Applying Colored-Graph Isomorphism for Electrical Circuit Matching in Circuit Repository

Ananda Maiti[1] and Balakrushna Tripathy[2]

[1,2] School of Computing Science and Engineering, VIT University
Vellore, TN 632 014, India

Abstract

In electrical engineering educational industrial applications it is often required for the students to build and analyze well-known circuits according to their syllabus. In this paper a colored graph isomorphism based model is discussed that is being developed to match two electrical circuits. The procedure involves two steps, first being, to store the circuit in the data base in form of a colored graph and then matching users input circuit with it using graph isomorphism. Since graph isomorphism is in NP, it has no known polynomial time solution. However, in this particular problem of circuits, the colors and weight used in the nodes vary and the graphs generated are sparsely connected. Thus the algorithm runs in a reasonable time.

Keywords: *circuit matching, graph isomorphism, colored graph, circuits, circuit repository.*

1. Introduction

The circuit matching problem has great use in CAD designs and automations, supporting users in verifying and validating new circuit design by checking it components. It is also useful in educational purposes where a students' knowledge about a circuit may be tested by matching their circuits with an existing one to determine how well they can construct circuits. The paper describes only the development of the matching algorithm and the repository building mechanisms.
A circuit repository a database or storage location to store a large number of circuits as a set of connected components. If a user wants to find and match their circuits to the existing ones, or find sub-circuits, within an existing one, their circuits are matched with the circuits in repository.

The problem addressed in this paper is matching two circuits and determining whether the two are exactly equal or not. The solution requires an algorithm that can compare two circuits in an efficient and deterministic manner.

Two well-known algorithms have been proposed in the past that are used to solve the circuit matching problems. They are:

a. Luellau's algorithm [1]

It describes a computer program called 'BLEX'. It takes a circuit described at the transistor level and performs 'block extraction' process to find larger blocks such as flip-flops and gates.

The circuits are represented as a graph with the components vertices of two types: spiders and Nodes. Nodes correspond to the actual components and spiders correspond to the connection points between components. The spider nodes are labeled by product of weight of all directed edges into the vertices. The weights are predetermined with some prime numbers only.

One of the major disadvantages of Luellau's algorithm is its use of products of prime factors to label the vertices. The maximum value that can be stored in the integer format for in a computer is limited. If the product value increased then, an overflow occurs. Thus to avoid this condition the number of devices are limited. Another disadvantage is finding the starting vertices. A circuit may not have a vertex with unique set of edges.

b. Ohlrich's Algorithm [2]

The Ohlrich's algorithm makes some improvement on the previous algorithm by decreasing the time to detect the starting vertices. The running time is reasonable. This algorithm re-labels each vertex during run-time. Unlike the previous one, here there is no limit on the number of devices and the circuits matched can be large. However, the algorithm itself is very complicated and hard to implement. Secondly, the Ohlrich's Algorithm is described as a prototype [3]. Hence it is not guaranteed to solve the problem with any kind of circuit.

Both the algorithm has the problem that it cannot cover all kind of circuits using a single mechanism.

2. The Colored Graph Approach

A colored graph is a graph in which the nodes are assigned a color according to a color function. In this section the, it is discussed, how the problem of circuit matching is reduced to colored graph isomorphism.

2.1 Conversion to graph

The electrical circuits contain a set of different components that are connected to each other by wires or conducting material. To represent the circuit a common format such as SPICE is available. However, the SPICE format is not suitable for describing a graph or for matching with each other. In this model, the circuit is represented as a colored weighted undirected graph:

a) The nodes are each components terminals or parts. For e.g. a diode has two terminal hence it is represented by two nodes D+ and D- which are connected, W is used to denote a connection point.
b) The color of a node is determined according to which part of which electrical component it represents. For e.g. resistors are always colored 7 (red in figure 2 and 7 in color vector in table 1)
c) The weight of the node represents the value of the components itself. For e.g. a resistor is weighted by its resistance value. This is however optional is the components is generic such as ground.
d) The nodes are connected in the graph if they are connected in the circuit

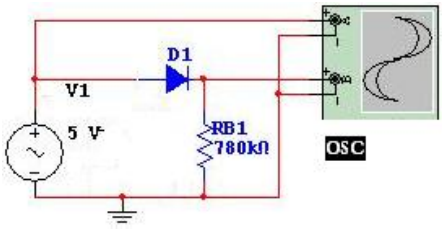

Fig. 1. Half-wave rectifier Circuit Diagram

The Half-wave rectifier circuit diagram is converted to the graph as an example. The circuit contains 5 components, ground (G), Oscilloscope (O), signal generator (V), diode (D) and a resistor (R).

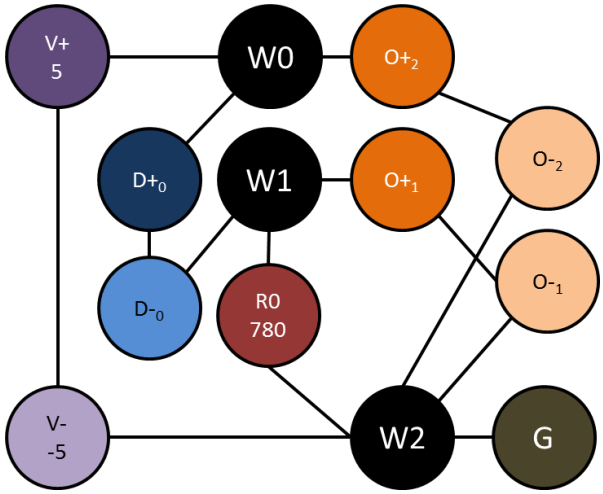

Fig. 2. The Graph
(The labels are component names and the weights)

Each terminal of the components becomes a node or vertex in the graph. For e.g. the diode becomes D+ and D- and Oscilloscope with two channels is represented by O+1, O+2 and O-1 and O-2. A component such as the resistor or ground whose orientation does not matter is represented by a single node. The Anodes of the channels of the oscilloscope has the same color since they are same components and perform the same function. An edge in the graph represents the connections between each component in the circuit.

The colored graph is stored or processed as a 2 dimensional array of adjacency matrix and a 1 dimensional array of colors.

Table 1: The Adjacency Matrix

Label	*G*	*V+*	*V-*	*O+1*	*O-1*	*O+2*	*O-2*	*D+0*	*D-0*	*R0*	*W0*	*W1*	*W2*
Color	*0*	*1*	*2*	*3*	*4*	*3*	*4*	*5*	*6*	*7*	*12*	*12*	*12*
Node	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	0	1	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	1	0	0	0	0	0	0	1	0
4	0	0	0	1	0	0	0	0	0	0	0	0	1
5	0	0	0	0	0	0	1	0	0	0	1	0	0
6	0	0	0	0	0	1	0	0	0	0	0	0	1
7	0	0	0	0	0	0	0	0	1	0	1	0	0
8	0	0	0	0	0	0	0	1	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	0	0	1	1
10	0	1	0	0	0	1	0	1	0	0	0	0	0
11	0	0	0	1	0	0	0	0	1	1	0	0	0
12	1	0	1	0	1	0	1	0	0	1	0	0	0

It is obvious that any kind of circuit can be represented using this model. The model generates a unique graph for every unique electrical circuit. The level of components can be altered from simple elements such as diodes, resistors, transistors etc. to more complex components such as flip-slops and digital gates. For this model and algorithm to work both the circuit repository and the users interface must have the same color function i.e. color the modes using the same color for the same components.

2.1 Graph Isomorphism

After the graph is formed it is stored in a circuit repository. If the users want to check their circuits, then they draw a circuit which is converted to a graph and matched with an existing graph in the repository. Both the graphs must follow same coloring scheme for same components, for example color for a resistor must be same in both graphs. The graph isomorphism algorithm must ensure the following:

a. There exist a one to one relation between vertex set V and edge set E in both graphs.
b. If there exist such a relation then for every relation between nodes x and y in the graphs, color of x must be equal to color of y.

The graph comparison algorithm works in two phases:

a. Checking if the graphs have same |V| and |E| and the number of vertices of each color is same in both the graphs.
b. If the graphs pass the first phase, then the vertices of the color with least multiplicity i.e. the color with lowest number of vertices are selected as starting vertices. Then recursively, it is checked whether, any two of the vertices have same neighborhood of vertices with the same colors and weight. If a pair is found then the each vertices in the neighborhoods is checked similarly.

3. Comparing the Circuits

3.1 Algorithm

The algorithm to compare the two circuits is now equivalent to solve the colored graph isomorphism. The algorithm is as follows:

Inputs: Graph A and B in form of adjacency matrix, Color Vector consisting of all colors in the same order as in the graphs C_A and C_B.

Output: Yes if the Graph are same and No if otherwise.

Step 1: Check if the number edges |E| and |V| in A and B are same. If not then return false, otherwise proceed.

Step 2: Sort the Color vectors C_A and C_B along with the vertices set V_A and V_B. Check if $C_A = C_B$. If not then return false, otherwise proceed.

Step 3: Find the Color with minimum numbers of vertices from vectors C_A and C_B along with the vertices of that color in set V_A and V_B and store them in $TempC_A$ and $TempC_B$ respectively

Step 4: For every combination of a pair of vertices x, y in $TempC_A$ and $TempC_B$, set a temporary array new_visited where only new_visited[y] = true and call procedure isEquiv(x, y).

Procedure isEquiv (vertex x, vertex y, vertex [] $Visited_A$, vertex [] $Visited_B$)

1. If Color(x) ≠ Color(y) or Weight(x) ≠ Weight(y) then return false.
2. If x and y have same neighborhood in colors and vertices then proceed otherwise return false.
3. Initialize an empty array T to store vertices.
4. For every combination of vertices e in neighborhood N(x) in A and f in N(y) in B such that color(e) = color(f), weight(e) = weight(f), e ∉ $Visited_A$, f ∉ $Visited_B$ and e, f ∉ T, set temporary arrays $new\_visited_A$ = $Visited_A$ where $new\_visited_A[x]$ = true, $new\_visited_B$ = $Visited_B$ where $new\_visited_B[y]$ = true and call isEquiv (e, f, $new\_visited_A$, $new\_visited_B$).
5. If e and f are equivalent then store e and f in T.
6. If each vertices in N(x) is stored alongside each vertices in N(y) in T, other than the vertices visited, then return true otherwise return false.

Step 5: If for any x, y the procedure returns true, the Graphs are isomorphic otherwise not.

3.2 Analysis of the Algorithm

The algorithm is deterministic as it performs in the same way every time it is run a particular graph.

Some advantages of the colored graph approach:

a. There is no requirement to handle each and every block or components in the circuit separately. A universal set of colors can be maintained to represent the components. Hence different components for analogue and digital circuits can be represented as the colored graph in the same manner.
b. There is no limit to the number of components of any type, since the components are marked from 0...n i.e. in whole numbers.
c. The complexity of the components does not have to be same. This model can accurately match circuits

containing basic components such as diodes, resistor etc. to higher level components such as IC, Logic Gates and Flip flop.

d. The weight of the graph, which represents the values of the components such as resistance and capacitance, can be compared, thus generating an accurate match of circuits.

There is a big disadvantage in terms of complexity of the algorithm. The time complexity of the proposed algorithm in the worst-case scenario is exponential. Graph isomorphism is in NP [1,2], hence there is no known algorithm that can solve the comparison in polynomial time. However, as Luellau states, most practical circuits when transformed to a graph are sparsely connected. Moreover since both the color and weight i.e. the component type and its values are matched, the number of recursions is largely reduced. Hence this algorithm works in acceptable time to solve the circuit matching problem. For e.g. with the above example graph with 10 nodes and 15 edges, the number of colors were randomly altered from 2 to 7(see figure 3). Consequently, the color multiplicity decreased and thus the running time also decreased. Of course, if the number of colors is reduced to 1 then the programs takes an enormous amount of time. However in a practical circuit it not possible to get a graph with all node of same color since the circuits use different components and there terminals are differently colored. Hence it is expected to have a graph with large number of colors and smaller number of edges from areal laboratory circuit.

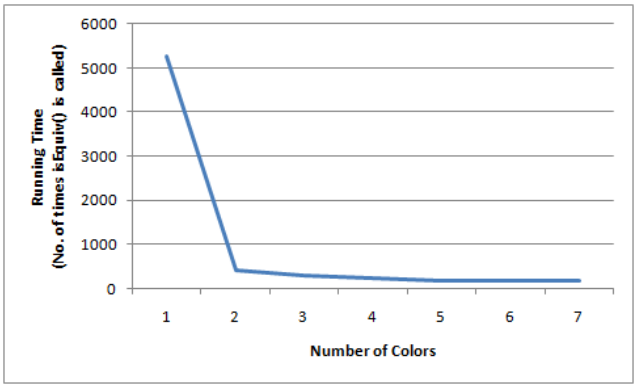

Figure 3. The performance of the algorithm

In case of the half wave rectifier, there are 10 nodes and 8 colors. There are 6 colors with exactly one node each. Let say that the figure 2 is graph A and a similar graph B as in figure 4. The second phase of the algorithm can start with any of them. Let us assume it starts by comparing G in both graphs. First, the degree (G) = degree (G'). Next the neighborhood N (G) in A = {V-, R, O- , O-} which is equal to neighborhood N(G') in B = {V-', R', O-' , O-'}. Hence we again check if V-, V-'and R, R' are equivalent in the same way. We don't Compare G and G' again since the nodes are marked visited. Note that there are two O-nodes in the neighborhood. Hence all combinations are check for them.

This scheme also allows for quick building of the circuit repository and databases. The developer needs to draw the circuit which is converted and saved as the graph in the database.

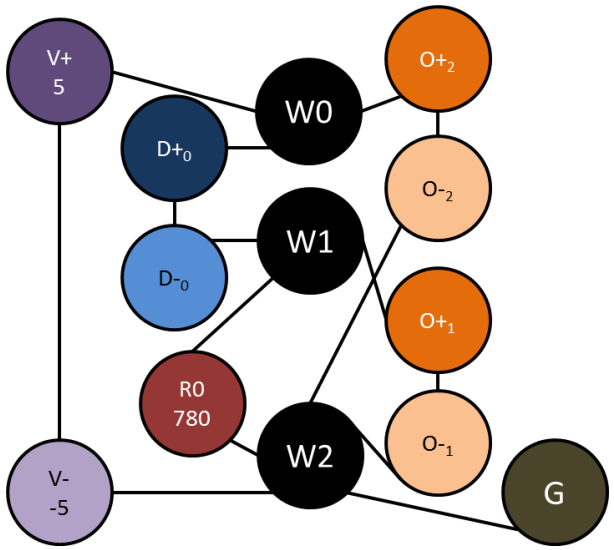

Figure 4. Graph B

4. Future Work

One scope of improvement in the scheme is lowering he worst case running time. The running time for a random polynomial time coin flipping algorithm for color preserving graph isomorphism has been proved to be $O(V^4 \log V)$ by Babai[4]. Color preserving isomorphism with restricted color multiplicity has been reported to in the Nick's Class (NC) [5]. It means the isomorphism is solved in polynomial time if it is run on a parallel computer. Hence if a proper algorithm is deployed then the circuit matching problem can be solved in polynomial time.

Another scope of improvement is modifying the algorithm to find smaller circuits within a larger circuit. To do this, the vertices of the larger graph are removed with color not belonging to the smaller graph. After that in most cases the graph becomes a disconnected set of smaller graphs. These can be easily matched individually. Even if it

remains connected, the isEquiv() procedure runs only for the number of vertices in the smaller graph.

5. Conclusions

Reducing the problem of matching electrical circuits in a colored graph as an abstract graph isomorphism problem helps formulating a general domain independent solution. A colored graph approach is discussed for producing graphs for the electrical circuits. The circuit matching problems can be used to implement important applications such as student's laboratory where a student is asked to construct a circuit digitally and it is compared to a stored graph of that circuit in the database for equality to evaluate the students' performance. Other applications include sub-circuit matching i.e. finding a smaller circuit in a large circuit.

References

[1] F. Luellau, T. Hoepken, and E. Barke. "A technology independent block extraction algorithm." In 21st Proceedings of the Design Automation Conference on Design automation, pages 610–615, IEEE Press, 1984.

[2] M. Ohlrich, C. Ebeling, E. Ginting, and L. Sather. "Subgemini: identifying subcircuits using a fast subgraph isomorphism algorithm" In Proceedings of the 30th international on Design automation conference, pages 31–37, ACM Press, 1993.

[3] J. Whitham, "A Graph Matching Search Algorithm for an Electronic Circuit Repository", Univ. of York, 2004.

[4] Babai, L., "Monte Carlo algorithms in graph isomorphism testing", Tech. Rep. 79-10, Dep. Math. et Stat., Univ. de Montreal, 1979.

[5] Luks, Eugene M., "Parallel algorithms for permutation groups and graph isomorphism," Foundations of Computer Science, 1986., 27th Annual Symposium on , vol., no., pp.292-302, 27-29 Oct. 1986.

Ananda Maiti is with the School of Computing Science and Engineering, VIT University, Vellore India.

B. K. Tripathy is a senior professor with the School of Computing Science and Engineering, VIT University, Vellore India.