# Finding Spanning Trees and Hamiltonian Circuits in an Un-Oriented Graph an Algebraic Approach

Cristian E. Onete
NXP Semiconductors
Eindhoven, The Netherlands

cristian.onete@ieee.org

A Maria Cristina C. Onete
CASED & TU Darmstadt,
Darmstadt, Germany

cristina.onete@cased.de

***Abstract*— In this paper, symbolic matrices and a simple algebraic method to list spanning trees and find Hamiltonian circuits in a simple un-oriented graph are used. A concrete, fully-parallel algorithm that achieves both goals, with examples is shown. A necessary and sufficient condition for Hamiltonicity is presented, too.**

***Keywords-Necessary and sufficient condition for Hamiltonicity, Graph theory, Spanning tree, Hamiltonian circuit.***

## I. INTRODUCTION

Finding all the spanning trees in a graph has many applications, amongst which electrical circuit analysis and inverse-free determinant calculation. The related problem of finding Hamiltonian circuits has applications in, for instance, finding ground paths in electrical circuits. A method to find spanning trees uses Wang Algebra (WA), which allows for symbolic determinant computation [1,2,5]. Wang Algebra is used for electronic circuits in finding the symbolic determinant of the short-circuit admittance matrix $\mathbf{Y_{sc}}$ [2]. For circuits with only bi-terminal devices, $\mathbf{Y_{sc}}$ is (N-1)x(N-1), with main diagonal elements equaling the sum of the admittances connected to the nodes, and elements (i, j) equaling the negative of the admittance between nodes i and j. For a complete 4-node circuit (Fig. 2), $\mathbf{Y_{sc}}$ is as (1).

$$Y_{sc} = \begin{bmatrix} (y_{12}+y_{13}+y_{14}) & -y_{12} & -y_{13} \\ -y_{12} & (y_{12}+y_{23}+y_{24}) & -y_{23} \\ -y_{13} & -y_{23} & (y_{13}+y_{23}+y_{34}) \end{bmatrix} \quad (1)$$

Previous work [1,2,5,6] shows that the determinant of $\mathbf{Y_{sc}}$ is the logical WA product of the main diagonal entries of $\mathbf{Y_{sc}}$. For our example on 4 nodes, the determinant is as in (2).

$$\text{Det}(\mathbf{Y_{sc}})=(y_{12}+y_{13}+y_{14}) \text{ x } (y_{12}+y_{23}+y_{24}) \text{ x } (y_{13}+y_{23}+y_{34}) \quad (2)$$

This WA approach is cumbersome, as the terms are hard to compute, many canceling out. The remaining terms are exactly all the spanning trees in the original graph.

Note that each entry $y_{ij}$ is the admittance associated with the link (i, j) in the graph. The modified short-circuit admittance matrix $\mathbf{MY_{sc}}$ in equation (3) only considers the links included in the determinant, and not the values of the admittances associated with them.

$$MY_{sc} = \begin{bmatrix} [(1,2)+(1,3)+(1,4)] & -(1,2) & -(1,3) \\ -(1,2) & [(2,1)+(2,3)+(2,4)] & -(2,3) \\ -(3,1) & -(3,2) & [(3,1)+(3,2)+(3,4)] \end{bmatrix} (3)$$

The WA determinant of $\mathbf{MY_{sc}}$ is then described by (4).

$$\text{Det}(MY_{sc})=[(1,2)+(1,3)+(1,4)]\text{x}[(2,1)+(2,3)+(2,4)]\text{x}[(3,1)+(3,2)+(3,4)] \quad (4)$$

This notation is further used in order to find spanning trees and Hamiltonian circuits in simple, undirected graphs.

## II. DESCRIPTION OF A HAMILTONIAN GRAPH

The graph in Fig. 1 is simple, undirected, with N nodes and N links. It represents a Hamiltonian circuit, i.e. a circuit visiting each node in the graph exactly once. For such circuits we use the notation in (5).

$$H = \{(1,2)(2,3)(3,4)...(N,1)\} \quad (5)$$

In the following it is assumed that consecutive links are concatenated. Removing a link from the Hamiltonian circuit yields a spanning tree. Spanning trees that can be extended to Hamiltonian circuits are called *Hamiltonian* trees and are denoted as in (6). Here $H_T$ is Hamiltonian for Fig. 1, as adding link (N, 1) yields a Hamiltonian circuit.

$$H_T=(1,2)(2,3)(3,4)...(N-1,N) \quad (6)$$

Let us note that a necessary and sufficient condition for Hamiltonicity is shown already in [7]; jumping ahead, this condition is also based on Hamiltonian trees and circuits, but it does not rely on the structure of the source and destination sets of candidate sub-graphs as it is done here. Instead, the approach of [7] concerns sub-matrices in the modified incidence matrix of graphs (**MIM**).

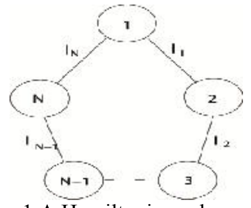

Fig. 1 A Hamiltonian sub graph

Equations (5) and (6) assume an arbitrary orientation of the links, with a source and a destination node, i.e. a node the link leaves from, and a node it arrives at. Denote the set of source nodes by **S** and the set of destination nodes by **D**. Since the original graph is undirected, however, (i,j) is essentially equivalent to (j,i). Let H be a Hamiltonian circuit; the following properties hold:

1. There exists a link orientation in H such that each of the nodes appears only once in both **S** and **D**. In particular, **S**=**D**={1,2, …, N}. Note that {(1,2)(2,3), … , (N-1,N)(1,N)} is a valid Hamiltonian circuit although node 1 appears twice in **S** and node N appears twice in **D**. This is because we can reorient link (1,N) so that each node appears exactly once in **S** and in **D** respectively.
2. For all subsets P of the links of H and all link orientations, the subsets $\mathbf{S_P}$ and $\mathbf{D_P}$ are not equal.

In this paper, we use the observation that eliminating a single link from a Hamiltonian circuit yields a spanning tree. We use the converse property, trying to find spanning trees which yield Hamiltonian circuits. Property 1 eliminates candidate trees only if there is **no** link orientation such that **S** and **D** contain each node exactly once. In practice, we remove candidate trees if they present circuits even in one **particular** orientation; this follows because the orientation is arbitrary to begin with, and so each re-orientation yielding a candidate tree will automatically give a circuit for another candidate tree. Some speed-ups require link-rearranging before candidate elimination are shown, too.

Our approach here relies on finding Hamiltonian trees, which are a particular kind of spanning trees. A method for enumerating all the spanning trees in a graph is already shown in e.g. [6]. In this paper a different approach is used, however the methods are somewhat equivalent. Whereas [6] uses the properties of a modified incidence matrix, we start from the fact that all the spanning trees in a graph are terms of the symbolic determinant of the short-circuit admittance matrix.

However, Wang Algebra is not used. In particular, the disadvantage of using WA for finding the determinant is that it is rather inefficient: many of the computed terms cancel out. In the present approach, these terms are eliminated very early.

## III. Spanning trees and Hamiltonicity

A graph is called Hamiltonian if it has a Hamiltonian circuit. Consider an undirected, simple (i.e. there are no loops and no parallel links) graph G on N nodes and with L links. The following theorem holds:

**Theorem 1** Graph G(N, L) is Hamiltonian **if and only if** there exists a subset of links $L_H$ in L such that the sub-graph H(N, $L_H$) has properties 1 and 2 in the previous section i.e.: 1. There exists a link orientation in H such that each of the nodes appears only once in both **S** and **D**. In particular, **S**=**D**={1,2, …, N}. 2. For all subsets P of the links of H and all link orientations, the subsets $\mathbf{S_P}$ and $\mathbf{D_P}$ of source and destination nodes are not equal.

**Proof:** Clearly, such a graph is a Hamiltonian circuit, thus the graph is Hamiltonian if and only if properties 1 and 2 hold. □

Note that [1,7] already state necessary and sufficient conditions for Hamiltonicity in simple undirected graphs. The novelty of our approach w.r.t. [7] has been already discussed above. The solution in [1] relies on incidence matrices and uses Diophantine equations. Our approach is different, as it relies on spanning trees and graph search. It also does not require the active use of WA, i.e. it does not actively compute the terms of the symbolic determinant. By using properties 1 and 2, the terms that cancel out early on are removed, since they represent circuits within the spanning graphs. In particular, the spanning trees from the WA determinant are derived **without** using equation (1), as explained in Theorem 2.

**Theorem 2** Let G(N, L) be a simple, undirected graph and let $\mathbf{MY_{sc}}$ be the modified short-circuit admittance matrix. Consider the concatenation of the main diagonal terms. A term of this concatenation (represented by a list of links as in equation (6)) is a spanning tree in the graph **if and only if** property 2 above holds for every sub-graph of this term.

**Proof:** As stated before, if one writes out all the terms of the determinant, some of them will cancel out: these are the terms which do **not** yield spanning trees. Spanning trees essentially fulfill two properties: they visit each node, and they are trees i.e. they have no internal circuits. We use the second property of Hamiltonian circuits, namely that they must contain no sub-circuits. In particular, the sub-graph described by the entire term must not be a circuit. Therefore as soon as property 2 holds for some sub-graph of the term, or for the term itself, it cannot be a spanning tree anymore. On the other hand, all spanning tree candidates are terms of the determinant [1,2,5,6], hence there cannot exist a spanning tree which does **not** appear in the determinant. □

This theorem also shows how to construct spanning trees i.e. by iteratively concatenating terms and removing those having internal circuits. One can also use Theorems 1 and 2 together to find Hamiltonian circuits in simple, undirected graphs.

## IV. FINDING HAMILTONIAN CIRCUITS

Consider the complete 4 nodes graph in Fig. 2.

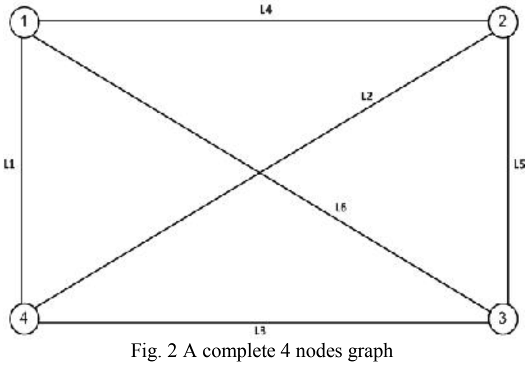

Fig. 2 A complete 4 nodes graph

Its modified short circuit matrix is as in (3), yielding the determinant in (4). We expand the first term of (4) as follows:

$$\begin{aligned}\text{Prod}=&(1,2)[(2,1)+(2,3)+(2,4)][(3,1)+(3,2)+(3,4)]+\\&(1,3)[(2,1)+(2,3)+(2,4)][(3,1)+(3,2)+(3,4)]+\end{aligned} \quad (7)$$

(1,4)[(2,1)+(2,3)+(2,4)][(3,1)+(3,2)+(3,4)].

We note a very attractive property of the Wang expansion of the determinant, namely that each term can be analyzed in parallel, thus reducing the time complexity of the algorithm. Take only the first sub term, namely, (1,2)[(2,1)+(2,3)+(2,4)][(3,1)+(3,2)+(3,4)]. We expand this term:

$$\begin{aligned}\text{Term1} = {} & (1,2)(2,1)[(3,1)+(3,2)+(3,4)] + \\ & (1,2)(2,3)[(3,1)+(3,2)+(3,4)] + \\ & (1,2)(2,4)[(3,1)+(3,2)+(3,4)].\end{aligned} \tag{8}$$

Let us notice that the first term can be wholly eliminated, because it already has a circuit in the first two terms. We continue the expansion and obtain the following sub terms:

$$\begin{array}{ll} \{(1,2)(2,3)(3,1)\} & \{(1,2)(2,3)(3,2)\} \\ \{(1,2)(2,3)(3,4)\} & \{(1,2)(2,4)(3,1)\} \\ \{(1,2)(2,4)(3,2)\} & \{(1,2)(2,4)(3,4)\} \end{array} \tag{9}$$

The first and second terms have circuits according to property 2. The fifth term features node 2 twice in the source and twice in the destination node sets, therefore by property 1 it cannot yield a Hamiltonian circuit. On the other hand, it is a candidate for a spanning tree. The sixth term features, in this link orientation, node 4 twice in the destination node. If we apply this method fully in parallel and without any optimization, we can dismiss this term already, as we are going to find the different orientation in another term. The term remains, however, a candidate for a spanning tree.

If we are looking for spanning trees, we have four candidates now: terms 3,4,5, and 6. If we are looking for Hamiltonian circuits, we have two candidates left: terms 3 and 4. We show in the next section how to optimize this algorithm in order to find both spanning trees and Hamiltonian circuits faster: in this version of the algorithm, sub term 6 will also be included as a candidate.

As the terms are now fully expanded, all the candidates for spanning trees actually do represent spanning trees. We compare the link sets to see if they are identical and find that they are all different. The algorithm continues for Term 2 in equation (8), and then for the last term.

If we are looking for Hamiltonian circuits, we look now at the two candidates. We find Hamiltonian circuits if one of the candidate trees 3 and 4 are Hamiltonian trees. Since the trees are spanning trees, they contain no circuits and thus property 2 holds. They are spanning trees, too, and thus it holds that there exist nodes i and j such that i = {1, … , N}**S** and j = {1, … , N}**D**. The final step in the algorithm is, given the candidate tree, to find nodes i and j (by comparing the sets S and D with the set of all nodes) and to look for link (i,j) or (j,i) in the original graph. If such a link exists, then this completes the Hamiltonian circuit. If no such link exists for any candidate tree, the graph is not Hamiltonian.

For our example, take the first candidate: {(1,2)(2,3)(3,4)}. We have **S**={1,2,3} and **D**={2,3,4}, thus i = 4 and j = 1. Since the link (4,1) exists in the graph in Fig. 2, we obtain the following Hamiltonian circuit: {(1,2)(2,3)(3,4) (4,1)}.

Note that for this first term we have skipped the following expansions, which would have been necessary in WA: (i) we have removed sub term 1 from equation (9); (ii) we have removed two more expansion sub terms from the second expansion, and thus found 4 spanning trees. If we look at the complexity of finding a Hamiltonian circuit, we have found the first Hamiltonian circuit by additionally removing two candidate trees (5 and 6) and focusing on determining whether the remaining candidates (3 and 4) are Hamiltonian.

We illustrate how to run the algorithm and find all the spanning trees by continuing with terms 2 and 3 in (8). We remove the following candidates from Term 2 because they contain circuits: {(1,3)(2,1)(3,1)};{(1,3)(2,1)(3,2)}; {(1,3)(2,3)(3,1)}; {(1,3)(2,3)(3,2)}; {(1,3)(2,4) (3,1)} and keep the following spanning trees:{(1,3)(2,1)(3,4)}; {(1,3)(2,3)(3,4)}; {(1,3)(2,4)(3,2)}; {(1,3) (2,4) (3,4)}.

The same algorithm is repeated for the third term, yielding the trees:{(1,4)(2,1)(3,1)}; {(1,4)(2,1)(3,2)}; {(1,4)(2,1)(3,4)}; {(1,4)(2,3)(3,1)}; {(1,4);(2,3);(3,4)}; {(1,4)(2,4)(3,1)}; {(1,4)(2,4)(3,2)}; {(1,4)(2,4)(3,4)}.

Note that we have found a total of $4^2 = 16$ spanning trees; this is indeed the total number of spanning trees, as a complete simple graph on N nodes has exactly $N^{N-2}$ trees.

Thus, our determinant-based method is suitable for finding both spanning trees and Hamiltonian circuits. A similar idea is also used by [3] for finding the Hamiltonian cycles, whose approach is, however, very complicated owing to the use of Gröbner bases.

We formalize the previous algorithm as follows:

**Algorithm 1 [Spanning trees/Hamiltonian circuits]**
**Input**: matrix $\mathbf{MY_{sc}}$ for G(N, L); **Output**: Spanning tree set T/Hamiltonian circuit $H_c$.
1. Write WA concatenation of the main diagonal entries. Initialize sets T = H = {}.
2. Expand the first factor by the associativity law. Treat each sub term in parallel and initiate lists **S** and **D** with the source and destination nodes of the link from the first factor. In the case of (8) for example, the first term corresponds to initial lists **S** = {1} and **D** = {2}, the second term corresponds to initial lists **S** = {1}, **D** = {3}, whereas the third term corresponds to **S** = {1}, **D** = {4}.

3. For each sub term, expand the next factor of the concatenation, update **S** and **D** accordingly, and check that no new circuit was formed with the addition of the new link (i.e. one checks that property 2 holds in each newly created component of the graph). Eliminate all sub terms which create circuits. If the desired output is the Hamiltonian circuit, then check additionally that property 1 holds for that particular link orientation, i.e. check that no node appears twice in the source/destination node set. Eliminate sub terms for which this property does not hold.

4. Repeat for all the factors. The terms that are left are the spanning trees of the graph; update T to contain them all. If the desired output is a Hamiltonian circuit, update H to contain the remaining Hamiltonian tree candidates. Find i and j such that i = {1, … , N}**S** and j = {1, … , N}**D.** For each

candidate, if there exists a link (i,j) or (j,i) in the tree, output candidate concatenated with (i,j) as Hamiltonian circuit. A concrete way of running step 3 is as follows: given initial link set $L^{i-1}$, concatenate it with the newest link (i,j) from the expansion to obtain $L^i$. Assume $L^i$ contains a circuit. This circuit can't have existed in $L^{i-1}$, (we would have deleted it earlier). Thus link (i,j) closes the circuit. Initialize S={i}, D={j}. Find all links (s,i) and (j,t) in $L^{i-1}$. If there is no link of the form (s,i) or no link of the form (j,t) halt: there is no circuit in $L^i$. For each pair (s,i), (j,t), update S and D accordingly and compare them. Repeat this step, i.e. find links (u,s) and (t,v), but this time if no link satisfies the relation, do not halt, but rather choose a different (s,i), (j,t) pair, until all are exhausted. We repeat the process until either S = D for some subset of links or until all possibilities are exhausted.

As in [6], the inefficiency factor of this method is:

$$R = \frac{n^n}{(n+1)^{(n-1)}} \qquad (10)$$

for n:=N-1. The algorithm has an exponential total runtime, as the number of spanning trees is $N^{(N-2)}$ for a complete graph. However, the algorithm is fully parallel and as in [6], there is no backtracking. Moreover, in a circuit comprising different elements which are bridged by a single connection, it is not necessary to search for spanning trees in the entire circuit! The circuit can be divided into sub-sections, each pair of components connected by a single isthmus, and the method can be applied to each section. A final spanning tree will then contain the trees found in each component and the isthmi between the components. The complexity then is reduced to the complexity of the method for the biggest component.

Furthermore, special properties of graphs can be used, such as symmetry, to our advantage. For complete graphs symmetry can be used to reduce the runtime of the spanning tree algorithm. Given one spanning tree we immediately generate another N-1 trees by permuting the nodes. For example given spanning tree {(1,2)(1,3)....(1, N)}, we re-label the nodes such that node i becomes i+1, obtaining {(2,3)(2,4) .... (2,1)}. Repeating the process yields another spanning tree and so on. This optimization is particularly good for graphs which are too large for the algorithm to be run fully in parallel (on $(N-1)^{N-1}$ processors). In fact, we only need to run the algorithm for the first term, i.e. in our example Term1 in equation (9), and then re-label the nodes. This means that we reduce the parallel complexity by a factor of N-2 for a complete graph!

Why does this work? Complete graphs are fully symmetric, meaning that the first term of the determinant includes all possible combinations of nodes 1 and nodes 2 to N, the second term includes all combinations for node 2, etc. By always performing the permutation for each new tree, we obtain the other terms (1,k), k>2 as permutations and thus it is not necessary to iterate the process.

In our example, given spanning tree {(1,2)(2,3)(3,4)}, we find related trees: {(2,3)(3,4)(4,1)}; {(3,4)(4,1)(1,2)}; {(4,1)(1,2)(2,3)}. Given the second spanning tree {(1,2)(2,4)(3,1)}, we also get: {(2,3)(3,1)(4,2)};{(3,4)(4,2)(1,3)} and {(4,1)(1, 3)(2,4)} and so on, until we have all 16 trees.

The optimization can also be used also for graphs which are "almost" complete i.e. each node is less than N(N-1)/2 connected. We first complete the graph by adding links to it until each pair of nodes is connected (keep track of added links by listing them in a set L). Run algorithm to find spanning trees. For each spanning tree, compare the link set with L. If there exists a link in L which appears in the tree, eliminate the candidate tree from T. Repeat until all the spanning trees have been tested and output the resulting T.

We can also optimize the Hamiltonian circuit algorithm if a fully parallel implementation is not possible. For Hamiltonian circuits, it is relevant to find only the first Hamiltonian circuit. So far we have eliminated all the candidates if for a particular numbering property 1 did not hold. However, if we are running the algorithm sequentially, we gain time as follows: Whenever a new link appears, creating doubles in the S and D lists, reverse the link (i.e. replace (i,j) by (j, i)) and test again. If there are still doubles in S and D, eliminate candidate; otherwise, keep it. In this way, we transform {(1,2)(2,4)(3,4)} to {(1,2)(2,4)(4,3)} and get a Hamiltonian circuit: {(1,2)(2,4)(4,3)(3,1)}.

## IV. Experimental results

We ran the algorithm to find Hamiltonian circuits sequentially on a cubic planar graph with different number of nodes. The runtime is always better than $N^3$ for N<14 nodes. Hamiltonicity can thus be checked sub-exponentially for reasonably small graphs. This is useful for tracing a ground line in electronic circuits or finding a closed route in a town. Furthermore, intermediate candidate lists are kept at a minimum as we eliminate circuits as soon as they are found.

## V. Conclusions

In this paper we show **a necessary and sufficient condition** for a simple undirected graph to be Hamiltonian. Moreover, we describe a **fully-algebraic, fully-parallel** algorithm to list **both** spanning trees and Hamiltonian circuits which can be optimized in order to minimize resource use. The approach is useful for circuit analysis, floor planning, tracks routing, etc.

## References

[1] R. F. Duffin, T. D. Morley, "Wang Algebra and Matroids", *IEEE Trans .on Circuits and Systems*, vol. CAS-25, pp 758 – 762, 1978.

[2] W-K Chen "Graph Theory and Its Engineering Applications", World Scientific Publishing Co. Pte. Ltd, 1997.

[3] V. Ejov, J. A. Filar, S. K. Lucas and J. L. Nelson, "Solving The Hamiltonian Cycle Problem Using Symbolic Determinants", Taiwanese Journal Of Mathematics, Vol. 10, No. 2, pp. 327-338, February 2006.

[4] Balabanian N., Bickart T.A., "Electrical Network Theory", Wiley 1969.

[5] Guohun Zhu, G., Song C., Hirota K., Dong F., Wu Y., "Necessary and Sufficient Conditions for Hamiltonian based on Linear Diophantine Equation Systems with Cycle Vector", 2009 Third International Conference on Genetic and Evolutionary Computing, pp. 847-850.

[6] C. E. Onete, M.C.C. Onete; "Enumerating all the spanning trees in an un-oriented graph – A new approach", Proc. SM2ACD, 2010, paper 1-13-06.

[7] M.C.C. Onete, C.E. Onete; "A novel condition for Hamiltonicity: Constructing Hamiltonian Circuits", Proc. EuroCon & Conftele 2011, accepted.