Hamiltonian Graphs and the Traveling Salesman Problem

Dhananjay P. Mehendale
Sir Parashurambhau College, Tilak Road, Pune 411030, India

Abstract

A new characterization of Hamiltonian graphs using $f$-cutset matrix is proposed. A new exact polynomial time algorithm for the traveling salesman problem (TSP) based on this new characterization is developed. A new characterization of shortest Hamiltonian tour for a weighted complete graph satisfying triangle inequality (i.e. for tours passing through every city on a realistic map of cities where cities can be taken as points on a Euclidean plane) is also proposed.

1. Characterization of Hamiltonian Graphs using $f$-cutset Matrix: Let $G$ be a $(p, q)$ graph, i.e. a graph on $p$ points (vertices) and $q$ lines (edges) with the following vertex set $V(G)$ and edge set $E(G)$ respectively:

$V(G) = \{v_1, v_2, \cdots, v_p\}$ and

$E(G) = \{e_1, e_2, \cdots, e_q\}$

Let $A_G = [a_{ij}]_{p \times p}$ denotes the adjacency matrix of $G$.

By choosing a spanning tree in the given connected graph one can construct the fundamental cutest matrix ([1], page 153) associated with this choice of tree in the form

$C_f = [C_c : I_{p-1}]$

where the last $(p - 1)$ columns forming the identity matrix correspond to $(p - 1)$ branches of the spanning tree, and the first $(q - p + 1)$ columns forming matrix $C_c$ correspond to the chords. The presence of entry “1” in a column and row of $C_f$ depicts the presence of the edge represented by that column in the $f$-cutest represented by that row. Each row in $C_f$ is a fundamental cutest vector. The rank of $C_f$ is $(p - 1)$ and these fundamental cutest vectors form the vector space basis of the cutest space,
which is subspace of the vector space associated with the graph. If we develop an algorithm which selects edges from the fundamental cutsets ($f$-cutsets) to form a Hamiltonian circuit then it is clear to see that

1) Since every vertex must get incorporated in every Hamiltonian circuit and only once, so, every $f$-cutset must contribute positive and even number of edges.

2) Since $C_f$ partitions into $C_c$ and Identity matrix, $I_{p-1}$, and since every row of an identity matrix contains at most one nonzero entry, so every $f$-cutset must contribute at least one chord to every Hamiltonian circuit in order to maintain evenness and positivity of the number of edges chosen on the corresponding $f$-cutset. Thus, the presence of a branch from every $f$-cutset in a Hamiltonian circuit is not necessary but the presence of at least one chord from every $f$-cutset is a must for a Hamiltonian circuit formed by chords and branches.

**Theorem 1.1:** Let $G$ be a $(p, q)$ graph. $G$ is Hamiltonian if and only if we can select even number (at least two) of edges on each row representing an $f$–cutset to form a connected graph such that at least one of the selected edge is a chord and the total count of thus selected edges (chords + branches) is $p$.

**Proof:** Let $G$ be a Hamiltonian graph. So, there is a Hamiltonian circuit in $G$. So, there exists a tree which is Hamiltonian path. Take this tree which is a Hamiltonian path containing $(p - 1)$ edges. Form $f$-cutset matrix for this tree. Among the chords represented by columns of $C_c$ there will exist a chord (which is actually the remaining part of the Hamiltonian circuit to be added to the tree equal to the Hamiltonian path to complete the Hamiltonian circuit) such that its corresponding column will be entirely made up of units. Thus, we take this chord and the corresponding branch on every $f$-cutset, in effect, even (= 2) number of edges are selected from each row representing an $f$–cutset and the total count of thus selected edges (1 chord + $(p - 1)$ branches) is equal to $p$, as desired forming a connected graph.

Suppose we have formed a subgraph of the given graph containing at least two edges on each row corresponding to every $f$–cutset to form a connected subgraph such that at least one is a chord, so that, in effect even (>$0$) number of edges get selected from each row representing an $f$–cutset and the total count of thus selected edges (chords + branches) is $p$. We show that this subgraph must be a Hamiltonian circuit. It can be easily seen that such a graph can have only two possibilities: Either it is a circuit of
length $p$, or a subcircuit of length smaller than $p$ with some incident (one or more) paths to the vertices of this subcircuit. In the first case nothing to prove. In the other case some $f$-cutset among the $f$-cutsets to which the edge incident on the pendant point of the path belongs, may be as a branch or a chord, must be contributing odd number of edges since there is no provision to reach and go away in the subgraph from this pendant point, a contradiction to the data.

$$\Box$$

**Some Interesting Observations:** (1) Search whether there is a column vector in $C_c$ entirely containing units (one unit in each row) i.e. in total $(p-1)$ units. In this case, this edge (chord) represented by the column of units and the edges representing all branches together sum up to in all $1+(p-1) = p$, edges forming a Hamiltonian circuit. Thus, the graph will be Hamiltonian.

(2) Search whether there are some two columns vectors in $C_c$ such that there is a unit in some rows in that column corresponding to a chord and there is a unit in the remaining rows in the column corresponding to the other chord among the total $(p-1)$ rows with exactly one overlap, i.e. there exists exactly one row which contains units in the chosen columns corresponding to both the chords. In this case, the two chosen edges (chords) represented by the two columns (determined as above with exactly one overlap) of $C_c$ and the edges representing all branches except the branch defining an $f$-cutset for which the corresponding row contains units in both the columns corresponding to the two chosen chords, together sum up to in all $2+(p-2) = p$, edges forming a Hamiltonian circuit. Thus, the graph will be Hamiltonian.

(3) Continuing on these lines, search if there exist some $k$ columns vectors in $C_c$ in which the units are distributed among these determined columns of $C_c$ such that there will exist in all $(k-1)$ overlaps as mentioned in (2), such that the $k$ chosen edges (chords) represented by the $k$ determined columns of $C_c$ and the edges representing all branches except the branches defining an $f$-cutset for which the corresponding row contains units in even number of columns corresponding to the $k$ chosen chords (i.e. there is overlap), and together sum up to in all $k+(p-k) = p$, edges forming a Hamiltonian circuit. Thus, the graph will be Hamiltonian.
We now state few definitions and develop a polynomial time algorithm using them:

**Definition 1.1:** A lattice is a rectangular array of dots made up of some \( m \) rows and \( n \) columns.

**Definition 1.2:** A lattice-cutset-graph associated with a \( f \)-cutset matrix is a graph obtained from the lattice of the size of \( f \)-cutset matrix obtained by treating those dots in the array as vertices of this new graph where, at the same place entry “1” is present in the \( f \)-cutset matrix, then joining these vertices in each column by vertical edges so that a path is formed along each column and further joining by horizontally going edges connecting all the vertices in a row to each other.

**Definition 1.3:** An induced-cutset--tree is a tree in the usual sense in the lattice-cutset-graph (and not in the original graph) such that its vertices belong to the first \((q - p + 1)\) columns of lattice-cutset-graph and there exists at least one vertex belonging to every row of vertices of the lattice-cutset-graph and when some columns of vertices are chosen we should take all the vertical edges belonging to those columns and all the horizontally going edges joining the vertices in succession belonging to same row in the chosen columns.

**Definition 1.4:** An extended-induced-cutset-tree is the one obtained by induced-cutset-tree by adding horizontally going edges such that one horizontal edge is to be added connecting last (in fact, any vertex will do) vertex present in that row (belonging to the \( C_c \) part) to the vertex belonging to the last \((p - 1)\) columns (belonging to the Identity part) in the same row only when the number of vertices that get incorporated in that row for the induced-cutset-tree are odd in number (due to the odd number of edges got selected in the \( C_c \) part with respect to that row).

A simpler algorithm will be given below which just consists of finding an induced-cutset-tree and then an extended-induced-cutset-tree from this induced-cutset-tree.

**Remark 1.1:** Forming an induced-cutset-tree and then the extended-induced-cutset-tree automatically takes care of the important requirements in
the above given observations of achieving exactly \((k-1)\) overlaps when \(k\) chords are chosen, so that the important equation to be satisfied by the chosen chords and branches to form a Hamiltonian circuit, namely, chords + branches = \(p\) is satisfied automatically.

**Steps to construct a lattice-cutset-graph and an extended-induced-cutset-tree as its subgraph from \(f\)-cutset matrix:**

1. Form \(f\)-cutset matrix, \(C_f = [C_c : I_{p-1}]\).
2. Form a lattice of size (rows and columns of dots) equal to the size of the \(f\)-cutset matrix, made up of dots and then proceed to form lattice-cutset-graph by taking the dots with entry “1” in the same place in the associated fundamental cutset matrix and then by connecting the appropriate edges as per the above definition.
3. Form an induced-cutset-tree from edges in the first \((q - p + 1)\) columns (forming matrix \(C_c\) of the \(f\)-cutset matrix that correspond to the chords) of the lattice-cutset-graph by choosing appropriate columns such that at least one vertex from each row of vertices gets incorporated.
4. Count the number of vertices belonging to each row of this tree and when the number of vertices contained in the row (corresponding to the \(C_c\) part of the \(f\)-cutset matrix) are odd in number then extend the tree by joining last (right most) vertex in the tree to the vertex in the same row and belonging to the last \((p-1)\) columns (corresponding to the identity matrix representing the \((p-1)\) branches of the \(f\)-cutset matrix) by a new edge to obtain an extended-induced-cutset-tree.

**Example 1.1:** Consider the following graph, \(H\) say.
We take the spanning tree, $T$, formed by edges \{a, b, c, d, e, f, g\} of this graph.

Then the $f$--cutset matrix, $C_f$ can be expressed as follows, where the first column represents the labels of the $f$--cutsets while the first row represents the labels of chords and branches. The $f$-cutset matrix, $C_f = [C_c : I_{p-1}]$ for the present case can be written as

$$
C_f = \begin{bmatrix}
    c_1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
    c_2 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
    c_3 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
    c_4 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
    c_5 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
    c_6 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
    c_7 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}
$$

Consider the induced-cutset-trees, $T_1$ and $T_2$, containing chords \{j, l\} and chords \{h, i, l\} respectively, as shown below. An extended-induced-cutset-tree, $ET_1$ is also shown as an example. The first row of alphabets depicts the used edge-labels:
Now,

(i) Using chords \( \{j, l\} \) we form a tree, \( T_1 \), formed in \( C_c \) and append the appropriate branches \( \{a, b, c, e, f, g\} \) leading to formation of extended tree, \( ET_1 \). This leads to the following Hamiltonian circuit:
\[ a \rightarrow b \rightarrow c \rightarrow f \rightarrow l \rightarrow e \rightarrow g \rightarrow j \rightarrow a \]

(ii) Using the chords \( \{h, i, l\} \) we can see that we have a tree, \( T_2 \), containing formed in \( C_c \) and appending the appropriate branches \( \{a, b, e, f, g\} \) we can get an extended tree, say \( ET_2 \), which will lead to the following Hamiltonian circuit: \( a \rightarrow b \rightarrow i \rightarrow g \rightarrow e \rightarrow l \rightarrow f \rightarrow h \rightarrow a \).

**Theorem 1.2:** An extended-induced-cutset-tree forms a Hamiltonian circuit if the total count of the chosen edges represented by columns of lattice-cutset-graph (chords + branches) is equal to \( p \).

**Proof:** Straightforward.

**Remark 1.2:** The edges \( i \) and \( j \) together cannot belong to any Hamiltonian circuit of graph \( H \) because we can’t form an extended-induced-cutset-tree along with other edges such that the count of the selected edges (chords + branches) = \( p \).
2. The Traveling Salesman Problem (TSP): This well-known problem asks for an efficient (polynomial time) algorithm to find shortest Hamiltonian circuit (or cycle), i.e. the one with smallest weight sum of its edges in a weighted complete graph. No efficient exact algorithm for this problem is known. It has been shown in the literature that the problem of finding Hamiltonian path between two pre-specified vertices, or that of finding Hamiltonian circuit, or that of finding shortest Hamiltonian tour for the traveling salesman etc. are all belong to the large class of NP complete problems (Page 234, Theorem 8.9 [2]). In fact it has been further shown that when the triangle inequality is not satisfied the traveling salesman problem is non-approximable unless NP complete problems have polynomial time solutions [3].

There exist some well-known efficient heuristic algorithms for TSP. The simplest one is the so called nearest-neighbor-method [4] with performance guarantee, $\alpha = \frac{1}{2}(\lceil \ln(n) \rceil + 1)$, and the efficient one is the so called minimum-weight- matching-algorithm [5] with performance guarantee $\alpha < \frac{3}{2}$.

In order to initiate the discussion we begin with a possible modification in the so called nearest-neighbor-method. The nearest-neighbor-method starts from a vertex $v_i$ in a weighted complete graph and select an edge among the edges the adjacent vertex $v_j$ such that the weight of the edge $e_k = (v_i, v_j)$ is minimum among the edges emerging from the vertex $v_i$. It continues with the same criterion for the incorporation of the next edge from vertex $v_j$ to a new vertex (i.e. not already visited one till all the vertices are exhausted and one has to select now the only left out choice $v_k$ to $v_i$ to complete the formation of the Hamiltonian circuit). Thus, to new nearest neighbor of $v_j$. Since the decision for choosing edge in this algorithm is based on purely local considerations, i.e. the selection is made which is locally best; there is no guarantee of this algorithm of attaining a good Hamiltonian circuit.

We propose below a modification in this algorithm which will make it somewhat global and thus will improve the chance of getting better performance. Let us take the given weighted complete graph as a symmetric digraph and each edge as two directed edges of same weight.
directed in opposite directions. It is clear to see that when one selects an
edge, say \((v_i, v_j)\), then one cannot select any other edge emerging from \(v_i\)
and any edge entering in \(v_j\). Thus, when one selects edge \((v_i, v_j)\) the
weight sum of (other) edges emerging from \(v_i\) and weight sum of (other)
edges entering in \(v_j\), other than edge \((v_i, v_j)\) gets excluded. Let us denote
the weight-inclusion-adjacency- matrix by \(WIA = [w^i_{jk}]_{p \times p}\), where \(w^i_{jk}\)
weight of the edge \((v_j, v_k)\) that get included in the weight sum of edges of
a Hamiltonian circuit when one selects the edge \((v_j, v_k)\) while forming that
Hamiltonian circuit. Similarly, let us define the weight-exclusion-
adjacency- matrix, \(WEA = [w^{e}_{jk}]_{p \times p}\), where \(w^{e}_{jk}\) = weight sum of the
edges that get excluded while one selects the edge \((v_j, v_k)\) as an edge for a
Hamiltonian circuit. Thus, \(w^{e}_{jk} = \sum_{l \neq j} w^i_{lk} + \sum_{m \neq k} w^i_{jm}\)
(As a further modification, we can add in \(w^{e}_{jk}\) the weight sum of those
edges which form a subcircuit with earlier selected edges and so can’t be
part of a Hamiltonian circuit. To keep the things simple here we do not take
into consideration this further modification.)

2.1 Modified-Nearest-Neighbor-Method: In the nearest-neighbor-method
there is only one criterion that is followed: the minimum weight nearest
neighbor is selected to join, in succession by starting from some vertex, till
one forms the Hamiltonian circuit. To make this algorithm somewhat global
one follows the same selection method by imposing two criteria:

Algorithm 2.1.1: (1) Using the given \(WIA = [w^i_{jk}]_{p \times p}\), construct the
weight-exclusion-adjacency-matrix, \(WEA = [w^{e}_{jk}]_{p \times p}\).
(2) Select an edge which obeys two criteria in any order:
   (i) The weight that gets included due to this selection is minimum.
   (ii) The weight that gets excluded due to this selection is maximum.
(3) Continue on these lines till one gets the desired Hamiltonian circuit. □
Note that observing the criteria (i), (ii) in (2) in any order and at any stage of the selection it is possible to carry out the nearest-neighbor-algorithm, i.e. condition (i) is observed first and then (among the choices) condition (ii) is observed next, or, vice versa, and at any stage of the selection, i.e. we can change the order of conditions to be observed even at any intermediate stage of the algorithm.

2.2 A Heuristic for TSP using Contractions: The following heuristic is certainly an improvement over the usual nearest-neighbor-method because it eliminates the restriction of choosing only adjacent edges in succession imposed in the nearest-neighbor-method:

Algorithm 2.2.1: (1) Choose an edge among the edges with smallest weight in the given weighted complete graph on \( p \) points and contract it. Keep the record of the contracted edge. This leads to formation of weighted complete graph on \( p-1 \) points.
(2) Repeat the procedure in step (1) for the resulted weighted complete graph on \( p-1 \) points till we reach to a simple graph with single vertex.
(3) Build the Hamiltonian circuit using the contracted edges whose record has been kept at every stage of contraction.

Remark 2.1: In the nearest-neighbor-method we select smallest edge among the adjacent edges but in the above given heuristic algorithm 2.4 each time we select the smallest available edge, not necessarily adjacent one, at each stage of selection. This idea of contraction can be taken up also in the modified-nearest-neighbor-method which will produce a complete digraph at each time, may be asymmetric in weight after some iterations, to continue the same procedure till a Hamiltonian circuit is formed.

2.3 A Method to Estimate Performance of any Heuristic: We use any heuristic algorithm to obtain a reasonably good Hamiltonian circuit in a weighted complete graph. We form, \( WVAB(G) \), the weighted-vertex-adjacency-bitableau, for the given weighted complete graph. We break the entries in the rows of the right tableau into arrays of columns such that the first array contains labels of vertices containing smallest weight in front of them written in the bracket. The weights of the entries in the successive arrays form the non-decreasing order.
When we get the Hamiltonian tour made up of entries entirely belonging to first array we have assuredly got an exact solution for the TSP. In other words, if the subgraph formed by the edges in the first array is Hamiltonian then every Hamiltonian circuit in it is a shortest Hamiltonian circuit. It is also clear that the value (sum of weights of edges) of any shortest Hamiltonian circuit should be at least equal to the sum of weights of the entries (written in the brackets in front of them) obtained by taking one entry from each row of the first array.

We now systematically describe the procedure of estimating the possible distance of the exact solution and the solution obtained by any heuristic in the following steps:

**Algorithm 2.3.1:**

1. Construct $WVAB(G)$, which is $VAB(G)$ with weights of the corresponding edges written in the brackets in front of the numbers in the right tableau.
2. Sort the numbers in the rows of the right tableau and arrange them in non-decreasing order of their weights written in the brackets and thus obtain sorted $WVAB(G)$, say $SWVAB(G)$.
3. Carry out partitioning of $SWVAB(G)$ into arrays of the entries in the right tableau such that the first array contains the entries with smallest weight in all the rows, the second column contains next smallest entries in all the rows, and so on and thus construct table of sorted arrays, say $SWA(G)$.
4. Using any of the heuristics obtain a Hamiltonian circuit (which will contain as many as possible entries belonging to arrays with smaller array numbers, depending upon the performance guarantee of the used heuristic.
5. Suppose the Hamiltonian circuit formed contains entries from first, second, ..., $p$-th rows belonging respectively to $i_1$-th, $i_2$-th, ..., $i_p$-th array and let the difference in weights in the entries in the $i_1$-th, $i_2$-th, ..., $i_p$-th arrays and first array in the respective first, second, ..., $p$-th rows be $w_{i_1}, w_{i_2}, \ldots, w_{i_p}$ respectively then the circuit obtained could differ from the shortest circuit by amount **at most** equal to $\sum_{j=1}^{p} w_{i_j}$.
**Example 2.3.1:** Consider the following $WVAB(K_6)$ for the weighted complete graph on six points:

$$WVAB(K_6) = \begin{bmatrix}
1 & 2(2) & 3(3) & 4(4) & 5(1) & 6(1) \\
2 & 1(2) & 3(1) & 4(3) & 5(2) & 6(3) \\
3 & 1(3) & 2(1) & 4(4) & 5(3) & 6(4) \\
4 & 1(4) & 2(3) & 3(4) & 5(4) & 6(3) \\
5 & 1(1) & 2(2) & 3(3) & 4(4) & 6(2) \\
6 & 1(1) & 2(3) & 3(4) & 4(3) & 5(2)
\end{bmatrix}$$

$$SWVAB(K_6) = \begin{bmatrix}
1 & 5(1) & 6(1) & 2(2) & 3(3) & 4(4) \\
2 & 3(1) & 1(2) & 5(2) & 4(3) & 6(3) \\
3 & 2(1) & 1(3) & 5(3) & 4(4) & 6(4) \\
4 & 2(3) & 6(3) & 1(4) & 3(4) & 5(4) \\
5 & 1(1) & 2(2) & 6(2) & 3(3) & 4(4) \\
6 & 1(1) & 5(2) & 2(3) & 4(3) & 3(4)
\end{bmatrix}$$

$$SWA(K_6) = \begin{bmatrix}
1 & 5(1) & 6(1) & 2(2) & & 3(3) & 4(4) \\
2 & 3(1) & 1(2) & 5(2) & & 4(3) & 6(3) \\
3 & 2(1) & 1(3) & 5(3) & & 4(4) & 6(4) \\
4 & 2(3) & 6(3) & 1(4) & 3(4) & 5(4) & \\
5 & 1(1) & 2(2) & 6(2) & & 3(3) & 4(4) \\
6 & 1(1) & 5(2) & & 2(3) & 4(3) & 3(4)
\end{bmatrix}$$

Now the sum of weights of entries in the first array, obtained by taking weight of (any) one entry from each row is equal to 8 units, so, the shortest Hamiltonian circuit (when it can be formed using entries in the first array) will have weight at least equal to 8 units. It is easy to see that we can’t form a Hamiltonian circuit using entries only from first array. Now, using some approximation algorithm suppose we obtain the circuit as follows:
1 → 6 → 5 → 2 → 3 → 4 → 1. Then, this Hamiltonian circuit could be away from the exact solution at most by $0 + 1 + 1 + 0 + 3 + 1 = 6$ units.

2.4 An Exact Algorithm for TSP using $f$–Cutset Matrix: We construct a lattice, lattice-cutset-graph, and proceed to obtain shortest-extended-induced-cutset-tree, by any efficient shortest tree finding algorithm. We proceed in the following steps:

**Algorithm 2.4.1:**

(1) Construct $f$-cutset matrix,

$$ C_f = [C_c : I_{p-1}]_{q \times (p-1)} $$

with respect to some spanning tree for the given weighted complete graph.

(2) Construct lattice, of size $q \times (p - 1)$, made up of dots.

(3) Construct lattice-cutset-graph using the given $f$-cutset matrix.

(4) Assign weight equal to the weight of the edge in the originally given weighted complete graph represented by a column in $C_c$ to first (or some) vertical edge belonging to lattice-cutset-graph in that column and assign weight “0” to all other vertical edges in that column (in order to avoid the multiple counting of that weight of that edge represented by the vertex pair written in the top row in the original graph in the desired shortest Hamiltonian circuit) and do this procedure for all columns.

(5) Assign weight equal to the weight corresponding to a branch to all edges reaching a vertex in the last $(p - 1)$ columns (branch part) and do the same for all branches.

(7) Assign weight equal to “zero” to all other horizontal edges connecting the points belonging to $C_c$ part when there exists at least one vertical edge emerging from its both ends.

(8) Assign weight to a horizontal edge connecting the points belonging to $C_c$ part equal to the weight of the edge of the chord represented by the column in which the pendant point belongs and has no vertical edge emerging from it.

(9) Choose columns containing smallest nonzero weight for the vertical edge (note that when we select a vertical edge we have by this act actually chosen a vertical path joining all the “1s” in succession on that column). Then at each stage of selection select the next column containing smallest nonzero edge among the remaining columns such that the
choice leads to formation of shortest induced-cutset-tree. 
(10) Find the shortest extended-induced-cutset-tree in this graph, by extending the induced-cutset-tree by horizontally going edges to the vertex in the last \((p - 1)\) columns in those rows containing odd number of vertices in the first \((q - p + 1)\) columns which will lead to formation of a desired Hamiltonian circuit.

**Remark 2.4.1:** Thus, the algorithm essentially consists of finding shortest-induced-cutset-tree in the lattice-cutset-graph by any efficient algorithm for finding shortest tree, similar to the one due to Kruskal [7] or Prim [8]. Actually this tree is not spanning i.e. does not contain every vertex (vertices in the lattice-cutset-graph are actually representing edges of the original graph present in different cutsets and the number of vertices associated with the same edge is equal to number of \(f\)-cutsets in which the edge is present) of lattice-cutset-graph. Also, first we have to obtain shortest-induced-cutset-tree, and it is important to note that its determination is not degree constrained, and later the extended-shortest-induced-cutset-tree to achieve evenness (the degree constraint comes here but the procedure does not become difficult due to this constraint, here one only need to extend the shortest-induced-cutset-tree by appending the branches wherever required to maintain the required evenness of selected vertices on a row) by adding of selected vertices in each row of lattice graph.

**Remark 2.4.2:** Note that assigning weights to the edges of extended-induced-cutset-tree is set up in a way described in the steps of the algorithm essentially to include the weights of actual chords and branches in the original graph forming the desired Hamiltonian circuit (again, in the original graph) only once.

**Example 2.4.1:** Consider weighted complete graph of example 2.2.1, and take any tree, say \(T\), as follows:
The corresponding $f$-cutset matrix for this tree is given below:

$$
C_f = \begin{bmatrix}
(1,4) & (1,6) & (2,3) & (2,4) & (2,5) & (2,6) & (3,4) & (3,5) & (3,6) & (4,6) & (1,2) & (1,3) & (1,5) & (4,5) & (5,6) \\
C_1 & 0 & 0 & l(1) & l(3) & l(2) & l(3) & 0 & 0 & 0 & 0 & l(2) & 0 & 0 & 0 & 0 \\
C_2 & 0 & 0 & l(1) & 0 & 0 & 0 & l(4) & l(3) & l(4) & 0 & 0 & l(3) & 0 & 0 & 0 & 0 \\
C_3 & l(4) & l(1) & 0 & l(3) & l(2) & l(3) & l(4) & l(3) & l(4) & 0 & 0 & 0 & l(1) & 0 & 0 & 0 \\
C_4 & l(4) & 0 & 0 & l(3) & 0 & 0 & l(4) & 0 & 0 & l(3) & 0 & 0 & 0 & l(4) & 0 \\
C_5 & 0 & l(1) & 0 & 0 & 0 & l(3) & 0 & 0 & l(4) & l(3) & 0 & 0 & 0 & 0 & l(2) & 0
\end{bmatrix}
$$

The first row in the above matrix contains pair of vertices noting edges while the first column records the labels of fundamental cutsets.

As per the above algorithm 2.4.1 columns corresponding to edges \{(2,3), (3,4), (4,6)\} together form the desired shortest induced-cutset-tree and when appended appropriately by columns corresponding to edges \{(1,2) (1,5), (1,6)\} we form shortest extended-induced-cutset-tree, as shown below:

Shortest extended-induced-cutset-tree, $ET$
Thus, the Hamiltonian circuit contains edges:
\{(2, 3), (3, 4), (4, 6), (1, 2), (1, 5), (5, 6)\}
and the actual Hamiltonian circuit obtained by this algorithm is:

\[
\begin{align*}
2 & \rightarrow 1 \\
1 & \rightarrow 4 \\
4 & \rightarrow 3 \\
3 & \rightarrow 2 \\
2 & \rightarrow 1
\end{align*}
\]

1 → 2 → 3 → 4 → 6 → 5 → 1, giving total weight “13 Units”. This is actually the shortest Hamiltonian circuit in the given complete graph!

**Remark 2.4.2:** From the above algorithm 2.4.1, the problem of finding shortest weight Hamiltonian circuit thus reduces to finding certain shortest tree (using appropriate columns in the \(f\)-cutset matrix) in the newly constructed associated lattice graph as per this algorithm!

### 2.5 A New TSP Algorithm for Graphs satisfying Triangle Inequality:

For complete graphs with vertices as points representing cities on a realistic planar map and where the weights on the edges are real distances we propose the following criterion for a Hamiltonian tour to be optimal. This criterion is motivated from appropriate extension of the idea of geodesic, i.e. the curve of shortest length joining two points on a surface that we determine for the surface under consideration using calculus of variation, e.g. by solving Euler-Lagrange equation one finds that straight line is geodesic for plane and for any two points on plane, the length of the straight line segment joining these two points is thus the shortest distance.

Now, suppose we are given three non-collinear points on the plane say A, B, C. What is the smooth curve of shortest length (geodesic) passing through all these points? Let us join points A, B and then B, C by a straight line segments (and make smooth their join at B) and obtain smooth curve A-B-C. Similarly, join points A, C and then C, B by a straight line segments (and make smooth their join at C) and obtain smooth curve A-C-B. Now which one among these curves A-B-C and A-C-B is shortest? Move the tangent vector along both the curves and note down the magnitude of rotation (at B and C respectively) for both these curves. The curve for which the magnitude of the angle of rotation of tangent vector (at B or C) will be least will be the shortest one!

Given \(p\) distinct points on a plane, \(\{v_1, v_2, \cdots, v_p\}\) and suppose we construct all possible Hamiltonian paths between \(v_1\) to \(v_p\) (passing through all other points only once). Which one among these is shortest? The one for which the sum of the magnitudes of the angles of rotation of tangent vector (at each intermediate node) will be least will
be the shortest Hamiltonian path! Also, suppose we construct all possible Hamiltonian tours (passing through all points only once). Which one among these is shortest? **The one for which the sum of the magnitudes of the angles of rotation of tangent vector (at each intermediate node) will be least will be the shortest Hamiltonian tour!**

**Algorithm 2.5.1 (An Economical Hamiltonian Tour for Real TS passing through Real Cities):**
Let \( \{r_1, r_2, \ldots, r_i, \ldots, r_p\} \) be the position vectors of points representing cities on plane (with respect to some origin).

1. Find the following vector (center of mass),
   \[
   R = \frac{1}{p} \sum_{i=1}^{p} r_i
   \]
2. Find \( r = \max |R - r_i| \)
3. By taking the point representing vector \( R \) (center of mass) as center draw a circle with radius equal to \( r \) (clearly, all points represented by position vectors \( \{r_1, r_2, \ldots, r_i, \ldots, r_p\} \) will lie inside this circle).
4. Draw radii to this circle passing through each point representing a position vector \( \{r_1, r_2, \ldots, r_i, \ldots, r_p\} \).
5. Name the end points of these radii on the circle by \( r_i \) if the radius is passing through \( r_i \), for all \( i = 1, 2, \ldots, p \).
6. Form the sequence of points by starting with some end point of some radius and moving to the next end point of radius in succession anticlockwise (or, clockwise) till you reach to the starting point. Let it be \( r_{i_1}' \rightarrow r_{i_2}' \rightarrow \cdots \rightarrow r_{i_1}' \)
7. Join (by vectors) and construct Hamiltonian circuit by replacing \( r_{i_j}' \) by \( r_{i_j} \) in the above sequence to form \( r_{i_1} \rightarrow r_{i_2} \rightarrow \cdots \rightarrow r_{i_1} \).

**Remark 2.5.1:** The Hamiltonian circuit thus obtained will have very much close to the minimum the sum of the magnitudes of the angles of rotations of the tangent vector going around this Hamiltonian circuit.
Acknowledgements

I am very much thankful to Mr. Riko Winterle and Dr. M. R. Modak for initiating this revision.

Reference

1. Deo Narsingh, Graph Theory with applications to engineering and computer science, Prentice-Hall of India Private Limited, New Delhi-110001, 1989.