Detección Eficiente de Lazos Algebraicos en Grandes Sistemas de EDAs

Denise Marzorati[♭,†], Joaquin Fernandez[♭] y Ernesto Kofman[♭,†]

[♭]*CIFASIS-CONICET, Bv. 27 de Febrero 210 bis, Rosario, Santa Fe, Argentina, marzorati@cifasis-conicet.gov.ar*
[†]*FCEIA, Universidad Nacional de Rosario, Av. Pellegrini 250, Rosario, Santa Fe, Argentina, dmarzorati@fceia.unr.edu.ar*

Resumen: 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 en grandes sistemas de Ecuaciones Diferenciales Algebraicas (EDAs) que cuentan con arreglos de variables y ecuaciones.

Palabras clave: *modelos a gran escala, causalización de EDAs, set–based graphs, componentes fuertemente conexas, modelica*
2000 AMS Subject Classification: 68W99

1. Introducción

Cierta clases de problemas requieren analizar el grado de dependencia entre componentes de ciertos sistemas. Usualmente, pueden ser transformados en problemas de Teoría de Grafos donde se deben hallar las CFC de algún grafo que represente las relaciones existentes en el sistema [2].

Uno de estos casos es el de la detección de lazos algebraicos (variables que deben ser resultas simultáneamente) para convertir un sistema de EDAs en uno equivalente compuesto únicamente por Ecuaciones Diferenciales Ordinarias (EDOs). Para ello se representa cada ecuación con un vértice, y se añade una arista dirigida de acuerdo al orden en que deben ser despedejadas las variables representadas por ambos nodos. Así, hallar las CFC de dicho grafo es equivalente a detectar los lazos algebraicos del sistema de EDAs.

Se han propuesto numerosos algoritmos para resolver este problema [1, 6, 7]. Sin embargo, todos ellos incurren en costos computacionales que dependen de la cantidad de vértices y aristas del grafo. Aunque esto resulta aceptable en muchos casos, el proceso de causalización puede aplicarse a sistemas de ecuaciones con millones de elementos, excediendo con facilidad el alcance de las técnicas actuales.

Es importante tener en cuenta de que, en la mayoría de los casos, estos sistemas a gran escala son el resultado de definiciones repetitivas. Esta característica puede ser aprovechada por el enfoque SBG [8], que agrupa vértices y aristas en conjuntos definidos por intensión, proveyendo además operaciones cuyo costo no depende del tamaño de dichos conjuntos.

La metodología SBG se diseñó con el objetivo de compilar lenguaje Modelica eficientemente, proceso que requiere convertir un modelo orientado a objetos a código de simulación ejecutable. El procedimiento está compuesto por varias etapas que pueden resolverse creando grafos que reflejen la estructura del modelo, y aplicando algoritmos conocidos a los mismos. Sin embargo, Modelica cuenta con el comando `for loop` que permite definir arreglos de variables y ecuaciones, lo que conlleva a operar en grafos con cientos o miles de vértices y aristas. Para abordar esta problemática SBG propone agrupar vértices y aristas generados por el mismo arreglo de variables o ecuaciones para operar sobre ellos de manera simultánea (y no individualmente).

Aprovechando esta representación compacta se han propuesto varios algoritmos: uno de ellos detecta componentes conexas con el objetivo de aplanar modelos orientados a objetos [4]; otro encuentra matchings máximos para ordenar horizontalmente sistemas de EDAs [5]. Asimismo, SBG se utilizó con éxito en la generación de código compacto que calcula la matriz Jacobiana esparcida de sistemas [3].

En concordancia con estas propuestas, el presente trabajo define un algoritmo SBG para hallar CFC. Bajo ciertas condiciones, es capaz de encontrar todos los conjuntos de CFC pertenecientes a un SBG con costo computacional constante respecto al tamaño de los Vértices-Conjunto y Aristas-Conjunto.

2. SET-BASED GRAPHS

Los resultados obtenidos en el presente trabajo se deben a la utilización del enfoque de *Set-Based Graphs* (SBG) [4, 8]. A continuación, presentamos las principales definiciones:

Definición 1 (Set–Based Graph) *Dado un grafo $G = (V, E)$, un SBG asociado al mismo se compone de:*

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

De manera similar pueden definirse SBGs dirigidos y bipartitos.

2.1. REPRESENTACIÓN COMPACTA

Una representación conveniente para SBGs es la siguiente:

- Todos los vértices y aristas son etiquetados con una tupla n-dimensional de naturales.
- Un mapa que represente vértices–conjunto, $V_{\text{map}} : \mathbb{N}^n \mapsto \mathbb{N}, V_{\text{map}}(v) = i$ si y solo si $v \in V^i$.
- Dos mapas que representan aristas, $\text{map}_1, \text{map}_2 : \mathbb{N}^n \mapsto \mathbb{N}^n$ tales que para cada arista $e = \{u, v\}$, $\text{map}_1(e) = u, \text{map}_2(e) = v$.
- Un mapa que represente aristas–conjunto, $E_{\text{map}} : \mathbb{N}^n \mapsto \mathbb{N}^2, E_{\text{map}}(e) = (i, j)$ si y solo si $e \in E^{i,j}$.

Definiendo de manera conveniente conjuntos y mapas apropiados, esta representación permite describir de manera compacta estructuras repetitivas presentes en grandes grafos.

3. ALGORITMO DE CFC

La idea principal del algoritmo es aislar cada una de las CFC de un grafo dirigido G, eliminando aristas que conectan vértices de distintas componentes. Para ello, en cada iteración se calcula el *mínimo vértice alcanzable*, notado como $\text{mrv}(v, G)$. Dos vértices adyacentes con distinto mrv pertenecen a dos CFC distintas, por lo que la arista que los conecta puede ser eliminada. Luego se invierte el sentido de las aristas para obtener el grafo reverso G^R, y aplicar el mismo procedimiento, hasta que no queden aristas candidatas a ser eliminadas.

Más rigurosamente, pueden demostrarse los siguientes resultados:

Proposición 1 *En un grafo dirigido G, borrar las aristas que conectan vértices de distintas CFC resulta en un nuevo grafo con las mismas CFC presentes en G.*

Proposición 2 *Dados dos vértices u y v pertenecientes a la misma CFC del grafo dirigido G, necesariamente $\text{mrv}(u, G) = \text{mrv}(v, G)$ y $\text{mrv}(u, G^R) = \text{mrv}(v, G^R)$.*

Lema 1 *Un grafo dirigido G tiene aisladas todas sus CFC si y solo si $\text{mrv}(v, G) = \text{mrv}(v, G^R)$ para todo $v \in V(G)$.*

Corolario 1 *Dado un grafo dirigido G donde el conjunto de aristas que conectan vértices de distintas CFC es vacío, cada mrv es el representante de su propia CFC.*

Primeramente, el Algoritmo 2 se encarga de llamar reiteradamente a STEP hasta que no queden aristas con extremos en distintas componentes, situación en la cual el Corolario 1 afirma que tendremos identificadas a todas las CFC de G. Por otro lado, el Algoritmo 1 se ocupa de calcular el mrv para todos los vértices del SBG simultáneamente, para luego identificar las aristas (u, v) tales que $\text{mrv}(u) \neq \text{mrv}(v)$ que por la Proposición 2 pertenecen a distintas CFC. Este conjunto de aristas es eliminado del SBG, ya que sin ellas contará con las mismas CFC, tal como lo asevera la Proposición 1.

Algorithm 1 Set-Based CFC step

function STEP($V, E, \mathrm{map}_B, \mathrm{map}_D$)
  $R_{\mathrm{map}} \leftarrow$ MINREACH($V, \mathrm{map}_B, \mathrm{map}_D$) ▷ Cálculo de $\mathrm{mrv}(v, G)$.
  $R^B_{\mathrm{map}} \leftarrow R_{\mathrm{map}} \circ \mathrm{map}_B$ ▷ Mapa de una arista (u, v) al $\mathrm{mrv}(u, G)$.
  $R^D_{\mathrm{map}} \leftarrow R_{\mathrm{map}} \circ \mathrm{map}_D$ ▷ Mapa de una arista (u, v) al $\mathrm{mrv}(v, G)$.
  $E_{\mathrm{same}} \leftarrow (R^B_{\mathrm{map}} - R^D_{\mathrm{map}})^{-1}[\{0\}]$ ▷ Aristas (u, v) que verifican $\mathrm{mrv}(u, G) = \mathrm{mrv}(v, G)$.
  $E_{\mathrm{diff}} \leftarrow E \setminus E_{\mathrm{same}}$ ▷ Aristas (u, v) que verifican $\mathrm{mrv}(u, G) \neq \mathrm{mrv}(v, G)$.
  $\mathrm{map}_B \leftarrow \mathrm{map}_B|_{E_{\mathrm{same}}}$ ▷ Quitar aristas de E_{diff}.
  $\mathrm{map}_D \leftarrow \mathrm{map}_D|_{E_{\mathrm{same}}}$ ▷ Quitar aristas de E_{diff}.
  return $(\mathrm{map}_B, \mathrm{map}_D, E_{\mathrm{same}}, E_{\mathrm{diff}}, R_{\mathrm{map}})$
end function

Algorithm 2 Set-Based CFC

function SCC($V, E, \mathrm{map}_B, \mathrm{map}_D$)
  $(\mathrm{map}_B, \mathrm{map}_D, E, E_{\mathrm{diff}}, R_{\mathrm{map}}) \leftarrow$ STEP($V, E, \mathrm{map}_B, \mathrm{map}_D$) ▷ Quitar aristas que conectan distintas CFC.
  do
    $\mathrm{map}_B \leftrightarrow \mathrm{map}_D$ ▷ Invertir el sentido de las aristas ($G \leftrightarrow G^R$).
    $(\mathrm{map}_B, \mathrm{map}_D, E, E_{\mathrm{diff}}, R_{\mathrm{map}}) \leftarrow$ STEP($V, E, \mathrm{map}_B, \mathrm{map}_D$) ▷ Quitar aristas que conectan distintas CFC.
  while $E_{\mathrm{diff}} \neq \emptyset \wedge E \neq \emptyset$
  return R_{map} ▷ Retornar mrv (representates de cada CFC).
end function

4. EJEMPLO

En esta sección presentamos un ejemplo que utiliza el algoritmo propuesto implementado en C++ como parte de la librería SBG [1]. Para evaluar la eficiencia del mismo se comparó el tiempo de ejecución con la del algoritmo de Tarjan presente en BGL (Boost Graph Library) escrita en C++.

La detección de lazos algebraicos en el modelo descripto por el programa Modelica de la Figura 1a es equivalente a hallar todas las CFC del grafo de la Figura 1b. En la Figura 1a se han indicado con comentarios las variables que serán despejadas de cada ecuación, evidenciando un gran lazo algebraico que contiene a todas las variables a excepción de `der(iL[1:N])`.

En la primer ejecución de STEP el algoritmo obtiene los siguientes resultados: $mrv(v, G) = 11$ si $v \in [12 : 30]$ y en caso contrario $mrv(v, G) = v$. Por este motivo borra las aristas $e \in [1 : 10]$. En su segunda llamada (dentro del bloque `do-while`) se obtienen los mismos representantes para todo $v \in V$. De este modo el conjunto E_{diff} es vacío, finalizando así la ejecución del algoritmo, identificando dos CFC: aquella que contiene a $v \in [1 : 10]$ y otra compuesta por $v \in [11 : 30]$.

Luego, variamos el tamaño del modelo con $N = 100$ hasta $N = 1000000$ registrando el tiempo de ejecución de ambas librerías (SBG y BGL) en la Tabla 1. Estos resultados verifican la propiedad de costo constante del algoritmo, aunque también evidencian que para tamaños chicos de modelo el algoritmo de Tarjan es más veloz. Sin embargo, nuestro enfoque tiene la ventaja de preservar una representación compacta tanto para la entrada como para la solución.

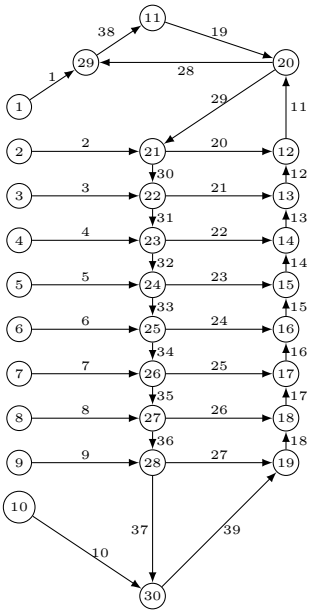
5. CONCLUSIONES

Presentamos un algoritmo de detección de componentes fuertemente conexas en grafos dirigidos. Dicho algoritmo es particularmente eficiente para identificar de lazos algebraicos en grandes sistemas de EDAs,

[1] Para ejecutar los ejemplos referirse a: `https://github.com/CIFASIS/sb-graph/releases/tag/v3.0.0`

```
model TestRL3
  constant Integer N=10;
  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;
```

(a) Modelo de un circuito eléctrico



(b) Grafo correspondiente al modelo

Tabla 1: Tiempo de ejecución para distintos valores de N

	Implementación SBG	**C++ BGL**
Tamaño	**Tiempo SCC [mseg]**	**Tiempo SCC [mseg]**
100	6,90	0,01
1000	6,97	0,11
10000	7,01	1,36
100000	7,04	14,21
1000000	7,04	140,01

ya que el costo computacional asociado al mismo no depende del tamaño de los arreglos implicados. Adicionalmente, el resultado se encuentra definido de manera intensiva, lo cual puede ser aprovechado por las etapas posteriores del proceso de causalización.

REFERENCIAS

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

[2] D. F. HSU, X. LAN, G. MILLER, AND D. BAIRD, *A comparative study of algorithm for computing strongly connected components*, in 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), IEEE, 2017, pp. 431–437.

[3] E. KOFMAN, J. FERNÁNDEZ, AND D. MARZORATI, *Compact sparse symbolic jacobian computation in large systems of odes*, Applied Mathematics and Computation, 403 (2021), p. 126181.

[4] D. MARZORATI, J. FERNÁNDEZ, AND E. KOFMAN, *Efficient connection processing in equation–based object–oriented models*, Applied Mathematics and Computation, 418 (2022), p. 126842.

[5] D. MARZORATI, J. FERNÁNDEZ, AND E. KOFMAN, *Efficient matching in large dae models*, ACM Transactions on Mathematical Software, (2024).

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

[7] R. TARJAN, *Depth-first search and linear graph algorithms*, SIAM journal on computing, 1 (1972), pp. 146–160.

[8] P. ZIMMERMANN, J. FERNÁNDEZ, AND E. KOFMAN, *Set-based graph methods for fast equation sorting in large dae systems*, in Proceedings of the 9th International Workshop on Equation-based Object-oriented Modeling Languages and Tools, 2019, pp. 45–54.