica el proceso de causalización de EDAs, su relación con la Teoría de Grafos, y la definición de Grafos Basados en Conjuntos.

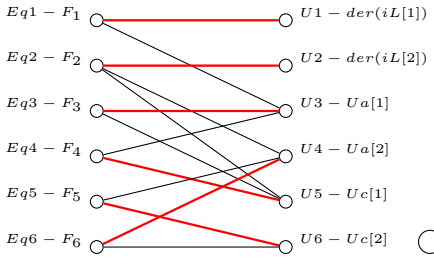
```
model CircuitC
  constant Integer N = 10;
  Real iL[N], Ua[N], Uc[N];
  parameter Real Ra = 1, Rb = 1, Rc = 1, L = 1, U0 = 1;
equation
  0 = Rb*iL[2] + Uc[1]*Rb/Rc - Ua[1];          // Loop 1
  0 = Ua[i]/Ra + Uc[i]/Rc + iL[2] - iL[1];     // Loop 1
  der(iL[1]) = U0 - Ua[1];
  for i in 2:N-1 loop
    0 = Rb*iL[i+1] + Uc[i]*Rb/Rc - Ua[i];      // Loop i
    0 = Ua[i]/Ra + Uc[i]/Rc + iL[i+1] - iL[i]; // Loop i
  end for
  for i in 1:N-1 loop
    der(iL[i]) = (Uc[i-1] - Ua[i])/L;
  end for;
  0 = Ua[N]/Ra + Uc[N]/Rc - iL[N];             // Loop N
  0 = Uc[N]*Rb/Rc - Ua[N];                     // Loop N
  der(iL[N]) = (Uc[N-1] - Ua[N])/L;
end CircuitC;
```

Listado 1.3: Ecuaciones del Circuito con Lazos Algebraicos.

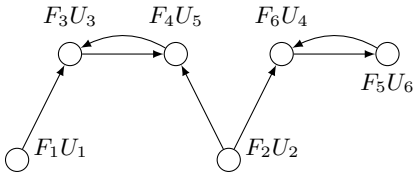
3.1 Causalización de EDAs y Teoría de Grafos

El proceso de causalización usualmente está compuesto por los siguientes pasos:

1. Se designa una ecuación para cada variable de la cual será despejada. Esto es equivalente encontrar un matching máximo en un grafo bipartito. En dicho grafo se agrega un vértice izquierdo por cada variable, y uno derecho por cada ecuación. Además, se añade una arista por cada variable que aparece en una ecuación, entre sus respectivos vértices. La Figura 2a muestra el grafo bipartito que se corresponde con el modelo del Listado 1.1.
   (a) Si el matching no satura todos los vértices, el sistema es de índice alto, por lo que es necesario aplicar un algoritmo de reducción de índice, como por ejemplo el de Pantelides, 1988. El procedimiento devuelve un nuevo sistema de ecuaciones, sobre el cual se debe reiniciar el proceso de causalización.
2. Se colapsan los pares de vértices matcheados en vértices únicos, y se dirigen las aristas no matcheadas de ecuaciones hacia incógnitas. De este modo, una arista (u, v) indica que la variable v debe ser calculada antes que u. Si dos vértices u, v pertenecen a la misma CFC, entonces deben ser computados en forma simultánea (lazo algebraico). La Figura 2b muestra el grafo dirigido del modelo del Listado 1.1.
3. Se deben encontrar las CFCs del grafo obtenido en el paso previo (para así detectar los lazos algebraicos).
4. Se busca un orden topológico para las CFCs para ordenar verticalmente las ecuaciones causalizadas. Este paso y el anterior podrían ejecutarse en simultáneo, por ejemplo con el algoritmo de Tarjan, 1972.
5. El código generado para la simulación se puede hacer más eficiente si se resuelven los lazos algebraicos (esto es posible si el sistema es lineal, como en el modelo del Listado 1.2), o bien reduciendo el número de variables de iteración mediante algoritmos de rasgado.



(a) Grafo Bipartito y un posible Matching Máximo



(b) Grafo Dirigido

Fig. 2: Construcción de los Grafos del modelo del Listado 1.1 con $N = 2$.

3.2 Set-Based Graphs

Los resultados obtenidos en este trabajo son posibles gracias a la utilización de *Set–Based Graphs* definidos por primer primera vez en Zimmermann et al., 2019, y luego revisitados en Marzorati et al., 2022, 2024. Los SBGs proponen una representación compacta para grafos, agrupando vértices y aristas en distintos conjuntos (esta agrupación no tiene relación con el concepto de hipergrafos). A continuación, presentamos las principales definiciones:

Definición 1 (Set–Based Graph). *Dado un grafo $G = (V, E)$, un SBG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ asociado al mismo se define como:*

- *Una partición $\mathcal{V} = \{V^1, \ldots, V^p\}$ de V a cuyos elementos llamaremos* Set–Vertices.
- *Una partición $\mathcal{E} = \{E^{i_1,j_1}, ..., E^{i_k,j_k}\}$ de E a cuyos elementos llamaremos* Set–Edges*, donde $E^{i,j} = \{\{u, v\} : u \in V_i \wedge v \in V_j\}$.*

Esta definición resulta útil para definir representaciones SBG para otras clases de grafos, tales como los grafos bipartitos y dirigidos, dadas por las Definiciones 2 y 3 respectivamente.

Definición 2 (Set–Based Graph Bipartito). *Un Set–Based Graph bipartito es un Set–Based Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ para el cual existe una partición de $\mathcal{V}$ compuesta por dos elementos $\mathcal{V}_1$, $\mathcal{V}_2$ tales que para cada set–edge $E^{a,b} \in \mathcal{E}$, si $V^a \in \mathcal{V}_i$ entonces $V^b \notin \mathcal{V}_i$.*

Definición 3 (Set–Based Graph Dirigido). *Dado un grafo dirigido $G = (V, E)$, un Set–Based Graph dirigido $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ asociado al mismo se define del siguiente modo:*

- *Una partición $\mathcal{V} = \{V^1, \ldots, V^p\}$ de V a cuyos elementos llamaremos* Set–Vertices.
- *Una partición $\mathcal{E} = \{E^{i_1,j_1}, ..., E^{i_k,j_k}\}$ de E a cuyos elementos llamaremos* Set–Edges*, donde $E^{i,j} = \{(u, v) : u \in V_i \wedge v \in V_j\}$.*

3.3 Representación SBG

Una manera de representar de forma compacta los SBGs dirigidos es la siguiente:

- Todos los vértices y aristas son etiquetados con una tupla n-dimensional de naturales.
- Se definen dos mapas $\text{map}_B, \text{map}_D : \mathbb{N}^n \mapsto \mathbb{N}^n$ para representar conexiones, tales que para cada arista $e = (u, v)$, $\text{map}_B(e) = u, \text{map}_D(e) = v$.
- Se agrega un mapa $V_{\text{map}} : \mathbb{N}^n \mapsto \mathbb{N}$ que represente set–vertices, donde $V_{\text{map}}(v) = i$ si $v \in V^i$.
- Un mapa $E_{\text{map}} : \mathbb{N}^n \mapsto \mathbb{N}^2$ que represente set–edges, donde $E_{\text{map}}(e) = (i, j)$ si $e \in E^{i,j}$.

Si los conjuntos y los mapas que describen el SBG pueden ser definidos por intensión, entonces contamos con una representación compacta del grafo original. Para clarificar este punto, a continuación presentaremos la representación compacta del sistema del Listado 1.1. En este caso, consideraremos nuevamente el grafo de la Figura 2b, que para el valor de $N = 10$ se transforma en el grafo de la Figura 3. En dicha gráfica se pueden observar las etiquetas asignadas a cada vértice y arista.

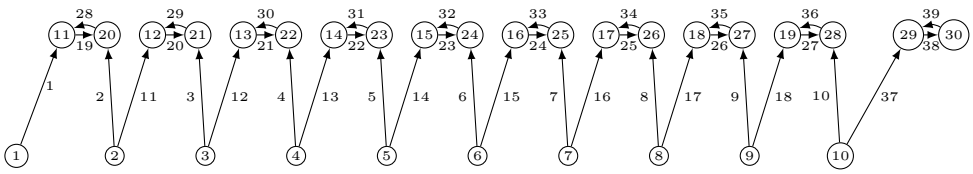


Fig. 3: Grafo dirigido correspondiente al sistema del Listado 1.1.

En este caso, los mapas map_B y map_D son los siguientes:

$$\text{map}_B(e) = \begin{cases} e & \text{if } e \in [1:10] \\ e-9 & \text{if } e \in [11:18] \\ e-8 & \text{if } e \in [19:36] \\ e-27 & \text{if } e \in [37:37] \\ e-9 & \text{if } e \in [38:39] \end{cases}; \ \text{map}_D(e) = \begin{cases} e+10 & \text{if } e \in [1:1] \\ e+18 & \text{if } e \in [2:10] \\ e+1 & \text{if } e \in [11:27] \\ e-17 & \text{if } e \in [28:36] \\ e-8 & \text{if } e \in [37:38] \\ e-10 & \text{if } e \in [39:39] \end{cases} \tag{1}$$

Esta representación de los mapas contará con la misma cantidad de tramos para cualquier valor de N. Por ejemplo, para $N = 100$, la definición será idéntica, tan solo teniendo que ajustar adecuadamente los valores de las constantes presentes en las expresiones, y las cotas de los intervalos.

Resta dar la definición de los mapas que describen los set–vertices y las set–edges. En este caso, agrupando los vértices de acuerdo a los arreglos de variables y ecuaciones a los que representan, definimos:

$$V_{\text{map}}(v) = \begin{cases} 1 & \text{if } v \in [1:1] \\ 2 & \text{if } v \in [2:10] \\ 3 & \text{if } v \in [11:19] \\ 4 & \text{if } v \in [20:28] \\ 5 & \text{if } e \in [29:29] \\ 6 & \text{if } e \in [30:30] \end{cases}; \; E_{\text{map}}(e) = \begin{cases} (1,3) & \text{if } e \in [1:1] \\ (2,4) & \text{if } e \in [2:10] \\ (2,3) & \text{if } e \in [11:18] \\ (3,4) & \text{if } e \in [19:27] \\ (4,3) & \text{if } e \in [28:36] \\ (2,5) & \text{if } e \in [37:37] \\ (5,6) & \text{if } e \in [38:38] \\ (6,5) & \text{if } e \in [39:39] \end{cases}$$

3.4 Trabajos Relacionados

El problema de causalizar sistemas de EDAs a gran escala definidos en base a estructuras repetitivas comenzó a tratarse hace una década. Los primeros intentos no tomaron en cuenta la presencia de lazos algebraicos, como se observa en Bergero et al., 2015; Schuchart et al., 2015. Luego, surgieron dos nuevas propuestas para compilar modelos a gran escala sin necesidad de expandir los arreglos involucrados: por un lado Set–Based Graphs introducidos en Zimmermann et al., 2019, y por otro el compilador de alto rendimiento MARCO presentado en Agosta et al., 2023. Sin embargo, los algoritmos allí propuestos asumen condiciones demasiado restrictivas, y pueden fallar al trabajar con sistemas como el definido en el Listado 1.1, que posee expresiones que involucran distintos índices de un mismo arreglo. Un trabajo más reciente es el de Abdelhak et al., 2023, que intenta preservar los arreglos el mayor tiempo posible, pero continúa operando de manera escalar en presencia de fallas.

En conclusión, según nuestro conocimiento, no existe una solución general previa para detectar lazos algebraicos preservando la representación compacta de arreglos y ecuaciones `for loop`.

3.5 Notación

De aquí en más utilizaremos la siguiente notación:

- Dado un mapa map_x y un conjunto A, al escribir $\text{map}_x[A]$ y $\text{map}_x^{-1}[A]$ nos referimos a la imagen y la pre-imagen de map_x a través de A, respectivamente.
- La expresión $\text{map}_x|A$ denota un mapa con la misma ley que map_x, pero cuyo dominio ha sido restringido a A.
- Dado un grafo dirigido G, su reverso se denotará G^R.
- Escribiremos $\text{mrv}(v, G)$ para referirnos al mínimo vértice alcanzable de v en el grafo dirigido G.

4 Resultados Principales

Esta sección describe el algoritmo propuesto, y analiza ciertas características del mismo.

4.1 Algoritmo Propuesto

El algoritmo propuesto calcula simultáneamente el mínimo vértice alcanzable para todo el grafo, aprovechando la representación compacta SBG. En cada paso, busca eliminar aristas que conecten distintas CFC, hasta que cada una de ellas quede aisladas. Dado un grafo $G = (V, E)$, el procedimiento se puede resumir en los siguientes pasos:

1. Se calcula $\text{mrv}(v, G)$ para todo $v \in V$.
2. Se eliminan todas las aristas $(u, v) \in E(G)$ que verifican $\text{mrv}(u, G) \neq \text{mrv}(v, G)$.
3. Se invierte el sentido de las aristas, $(G \leftarrow G^R)$.
4. Si en el paso anterior se eliminó alguna arista, comenzar desde el punto 3.
5. Fin. Cada CFC puede ser identificada fácilmente ya que todos los vértices que la componen comparten el mismo MRV.

La correctitud del algoritmo propuesto se comprueba a través de los siguientes resultados (omitiremos las pruebas de los mismos, ya que se encuentran fuera del alcance de este trabajo):

Proposición 1. *Dado un grafo dirigido G, si quitamos una arista $(u, v) \in E(G)$, tal que u y v pertenecen a distintas CFC, el grafo resultante posee las mismas CFC que G.*

Proposición 2. *Si dos vértices $u, v \in V(G)$ pertenecen a la misma CFC de un grafo dirigido G, entonces* $\text{mrv}(u, G) = \text{mrv}(v, G)$ *y* $\text{mrv}(u, G^R) = \text{mrv}(v, G^R)$.

Lema 1. *Las CFC de un grafo dirigido $G = (V, E)$ están aisladas unas de otras si y solo si* $\text{mrv}(v, G) = \text{mrv}(v, G^R)$ *para todo $v \in V$.*

Tenemos el siguiente corolario como consecuencia del lema anterior:

Corolario 1. *En un grafo dirigido donde el conjunto de aristas que conectan distintas CFC es vacío, cada componente queda unívocamente identificada por el MRV de sus vértices.*

La Proposición 2 explica que los pares de vértices que satisfacen $\text{mrv}(u, G) \neq \text{mrv}(v, G)$ o bien $\text{mrv}(u, G^R) \neq \text{mrv}(v, G^R)$ pertenecen a distintas CFCs, y luego la Proposición 1 nos asegura que al eliminar las aristas cuyos extremos satisfacen estas condiciones, las CFCs del grafo resultante no cambian con respecto a las de G. Por otro lado, el Lema 1 concluye que si no quedan aristas de este tipo en el grafo, entonces sus CFC se encuentran aisladas. Finalmente, el Corolario 1 verifica que el último paso del algoritmo propuesto retorna una caracterización correcta de las CFC del grafo.

En base al procedimiento ya explicado, y los resultados obtenidos, los Algoritmos 1 y 2 detectan las componentes fuertemente conexas de un SBG dirigido representado por un conjunto de vértices V, un conjunto de aristas E, y dos mapas $\text{map}_B(\cdot)$ y $\text{map}_D(\cdot)$ que describen los extremos de las aristas. El Algoritmo 1 elimina las aristas que tienen distinto MRV en su destino y origen en alguna dirección, como mencionamos en el paso 2.

Algoritmo 1 Paso de Set-Based CFC

function STEP($V, E, \text{map}_B, \text{map}_D$)
  $R_{\text{map}} \leftarrow$ MINREACH($V, \text{map}_B, \text{map}_D$) ▷ Calcula $\text{mrv}(v, G)$ para todo $v \in V$.
  $R^B_{\text{map}} \leftarrow R_{\text{map}} \circ \text{map}_B$ ▷ Mapa de aristas (u, v) a $\text{mrv}(u, G)$.
  $R^D_{\text{map}} \leftarrow R_{\text{map}} \circ \text{map}_D$ ▷ Mapa de aristas (u, v) a $\text{mrv}(v, G)$.
  $E_{\text{same}} \leftarrow (R^B_{\text{map}} - R^D_{\text{map}})^{-1}[\{0\}]$ ▷ Conjunto de aristas (u, v) que verifican $\text{mrv}(u, G) = \text{mrv}(v, G)$.
  $E_{\text{diff}} \leftarrow E \setminus E_{\text{same}}$ ▷ Aristas que deben ser eliminadas.
  $\text{map}_B \leftarrow \text{map}_B|_{E_{\text{same}}}$ ▷ Elimina aristas de E_{diff} en map_B.
  $\text{map}_D \leftarrow \text{map}_D|_{E_{\text{same}}}$ ▷ Elimina aristas de E_{diff} en map_D.
  return $(\text{map}_B, \text{map}_D, E_{\text{same}}, E_{\text{diff}}, R_{\text{map}})$
end function

Luego, el Algoritmo 2 itera invocando a STEP, hasta que el conjunto de aristas que conectan distintas CFCs sea vacío. Una vez que esto sucede, devuelve el MRV para cada vértice, identificando así cada CFC del SBG.

Algoritmo 2 Set-Based CFC

function SCC($V, E, \text{map}_B, \text{map}_D$)
  $(\text{map}_B, \text{map}_D, E, E_{\text{diff}}, R_{\text{map}}) \leftarrow$ STEP($V, E, \text{map}_B, \text{map}_D$) ▷ Elimina aristas con distinto MRV.
  do
    $\text{map}_B \leftrightarrow \text{map}_D$ ▷ Revierte el grafo $(G \leftrightarrow G^R)$.
    $(\text{map}_B, \text{map}_D, E, E_{\text{diff}}, R_{\text{map}}) \leftarrow$ STEP($V, E, \text{map}_B, \text{map}_D$) ▷ Elimina aristas con distinto MRV.
  while $E_{\text{diff}} \neq \emptyset \wedge E \neq \emptyset$
  return R_{map} ▷ Devuelve los representantes de cada CFC.
end function

Todas las operaciones presentes en los Algoritmos 1 y 2 involucran conjuntos. El costo computacional de las mismas es independiente del tamaño de cada conjunto. Sin embargo, esto no provee ninguna cota para el número de iteraciones requeridas para completar la ejecución del Algoritmo 2. Analizaremos en mayor detalle esta cuestión en la Sección 4.2.

4.2 Análisis de Costo

El siguiente lema establece una cota superior para la cantidad de iteraciones que el Algoritmo 2 debe realizar para identificar todas las CFC de un grafo dirigido G.

Lema 2. *Dado un grafo dirigido G, el Algoritmo 2 termina luego de ejecutar a lo sumo $2 \cdot M - 1$ llamadas a la función* STEP*, donde M es el número de CFC de G.*

Este resultado no afirma que la cantidad de iteraciones requeridas es independiente del tamaño de los intervalos que definen al SBG de G, ya que la cantidad de CFCs de G puede depender de dicho tamaño, como sucede en el grafo de la Figura 3. Es por esto que a continuación presentamos condiciones de regularidad en la estructura de G suficientes para garantizar costo computacional constante.

Lema 3. *Sean $G_1, \ldots, G_N$ un conjunto de CFCs sin aristas salientes al resto de los vérties de un grafo dirigido G; y $m_i = \min\{v \in V(G_i)\}$. Supongamos que las aristas $(v_i, v_j) \in E(G)$ tal que $v_i \in G_i$ y $v_j \in G_j$ con $i \neq j$, existen solamente si $m_i < m_j$. Entonces, cada una de las $G_1, \ldots, G_N$ quedan aisladas en la siguiente llamada a* STEP *en el Algoritmo 2.*

De manera similar, encontramos un resultado análogo para el caso en que el sentido de las aristas se encuentra invertido:

Lema 4. *Sean $G_1, \ldots, G_N$ un conjunto de CFCs sin aristas entrantes del resto de los vérties de un grafo dirigido G; y $m_i = \min\{v \in V(G_i)\}$. Supongamos que las aristas $(v_i, v_j) \in E(G)$ tal que $v_i \in G_i$ y $v_j \in G_j$ con $i \neq j$, existen solamente si $m_i > m_j$. Entonces, cada una de las $G_1, \ldots, G_N$ quedan aisladas después de dos llamadas a* STEP *en el Algoritmo 2.*

Tanto el Lema 3 como el Lema 4 explican que un *conjunto grande* de CFCs quedan aisladas entre ellas a lo sumo dos pasos, si las conexiones dentro de cada CFC siguen un orden estricto. Considerando que cualquier estructura regular definida de manera compacta seguirá un orden estricto, estos resultados constatan que la cantidad de iteraciones necesarias del Algoritmo 2 son independientes del tamaño de los intervalos involucrados.

5 Ejemplos y Resultados

Para demostrar las ventajas y desventajas de la metodología propuesta, analizamos los resultados obtenidos al causalizar tres modelos con distintos tipos de lazos algebraicos. En todos los casos, los experimentos se realizaron en una notebook que tiene un procesdor Intel Core i5-1135G7 y 8GB de RAM, y el sistema operativo Ubuntu 20.04.

Para cada modelo, construimos el SBG bipartito correspondiente, y luego calculamos un matching máximo con el algoritmo de Marzorati et al., 2024. A partir de este resultado, se construyó el SBG dirigido sobre el cual aplicamos el Algoritmo 2 para detectar lazos algebraicos. Para cada instancia, se modificó el tamaño de los arreglos dado por el parámetro N, verificando que el proceso completo se ejecuta en tiempo constante con respecto a dicho valor.

Además, se realizaron los mismos experimentos, pero usando el algoritmo de Tarjan de la C++ Boost Graph Library, para contar con una comparación de los resultados con herramientas estándar de grafos.

En todo los casos se midió el tiempo de ejecución de 10 corridas del algoritmo propuesto para identificar CFC, para luego informar el promedio entre todas ellas.

5.1 Un modelo de un circuito con N lazos algebraicos

El primer ejemplo es el presentado en la Sección 2, descripto en el Listado 1.1. El mismo contiene N lazos algebraicos, donde cada uno se compone del par de variables {`Ua[i], Uc[i]`} que deben ser calculadas en simultáneo.

La segunda y tercer columna de la Tabla 1 reportan el tiempo de ejecución del algoritmo SBG y el de Tarjan para esta instancia, variando el valor de N. Como era de esperar, el tiempo de SBG se mantiene constante, mientras que la versión de la BGL exhibe una dependencia lineal con respecto a N.

Tabla 1: Tiempo de CPU (en milisegundos) para distintos modelos y algoritmos de CFC, según el tamaño del arreglo `N`.

N	Circuit		Test RL3	
	SBG	BGL	SBG	BGL
100	1.32	0.01	6.25	0.01
1,000	1.33	0.09	6.36	0.11
10,000	1.35	0.91	6.37	1.36
100,000	1.34	9.74	6.37	14.21
1,000,000	1.34	96.27	6.36	140.01

A continuación describimos los resultados de cada iteración del Algoritmo 2, para explicar en más detalle cómo funciona el mismo. Tomaremos $N = 10$, donde el SBG correspondiente puede observarse en la Figura 3:

- La primer llamada de STEP calcula:

$$\mathrm{mrv}(v, G) = \begin{cases} v & \text{si } v \in [1:19] \\ v-9 & \text{si } v \in [20:28] \\ 29 & \text{si } v \in [29:30] \end{cases} \tag{2}$$

- Se eliminan las aristas $[1:1]$, $[2:10]$, $[11:18]$, y $[37:37]$, ya que sus extremos tienen distinto MRVs.
- Al invocar STEP por segunda vez, el algoritmo invierte el sentido de las aristas para computar $\mathrm{mrv}(v, G^r)$, obteniendo nuevamente el resultado descripto en la Ecuación 2. Así, no es necesario eliminar aristas.
- Como no quedan aristas por eliminar, el algoritmo devuelve el siguiente resultado:

$$R_{\mathrm{map}}(v) = \begin{cases} v & \text{si } v \in [1:19] \\ v-9 & \text{si } v \in [20:28] \\ 29 & \text{si } v \in [29:30] \end{cases}$$

Se observan entonces 20 CFCs, de las cuales 10 están conformadas por un único vértice (vértices del 1 al 10). Las 10 CFCs restantes están compuestas por dos vértices: $\{11, 20\}, \{12, 21\}, \ldots, \{19, 28\}$ y $\{29, 30\}$.

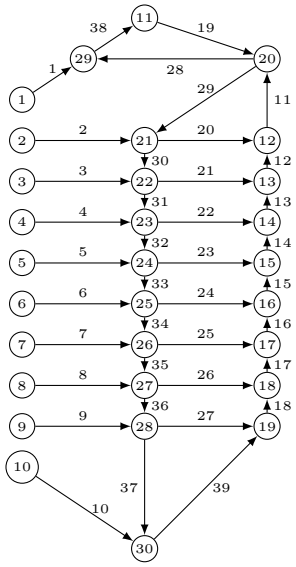
Estos resultados demuestran que los valores de `der(iL[i])` pueden ser calculados de forma individual, y por otro lado, el despeje de `Ua[i]` es necesario hacerlo junto al despeje de `Uc[i]`.

5.2 Un modelo de un circuito con un lazo algebraico a gran escala

El tercer ejemplo es el modelo del Listado 1.4, que describe otro circuito eléctrico de resistencias e inductores. Este modelo tiene un lazo algebraico a gran escala, que incluye los arreglos `iR` y `uL` completos.

```
model TestRL3
  constant Integer N=100;
  Real iL[N], iR[N], uL[N];
  parameter Real L=1, R=1, L1=1, U=1, R0=1;
equation
  for i in 1:N loop
    L*der(iL[i]) = uL[i]; // der(iL[1:N])
  end for;
  for i in 1:N-1 loop
    iR[i] - iR[i+1] - iL[i] = 0; // iR[1], iR
        [3:N]
    uL[i] - uL[i+1] - R*iR[i+1] = 0; // iR[2],
        uL[2:N-1]
  end for;
  U - uL[1] - R*iR[1] = 0; // uL[1]
  uL[N] - (iR[N] - iL[N])*R0 = 0; // uL[N]
end TestRL3;
```

Listado (1.4) Modelo TestRL3 con N=100



(a) Modelo TestRL3 (b) Grafo Dirigido para TestRL3

Fig. 4: Gráficas de TestRL3.

Las dos últimas columnas de la Tabla 1 informan el tiempo de ejecución del algoritmo SBG y el de Tarjan para esta instancia, variando el valor de N. Se observa el mismo comportamiento de los dos ejemplos anteriores.

Mostramos el grafo correspondiente al modelo en la Figura 4b, para el cual el Algoritmo 2 ejecuta los siguientes pasos:

– La primer llamada a STEP retorna:

$$\mathrm{mrv}(v, G) = \begin{cases} 11 & \text{si } v \in [12:30] \\ v & \text{si } v \in [1:11] \end{cases} \tag{3}$$

– Luego se eliminan las aristas $[1:10]$.
– En la segunda llamada a STEP (luego de revertir el sentido de las aristas) se obtiene el mismo mapa de la Ecuación (3).
– Como no hay aristas eliminadas en el último paso, el resultado final es:

$$R_{\mathrm{map}}(v) = \begin{cases} 11 & \text{si } v \in [12:30] \\ v & \text{si } v \in [1:11] \end{cases} \tag{4}$$

El R_{map} resultante constata que hay una CFC grande que incluye a los vértices $[11:30]$ representada por 11. Además, hay 10 CFCs conformadas por un único vértice (compuestas por los vértices de 1 a 10 cada una).

Aunque en este caso fueron necesarias solo dos invocaciones de STEP, el cálculo del MRV resultó más costoso que en los ejemplos anteriores, ya que existe un camino largo hacia el MRV de 28 que es 11 (notar que debe recorrer 4 set-vertices). Aún teniendo en cuenta esta situación, el tiempo total de ejecución es de alrededor de 6ms, un resultado más que aceptable.

6 Conclusiones

Este trabajo definió un algoritmo novedoso para hallar Componentes Fuertemente Conexas en grafos con estructuras repetitivas. El mecanismo que hizo posible tales resultados es la representación compacta propuesta por los Set-Based Graphs. De este modo, el algoritmo exhibe costo computacional constante con respecto al tamaño de los set-vertices y set-edges que definen el SBG.

Este algoritmo es útil para hallar lazos algebraicos durante la causalización de grandes sistemas de EDAs. En ese caso, además de las cotas obtenidas para el tiempo de ejecución, el procedimiento preserva la estructura compacta del modelo para que las etapas posteriores también puedan sacar provecho de esta funcionalidad.

Actualmente, seguimos trabajando para completar el proceso de causalización de modelos orientados a objetos con la representación SBG. Por el momento, se han resuelto las siguientes etapas: aplanado (ver Marzorati et al., 2022), matching máximo (ver Marzorati et al., 2024) y aquí presentamos una solución para detectar CFCs.

El próximo paso consiste en proponer un algoritmo de orden topológico para SBGs, que ordene verticalmente el sistema deseado. Además, para sacar provecho de los resultados aquí obtenidos, es necesario definir un proceso de rasgado (ver Elmqvist and Otter, 1994) que opere con SBGs para generar código de simulación eficiente. Finalmente, buscaremos postular un mecanismo de reducción de índice, tomando como punto de partida el algoritmo de Pantelides, 1988.

Agradecimientos

Se agradecen las contribuciones del proyecto PICT–2021-I-A-00826 (ANPCYT) y PIP–2022/2024 11220210100093CO (CONICET).

References

Abdelhak, K., Casella, F., & Bachmann, B. (2023). Pseudo array causalization. *Modelica Conferences*, 177–188.

Agosta, G., Casella, F., Cattaneo, D., Cherubin, S., Leva, A., Scuttari, M., & Terraneo, F. (2023). Marco: An experimental high-performance compiler for large-scale modelica models. *Modelica Conferences*, 13–22.

Bergero, F., Botta, M., & Kofman, E. (2015). Efficient compilation of large scale modelica models. *11th International Modelica Conference*.

Brück, D., Elmqvist, H., Mattsson, S. E., & Olsson, H. (2002). Dymola for multi-engineering modeling and simulation. *Proceedings of Modelica 2002*.

Cellier, F. E., & Kofman, E. (2006). *Continuous system simulation*. Springer Science & Business Media.

Elmqvist, H., & Otter, M. (1994). Methods for tearing systems of equations in object-oriented modeling. *Proceedings ESM*, *94*, 1–3.

Fritzson, P., & Bunus, P. (2002). Modelica-a general object-oriented language for continuous and discrete-event system modeling and simulation. *Proceedings 35th Annual Simulation Symposium. SS 2002*, 365–380.

Fritzson, P., & Engelson, V. (1998). Modelica—a unified object-oriented language for system modeling and simulation. *European Conference on Object-Oriented Programming*, 67–90.

Fritzson, P., Pop, A., Abdelhak, K., Asghar, A., Bachmann, B., Braun, W., Bouskela, D., Braun, R., Buffoni, L., Casella, F., et al. (2020). The open-modelica integrated environment for modeling, simulation, and model-based development. *Modeling, Identification and Control*, *41*(4), 241–295.

Gabow, H. N. (2000). Path-based depth-rst search for strong and biconnected components. *Information Processing Letters*.

Marzorati, D., Fernández, J., & Kofman, E. (2022). Efficient connection processing in equation–based object–oriented models. *Applied Mathematics and Computation*, *418*, 126842.

Marzorati, D., Fernández, J., & Kofman, E. (2024). Efficient matching in large dae models. *ACM Transactions on Mathematical Software*.

Pantelides, C. C. (1988). The consistent initialization of differential-algebraic systems. *SIAM Journal on Scientific and Statistical Computing*, *9*(2), 213–231.

Rozhdestvensky, K., Ryzhov, V., Fedorova, T., Safronov, K., Tryaskin, N., Sulaiman, S. A., Ovinis, M., & Hassan, S. (2020). *Computer modeling and simulation of dynamic systems using wolfram systemmodeler*. Springer.

Schuchart, J., Waurich, V., Flehmig, M., Walther, M., Nagel, W. E., & Gubsch, I. (2015). Exploiting repeated structures and vectorization in modelica. *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015*, (118), 265–272.

Sharir, M. (1981). A strong-connectivity algorithm and its applications in data flow analysis. *Computers & Mathematics with Applications*, *7*(1), 67–72.

Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM journal on computing*, *1*(2), 146–160.

Zimmermann, P., Fernández, J., & Kofman, E. (2019). Set-based graph methods for fast equation sorting in large dae systems. *Proceedings of the 9th International Workshop on Equation-based Object-oriented Modeling Languages and Tools*, 45–54.