Graph theoretic approach for automating electric circuit

Leanne Dong

Gina Cody School of Engineering and Computer Science
Concordia University Montréal

June 3, 2020

Overview

Graph vocabulary

- Graph $G = (V, E)$ is a finite set of points (V: vertices or nodes) and lines joining some of these points (E edges, arcs, links); Graphs can be directed and undirected;
- Cycle/Circuit/Loop: a path such that the final node of the last link coincides with the initial node of the first link;
- Tree: a graph that is connected but has no cycles;
- Spanning tree T: For a connected graph G, this is just a subgraph that contains all nodes of G;
- Branches: The edges of a spanning tree T;
- Incidence matrix $A_{inc} = \{a_{ij}\}$ is

$$a_{ij} = \begin{cases} 1 & \text{if } j\text{th edge is incident on the } i\text{th node} \\ 0 & \text{if } j\text{th edge is not incident incident on the } i\text{th node} \end{cases}$$

Graph vocabulary

- Complete graph: this is a graph where each node is joined by an arc to every other node.
- Partial graph: a graph that contains all the nodes of the original graph but only some edges.
- subgraph: a graph that contains only a subset of the nodes of the original and all of the edges that join them.
- connected graph: a graph is connected if there is at least one chain from any nodes to every other node.

Why graph theory for electric circuits?

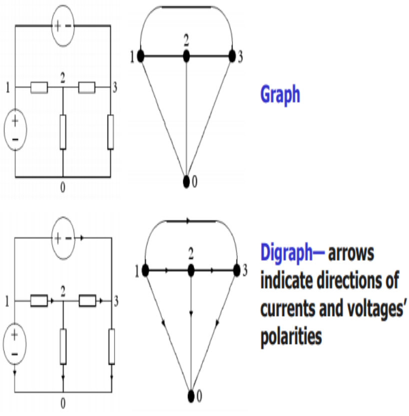
- Powerful, scalable methods to model complex networks arising many applications
- A Simple method which allows us to derive the differential equations for even very complicated circuits

Graph for electric circuit: The idea

- Algebraic graph theory for electronic network simulation first due to Poincaré and Veblen
- Gotlieb and Corneil (1967)'s idea of generating spanning trees by finding fundamental set of cycles (cycle base)
- The above was further improved by Paton (1969). The DFS search was used to construct cycle base: Computational very efficient!
- Further improved by Tiernan (1970). Efficient but complicated. Fortunately this has been implemented in the Boost C++ libraries. See in particular the graph library . Boost::graph provide a comprehensive range of headers including Tiernan's efficient search algorithm in finding the fundamental cycles of the graph.

Type of graphs in electrical theory

- Intuitively, circuits are graphs that describe the interconnection of electrical elements
  1. passive elements such as resistances, capacitances, inductances
  2. active elements
  3. sources (or excitation)
- Two variables: voltage (V) and currents (I).



Loop

- A loop is a set of branches of graph forming a closed path.
- Example: branches $\{a, c, d\}$, $\{b, d, e\}$ and $\{a, b, e, c\}$.

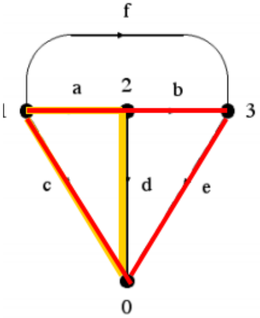


Figure: Braches and loops

Tree and co-tree

- A tree is a set of branches of a graph which contains no loop.
- Thus, a tree is a maximal set of branches can be connected without forming a loop.
- After a tree is chosen, the remaining branches form a co-tree

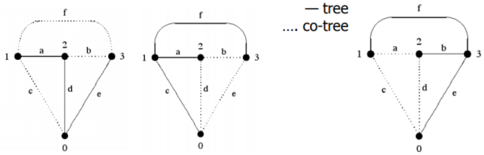


Figure: Tree and co-tree

A very simple example

Table: PSPICE netlist.

I1	0	1	1Amp
R2	1	0	1Ohm
R3	1	2	1Ohm
R4	2	0	1Ohm

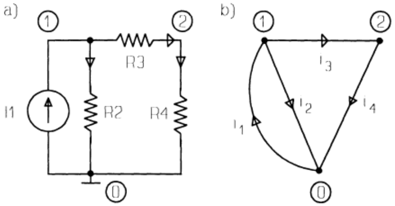


Figure: A simple circuit and its topology

Kirchhoff's law again

KVL: At each loop or mesh,

$$\sum_{loop} V_k = 0$$

KCL: At each node,

$$\sum I_{inflow} = \sum I_{outflow}$$

A very simple example: KCL

The KCL equations w.r.t. node 1, 2 and 0 are

$$\begin{aligned} -I_1 + I_2 + I_3 &= 0 \\ -I_3 + I_4 + \quad &= 0 \\ I_1 - I_2 - I_4 &= 0 \end{aligned}$$

Linear dependency! $\implies$, Reference node at which the KCL equations may be omitted.

A very simple example: KCL

Suppose we take node 0 as the reference node, so eliminate the 3rd equation, hence

$$\begin{pmatrix} -1 & 1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \end{pmatrix} = 0$$

or simply

$$AI = 0$$

where I is the vector of branch currents and A is known as the incidence matrix containing only coefficients 0 or 1, -1.

A very simple example: KVL

- Certain assembly of branches form loops (i.e. closed paths)
- In our last example, the loops are $\{I_1, R_2\}$, $\{I_1, R_3, R_4\}$ and $\{R_2, R_3, R_4\}$
- Conservation of energy: work done around a loop should be 0. Hence, the sum of voltages of branches constituting the loop must be 0. In the similar vein, the KVL equations can be written as a compact matrix form as

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 1 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = 0$$

or simply

$$BV = 0$$

A very simple example: summary

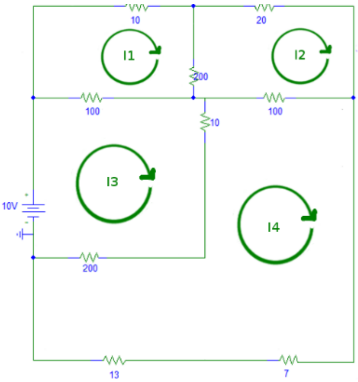
By the superposition principle, the topological matrices A and B both contain the same full information on the assembly of branches.

The formal procedure

1. Describe the circuit in a computer-readable format;
2. Find a set of independent loops;
3. For each loop, we associate a loop current;
4. Use KVL to set up a system of n equations, one for each independent loop;
5. Solve the system of equations and thus find the values of the loop currents (I);
6. Use the superposition principle to find the total current in each circuit branch.

A concrete example

- We show a set of independent loops with associated loop currents
- Resistance are conventionally assumed to be ohms.
- Voltage values are in volt.



A concrete example: setting up the Kirchhoff equations

At each branch b, the total voltage drop $V^{(b)}$ across the branch is the sum of the voltage drop due to the loop currents $I_{l'}$ that traverse that branch:

$$V^{(b)} = \sum_{b \in l} \sum_{l' \in L_b} R^{(b)} I_{l'}$$

So basically the inflow outflow relations,

$$\begin{aligned}
(10+200+100)I_1 - 200I_2 - 100I_3 + 0I_4 &= 0 \\
-200I_1 + (20+100+200)I_2 + 0I_3 - 100I_4 &= 0 \\
-100I_1 + 0I_2 + (100+10+200)I_3 - (200+10)I_4 &= 10 \\
0I_1 - 100I_2 - (10+200)I_3 + (100+7+13+200+10)I_4 &= 0
\end{aligned}$$

Methods

- Gotlieb and Corneil (1967): Implementated by a Physicist Prof. Milotti's group, see CircuitMath.
- The above was further improved by Paton (1969). The DFS search was used to construct cycle base: Computational very efficient!
- Further improved by Tiernan (1970).

In term of storage and speed, Paton and Tiernan are better. Moreover, Tiernan's algorithm has been written into the modern C++ numerics libraries like Boost

Method: Gotlieb and Corneil (1967)

Assumptions

- Undirected graph is represented by its adjacency matrix $A = \{a_{ij}\}$ is

$$a_{ij} = \begin{cases} 1 & \text{if } j\text{th node is connected with the } i\text{th node} \\ 0 & \text{if } j\text{th node is not connected with the } i\text{th node} \end{cases}$$

- Graph is connected
- Each connected component can be separated

Method: Gotlieb and Corneil (1967)

Key steps

- Construct a set of disjoint trees by choosing some edges.
    1. The set of n node is partitioned w.r.t. some chsen edges to form a set of disjoijnt trees with adjacency matrix B.
    2. For rach row i of A, set $b_{ji} = b_{ij} = 1$ if a_{ij} is the first superdiagonal element of the ith row of A to equal 1; otherwise set $b_{ji} = b_{ij} = 0$.
- Divide the set of nodes of the graph into connected components w r.t. the edges chosen in step 1.
- Amalgamate the componets by adding appropriate edges to yield a spanning tree.
- Produce a set of fundamental cycles

Method: Paton (1969)

This has been implemented by a computer scientist Philipp Sch. He demonstrated how in principle one can enumerate all cycles of a graph. However the method does not scale up well.

Method: Tiernan (1970)

A theoretically very eddifcient search algorithm that uses an exhaustive search to find all elementary circuits of a directed graph. For more detail, one shall consult with

- The official Boost Graph Library site : boost::Graph
- Textbook: The Boost Graph library
- Uni-Resource: Brown
- Tutorial: technical-recipes

Circuit equations

Having found the independent loops, we can set up the system of equations like the one we show before and solve them. To fix idea, we consider mainly simple linear circuits. The solutions can be trivially attained using the LU decomposition method via some scientific software. One choice is the routines **ludcmp** and **lubksb** in the Numerical Recipes C library

Branch currents and voltages

- The solution of linear equation tields all the independent loop currents;
- From them, we can find the branch currents by summing over all the independent loop currents that traverse a given branch.
- By looping over all branches, one can construct a new matrix $I = \{i_{ik}\}$ where the element i_{jk} stores the current in the branch that links the i-th node with the k-th node.
- Similarly, we calculate a voltage matrix $V = \{v_{jk}\}$, where the matrix elemet $v_{jk} = r_{jk}, i_{jk}$ stores the voltage drop in the branch that links the jth node with the kth node.
- Voltage are set by the power supplies w.r..t. a ground terminal: one can find the voltage of any node w.r.t. ground by following any path along the circuit that joins the node with the ground node and summing all the voltage drops.
- Finnaly, save the results on a file and save for further analysis

Potential improvements

- A comparison of graph-based method with the nodal method used in established programs such as SPICE
- Optimising circuit representation in the description file
- Deal with sparse matrix case, the program efficiency could be boosted with the use of sparse matrix code
- code could be cleaned up with C++, with introduction of linked lists represent loops and other graph structures
- Can be extended to handgle time-dependent voltage and currents, instead of the linear system we have seen, we can manipulate the corresponding DEs and use a DE solver
- Could be extended to include non-linear elements like semiconductor diades, transistors, etc, this would require the intro of a nonlinear equation solver

Potential improvements

- Can be implemented in Object-Oriented sense, utilise the efficient C++ library like Boost.
- Could work on a graphical interface both for input - by placing components in a special window - and for output - by automatically plotting output functions
- Could be used in the analysis of fluid flow - using the analogy between hydraulic and circuit networks.

Future plan

Develop existing approach in modern C++. Make use of algorithm library and Boost.

```
#ifndef BOOST_GRAPH_CYCLE_HPP
#define BOOST_GRAPH_CYCLE_HPP

#include <vector>

#include <boost/config.hpp>
#include <boost/graph/graph_concepts.hpp>
#include <boost/graph/graph_traits.hpp>
#include <boost/graph/properties.hpp>

#include <boost/concept/detail/concept_def.hpp>
namespace boost {
    namespace concepts {
        BOOST_concept(CycleVisitor,(Visitor)(Path)(Graph))
        {
            BOOST_CONCEPT_USAGE(CycleVisitor)
            {
                vis.cycle(p, g);
            }
        private:
            Visitor vis;
            Graph g;
            Path p;
        };
    } /* namespace concepts */
using concepts::CycleVisitorConcept;
} /* namespace boost */
#include <boost/concept/detail/concept_undef.hpp>
```