A novel condition for Hamiltonicity; Constructing Hamiltonian circuits

Maria Cristina C. Onete
CASED & TU Darmstadt
Darmstadt, Germany
cristina.onete@cased.de

Cristian E. Onete
IP & L Department
NXP Semiconductors
Eindhoven, The Netherlands
cristian.onete@ieee.org

Abstract— **In this paper, we derive a new necessary and sufficient condition for a simple, undirected graph to have a Hamiltonian circuit. Our novel approach uses a modified incidence matrix and constructs Hamiltonian circuits from the graph's spanning trees. We furthermore optimise the underlying spanning-tree algorithm, and give an example of our results.**

Keywords-component; Hamiltonian circuit, Hamiltonicity, Wang Algebra

I. INTRODUCTION

We consider here undirected graphs G(N, l) on N nodes and l links, which are simple, i.e. have no loops and no parallel links. Denote links as (i, j) where i and j are the endpoint nodes, and call two links *connected* if they share a node. A path is a subset of connected links. A node is *visited* by a path if it is an endpoint of one of the links in the path. A *circuit* is a path in which exactly one node is visited twice.

Finding a *Hamiltonian* circuit, i.e. one that visits N-1 nodes exactly once and the last node, twice, is a hard problem. A direct application of finding such a circuit is the design of ground paths in integrated circuits.

A graph is *Hamiltonian* if it has such a circuit. A necessary and sufficient condition for Hamiltonicity in a simple undirected graph is shown in [1]. The solution requires Diophantine equations and is based on different types of incidence matrices. Here we take a slightly different approach and construct Hamiltonian circuits by relying on the spanning-tree algorithm in [3]. This algorithm finds all the spanning trees in the graph; although exponential, the method is fully parallel, which aids the time complexity. Finding the trees is done by repeating a single sub-loop N-1 times; in this paper, we optimize the algorithm for a certain class of graphs, namely complete graphs, and we show how to reduce the problem of finding spanning trees to these simplified cases.

Our contributions – are as follows: (1) we give a characterisation of graphs and matrices which have Hamiltonian circuits, allowing us to (2) we present a necessary and sufficient condition for Hamiltonicity. Finally we (3) present and (4) optimize an algorithm to find Hamiltonian circuits using spanning trees.

II. HAMILTONIAN GRAPHS AND MATRICES

Let G(N, l) be a simple, undirected graph. If G is Hamiltonian, then its Hamiltonian circuit is the subgraph with N nodes and N links shown in Fig. 1. Without loss of generality, we re-label the nodes such that link l_k, k=1,…,N-1, connects nodes k and k+1, and l_N binds nodes N and 1.

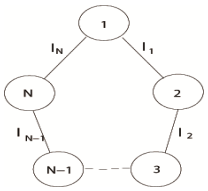

Fig. 1: A Hamiltonian sub-graph

The graph in Fig. 1 has an NxN square nodes-links incidence matrix as shown in equation (1).

$$I_M = \begin{bmatrix} 1 & 0 & 0 & \dots & -1 \\ -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix} \quad (1)$$

The signs in (1) refer to an arbitrary orientation of the links: a "1" at position (i,j) means link j leaves node i, and a "-1" means j arrives in i. We call i a *source* node in the first case, and a *destination* node otherwise. We denote by **S** and **D** the sets of source and destination nodes respectively; note that in a Hamiltonian circuit, each node is both a source and a destination node, but for different links.

Without loss of generality we can assume that for each link l_k=(k, k+1), k is a source and k+1, a destination node. In particular, the last link l_N is oriented from N to 1.

Note, importantly, that due to the fact that each node has exactly one link leaving it and one entering it, without loops and parallel links, the incidence matrix has exactly two non-zero entries on each row and column. This is characteristic for incidence matrices of Hamiltonian circuits, where only a single node is visited twice and the rest, only once.

In what follows we describe Hamiltonian cycles in terms of incidence matrices. Consider the diagonal NxN matrix L_N such that each nonzero entry $L_{k,k}=l_k=(k,k+1)$ – see (2).

We also use the following notation: for each link l_k=(k, $k+_N1$) we write ($k+_N1$, k) for the reverse orientation, i.e. for

the link taken with reversed orientation. We call this an *oriented-link notation*.

$$L_N = \begin{bmatrix} (1,2) & 0 & 0 & \dots & 0 \\ 0 & (2,3) & 0 & \dots & 0 \\ 0 & 0 & (3,4) & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & (N,1) \end{bmatrix} \quad (2)$$

We call the product $\mathbf{H}:=\mathbf{I_M}*\mathbf{L_N}$ a *Hamiltonian* matrix, as it describes a Hamiltonian circuit like the one in Fig. 1. The resulting **H** is shown in (3) in oriented-link notation.

$$\mathbf{H} = \begin{bmatrix} (1,2) & 0 & 0 & \dots & (1,N) \\ (2,1) & (2,3) & 0 & \dots & 0 \\ 0 & (3,2) & (3,4) & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & (N,1) \end{bmatrix} \quad (3)$$

Note that the Hamiltonian matrix tells us the orientation of the links incident to each node. Additionally, it has exactly two nonzero entries on each row and column. The graph in Fig. 1 is cyclic and simple, therefore the determinant of $\mathbf{I_M}$ is 0; thus, **H** is also singular. Its determinant is the product $(N, 1)*\prod^{N-1}_{k=1}(k, k+1)$ of main diagonal entries plus $(-1)^N$ times the product $(1, N)\prod^{N-1}_{k=1}(k+1, k)$. In particular, the first term is the Hamiltonian circuit where the links are traversed in the orientation of the links, whereas the second term is the same circuit in reverse order. Clearly $\mathbf{S} = \mathbf{D} = \{1, 2, \dots, N\}$.

These observations hold for any Hamiltonian circuit. In fact, if there existed a source node that is not also a destination node, the circuit would be incomplete; the row in the Hamiltonian matrix for this row would have only a single nonzero entry. If a row of the Hamiltonian matrix had, on the contrary, more than 2 nodes, this would indicate that the node is visited an extra time, thus creating circuits within the circuit. We characterise Hamiltonian matrices:

Proposition 1: A Hamiltonian Matrix **H** of a simple undirected graph G(N, l) is a square, NxN matrix of rank N-1, where each row and column have two non-zero entries and in the oriented-link notation, $\mathbf{S=D=\{}1, 2, \dots, N\mathbf{\}}$.

Note that the proposition does not specify the *position* of the nonzero entries in this matrix. In particular, we consider Hamiltonian matrices to be unique only up to permutation of their rows and columns. In the following, we generally call the matrix in (3) *the* Hamiltonian matrix of a graph.

The Hamiltonian matrix is simply the modified incidence matrix **MIM** [3, 4] of the graph in Fig. 1 in oriented-link notation. In general, the **MIM** is obtained by multiplying a diagonal l x l matrix having link l_k at position (k, k) by the links-nodes incidence matrix of a simple graph G(N, l), and by then using oriented-link notation to remove the signs (see [3] for details). Note that the links-nodes incidence matrix is the transposed of the nodes-links incidence matrix. For a more general treatment of incidence matrices we refer the reader to [2]. For example, the **MIM** of the complete graph in Fig. 2 is given in (4).

We formulate the following main result.

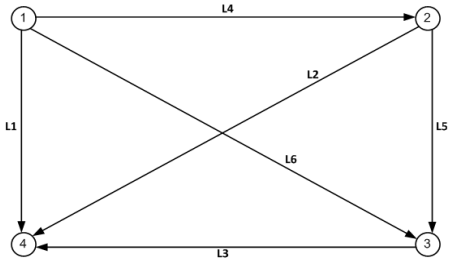

Fig. 2 A complete 4 nodes graph

$$\mathbf{MIM}=\begin{bmatrix} (1,4) & 0 & 0 & (1,2) & 0 & (1,3) \\ 0 & (2,4) & 0 & (2,1) & (2,3) & 0 \\ 0 & 0 & (3,4) & 0 & (3,2) & (3,1) \\ (4,1) & (4,2) & (4,3) & 0 & 0 & 0 \end{bmatrix} \quad (4)$$

Theorem 1: A graph G(N, l) is Hamiltonian if and only if its **MIM** contains an N x N submatrix which is Hamiltonian.

Proof: Clearly, if such a submatrix exists, then it describes a Hamiltonian circuit. Note that we do not require ALL the Hamiltonian circuits in the graph, but just one. Now suppose the graph is Hamiltonian, i.e. it has a Hamiltonian circuit. Assume towards contradiction that its **MIM** does not contain a Hamiltonian submatrix. For convenience we re-label the nodes and links in the graph such that the nodes are labelled consecutively with respect to the Hamiltonian circuit, and the links between them have the same labelling as that of Fig. 1. We then label the remaining links in the graph G consecutively, but in random order. We now write the incidence matrix of the graph G such that the first N columns correspond to the links in the Hamiltonian circuit and then compute the product with the diagonal matrix **L**. Clearly the resulting **MIM** will have the Hamiltonian matrix **H** of the circuit as the submatrix corresponding to the first N columns of the **MIM**. ■

Corollary 1: In particular a Hamiltonian N x N matrix corresponds to a minimal circuit in a graph on N nodes, i.e. a circuit which has no sub-circuits.

We illustrate Theorem 1 with the example of the Petersen graph, whose MIM is in equation (5).

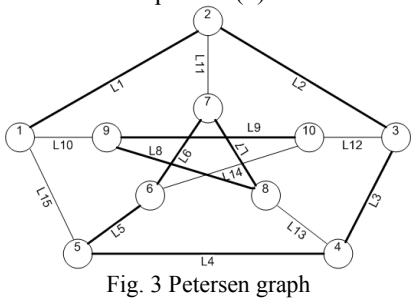

Fig. 3 Petersen graph

$$\mathbf{MIM}=\begin{bmatrix} (1,2) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (1,9) & 0 & 0 & 0 & 0 & (1,5) \\ (2,1) & (2,3) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (2,7) & 0 & 0 & 0 & 0 \\ 0 & (3,2) & (3,4) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (3,10) & 0 & 0 & 0 \\ 0 & 0 & (4,3) & (4,5) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (4,8) & 0 & 0 \\ 0 & 0 & 0 & (5,4) & (5,6) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (5,1) \\ 0 & 0 & 0 & 0 & (6,5) & (6,7) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (6,10) & 0 \\ 0 & 0 & 0 & 0 & 0 & (7,6) & (7,8) & 0 & 0 & 0 & (7,2) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & (8,7) & (8,9) & 0 & 0 & 0 & 0 & (4,8) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & (9,8) & (9,10) & (9,1) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (10,9) & 0 & 0 & (10,3) & 0 & (10,6) & 0 \end{bmatrix} \quad (5)$$

Note that although the first 9 rows and columns appear to form a Hamiltonian matrix, this matrix lacks one entry in the first row and on the last column. In fact, this graph has no Hamiltonian submatrix. It is a known result, however, that if one removes any node from this graph, it becomes Hamiltonian. We remove node 10 and all its adjacent links: L9, L12, and L14. The **MIM** of the modified graph becomes:

$$\mathbf{MIM}=\begin{bmatrix} (1,2) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (1,9) & 0 & 0 & (1,5) \\ (2,1) & (2,3) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (2,7) & 0 & 0 \\ 0 & (3,2) & (3,4) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & (4,3) & (4,5) & 0 & 0 & 0 & 0 & 0 & 0 & (4,8) & 0 \\ 0 & 0 & 0 & (5,4) & (5,6) & 0 & 0 & 0 & 0 & 0 & 0 & (5,1) \\ 0 & 0 & 0 & 0 & (6,5) & (6,7) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & (7,6) & (7,8) & 0 & 0 & (7,2) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & (8,7) & (8,9) & 0 & 0 & (4,8) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & (9,8) & (9,1) & 0 & 0 & 0 \\ \end{bmatrix} \quad (6)$$

We prove Hamiltonicity by constructing a Hamiltonian circuit. Note that if we remove one link from the circuit, we have a spanning tree. It is not, however, sufficient to find just one spanning tree, as it will not necessarily yield a Hamiltonian circuit.

Removing a circuit link means removing that column from the incidence matrix and the corresponding entry in the diagonal matrix L_N. We thus remove the same column from the Hamiltonian matrix. We can now remove the row corresponding to the link's source node from the Hamiltonian matrix, as this is also a destination node for another link. By removing the last row and column from **H**, we have $\mathbf{H_R}$ as in equation (5).

$$\mathbf{H_R}=\begin{bmatrix} (1,2) & 0 & 0 & \dots & 0 \\ (2,1) & (2,3) & 0 & \dots & 0 \\ 0 & (3,2) & (3,4) & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & (N-1,N) \end{bmatrix} \quad (7)$$

We call the spanning tree obtained by removing a single link from the Hamiltonian circuit a *Hamiltonian tree* and note that the reduced Hamiltonian matrix is a reduced incidence matrix (without a single node corresponding to the removed link) of the Hamiltonian tree.

Proposition 2: A reduced Hamiltonian matrix $\mathbf{H_R}$ of a Hamiltonian matrix **H** (of a graph G(N, l)) is the square, (N-1)x(N-1) matrix obtained by removing link (k, k+1) and node k from graph G. It holds that S_R={1, ..., N}\{k} and D_R={1, ..., N}\{k+1}. $\mathbf{H_R}$ is the **MIM** of a spanning tree.

Proof: This proposition follows trivially from the construction of the reduced Hamiltonian matrix. ■

Corollary 2: Given a reduced Hamiltonian matrix H_R and a graph G(N,l), such that $\mathbf{H_R}$ is a submatrix of the graph's **MIM**, denote the subset of links l_R to be the links used in the Hamiltonian tree. If there exists a link (i,j) or its reverse (j,i) in the set l\l_R with i≠j, $\mathbf{S_R}$ = {1, 2, ..., N} \{i}, and $\mathbf{D_R}$ = {1,2, ... , N} \ {j}, then the graph G is Hamiltonian.

Proof: From Proposition 2, we conclude that the reduced Hamiltonian matrix $\mathbf{H_R}$ corresponds to a spanning tree; as $\mathbf{S_R}$ = {1, 2, ..., N} \{i} and $\mathbf{D_R}$ = {1,2, ... , N} \ {j}, it clearly follows that adding the link {i,j} will complete a Hamiltonian circuit, thus making the graph Hamiltonian. ■

This Corollary hints at how to find Hamiltonian circuits in a graph once we can find ALL the spanning trees. Indeed, we describe the following general algorithm:

Algorithm 1 [Finding a Hamiltonian Circuit]:

Input: Graph G(N, l), Spanning Tree Algorithm; **Output:** Hamiltonian circuit **C** or statement Not Hamiltonian

1. Find Spanning Tree **T** with reduced Hamiltonian matrix $\mathbf{H_R}$ as a submatrix of **MIM.** Write l_R, the list of N-1 links in this spanning tree.
2. List source nodes and destination nodes. If any source or destination node repeats, abort and find the next spanning tree. If no nodes are repeated, identify nodes i and j as in Corollary 2. If there exists a link in l\l_R with endpoints i, j, then add (i,j) to the set of links in l_R. Output the refreshed list of links as the Hamiltonian circuit C. Otherwise, repeat step 1.
3. If no circuit is found, return: Not Hamiltonian.

Note that if we want to find all the Hamiltonian circuits, rather than only one, we can repeat the algorithm until they are all found. We use the spanning-tree enumeration method in [3], but optimise it as shown in section III.

III. FINDING ALL THE SPANNING TREES IN A GRAPH

The general problem of finding spanning trees is useful for electronic circuits, as each spanning tree is a term in the determinant associated with a circuit, which can thus be computed in an inverse-free manner.

Spanning trees are generally found in these general steps [3]: (1) Find and remove reference node data from the links-nodes incidence matrix; denote **U**= $\mathbf{MIM}^T$ of corresponding tree, where **U** has at most N-1 rows with a single nonzero entry (corresponding to the nodes linked to the reference node), and a number of rows with two nonzero entries. (2) In a top-down fashion, choose at least one row with a single nonzero entry and combine with N-2 other rows. (3) Test the resulting submatrix to see if it yields a spanning tree. If it does, return the tree. Repeat step (2).

This method is exponential, but can be run in parallel, which is a great advantage. Additionally, multiple components connected by single isthmi can be treated apart, and the complexity of finding the spanning trees in the entire graph is reduced to the same complexity in the largest component.

However, the method has two great disadvantages. Firstly, it does not take advantage of specific graph properties, such as symmetry. Secondly, the algorithm is run fully for each graph, i.e. all the combinations with at least one row with a single nonzero entry are tried out.

We modify the spanning-tree method in [3] leaving from the observation that, although the runtime is greatest when the graph is complete, the graph's symmetry allows us to terminate faster. In particular, let G(N, l) with l = N(N-1)/2 be complete. The nodes of this graph are perfectly symmetric in the sense that one can relabel them without in any way changing the graph. This also means that given some spanning tree T, we get another N-1 trees by introducing a permutation on the nodes. Take for example the spanning tree with link set {(1,2)(1,3) .... (1, N)}. Now introduce a permutation in the labelling of the nodes, such that node k becomes node k+1.

We have another spanning tree, namely {(2,3)(2,4) ... (2, 1)}. This process can be repeated N-1 times. In Fig. 2, given tree {(1,4)(2,4)(3,4)} we find related trees: {(2,1)(3,1)(4,1)}, {(3,2)(4,2)(1,2)}, {(4,3)(1,3)(2,3)}.

Complete graphs are symmetric, thus we can "fix" one link, i.e. we can simply fix the first row with a single nonzero entry and form just all the combination with that one. We are sure to find all the spanning trees in this way, because the graph is symmetric, so by using the permutations we are sure to have tried out all the other possibilities implicitly.

An advantage of working with complete graphs is that we know exactly how many spanning trees there are in a complete graph (namely, N^{N-2}), thus we can stop after finding them. In practice, this will be very fast if we use the method shown above, taking an average of ½ of all the possible combinations with the first row.

In short, the optimised method consists of the following large steps: (1) Initialise the **MIM** as before. Set a counter C to N^{N-2} and initialise the list of trees T = Φ. (2) Run algorithm top-down for the first row with a single nonzero entry until finding one spanning tree. Do C:=C-1 and add the tree to T. If C = 0 halt and output T. (3) By permutation as shown above, find the related other N-1 trees and compare them with the trees in T. If the tree is new, add it to T and do C:= C-1. (4) Repeat from step (2) until C=0.

What if the graph is not complete? We suggest the following method for graphs which are "almost" complete, i.e. whose nodes all have a degree at least equal to N(N-1)/4.

Algorithm 2 [Spanning trees for almost complete graphs]
Input: graph G(N, l); **Output**: all spanning trees of G

1. Initiate the set A= Φ. For each pair of unconnected nodes (i,j) add this link both to the graph and to A. Continue until the graph is complete.
2. Find all the spanning trees in the complete graph using the optimised method presented above. Output the list T of trees.
3. For each tree, compare its link list to A. If the intersection of the two sets is not the empty set, remove the tree from the list. Repeat until all the trees have been checked.

The complexity gain is twofold: firstly, the algorithm is only run for a single row with a single nonzero entry, rather than for a maximum of N-1 such rows. Concretely, this means reducing the run time to almost a half, while not losing any of the parallelism. Secondly, we have a stopping criterion that allows us to halt before all the possibilities are exhausted, if the graph is complete. On the average, this will save us on average another factor ½ of the complexity.

Take the worked example in [3] for the graph in Fig. 4, whose transposed MIM is in equation (8). With Algorithm 2, we find: 1) {(1,4)(2,4)(3,4)}, related trees: {(2,1)(3,1)(4,1)}, {(3,2)(4,2)(1,2)}, {(4,3)(1,3)(2,3)}; 2) {(1,4)(2,4)(3,2)}, related tree {(2,1)(3,1)(4,3)}; 3) {(1,4)(2,4)(3,1)}, related trees {(2,1)(3,1)(4,2)}, {(3,2)(4,2)(1,3)}, {(4,3)(1,3)(2,4)}, {(1,4)(2,4)(3,1)}; 4) {(1,4)(3,4)(2,1)}, related trees {(2,1)(4,1)(3,2)}, {(3,2)(1,2)(4,3)},{(4,3)(2,3)(1,4)}; 5) {(1,4)(2,3)(3,1)}, related tree {(2,1)(3,4)(4,2)} HALT. The algorithm is faster than [3] by a factor at least equal to 16/5.

$$\mathbf{U} = \begin{bmatrix} (1,4) & 0 & 0 \\ 0 & (2,4) & 0 \\ 0 & 0 & (3,4) \\ (1,2) & (2,1) & 0 \\ 0 & (2,3) & (3,2) \\ (1,3) & 0 & (3,1) \end{bmatrix} \quad (8)$$

Finally, take the reduced Petersen graph (i.e. the graph in Fig 3 without node 10). Choose reference node 1 and remove that row from the **MIM.** The degree of each node is smaller than 4, so we do not "complete" the graph, but work with it as it is. We show the resulting **MIM**:

$$\mathbf{MIM}= \begin{bmatrix} (2,1) & (2,3) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (2,7) & 0 & 0 \\ 0 & (3,2) & (3,4) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & (4,3) & (4,5) & 0 & 0 & 0 & 0 & 0 & 0 & (4,8) & 0 \\ 0 & 0 & 0 & (5,4) & (5,6) & 0 & 0 & 0 & 0 & 0 & 0 & (5,1) \\ 0 & 0 & 0 & 0 & (6,5) & (6,7) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & (7,6) & (7,8) & 0 & 0 & (7,2) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & (8,7) & (8,9) & 0 & 0 & (4,8) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & (9,8) & (9,1) & 0 & 0 & 0 \end{bmatrix} \quad (9)$$

Since the matrix is so wide, we prefer to use it like this instead of transposing it. We will have to now look for the columns with 1 nonzero entry. There are three such columns: 1, 9, 12. We only have to make combinations with the first column according to algorithm 2, so we need not permute the columns. In fact, we immediately have a spanning tree: {(2,1)(3,2)(4,3)(5,4)(6,5)(7,6)(8,7)(9,8)}; it holds that S_R={2, ..., 9} and D_R = {1, ... , 8}, In particular, link (1,9) is missing: we find this link in the graph and thus complete the Hamiltonian circuit.

IV. CONCLUSIONS

In this paper we show both a necessary and sufficient condition for a simple, undirected graph to be Hamiltonian, and a method to construct a Hamiltonian circuit, or even all the Hamiltonian circuits. This method relies on finding all the spanning trees in a graph, problem for which we give an optimised algorithm. Our examples also concretely show the improvements we have made to previous work, and we show that our results are compatible with well-known results in graph theory.

REFERENCES

[1] 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.

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

[3] C. E. Onete, M.C.C. Onete; "Enumerating all the spanning trees in an un-oriented graph – A new approach", SM2ACD, 2010, accepted for publication.

[4] 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.