Hamiltonian Cycles and Travelling Salesfolk

Alberto Gomez Gomez [1]

[1]Unaffiliated; Email: agg29@ou.ac.uk

Abstract
A method is given in this paper that makes it easier to solve both the Hamiltonian cycle problem and the travelling salesman problem in any number of space dimensions and in both their directed and undirected varieties.

Keywords: NP-completeness, Hamiltonian cycle problem, travelling salesman problem, maximum matching, perfect matching.

MSC Codes: *Primary* – 05C45; *Secondary* – 05C85, 05C05, 68R10, 68Q17

1. Introduction

In Mathematics and Computer Science, there is a set of problems for which no **efficient algorithm** has yet been found (that is, no algorithm is known that solves any of them in polynomial time). These problems are often called **non-deterministic polynomial-time complete problems** (or simply, **NP-complete problems**). Among these, one of the most notorious is the **travelling salesman problem** (TSP), which tries to find the 'best' path visiting each vertex in a graph just once and returning to where it started. Here, the word 'best' means 'shortest', 'cheapest', 'fastest', 'most scenic', 'most desirable', or whatever fits the context. In this paper, we shall often use the word 'lightest' in this sense. A series of algorithms (including heuristic ones, the branch-and-bound, nearest neighbour, and Christofides') are already available that give a solution, but this solution is not guaranteed to be an 'optimal' one. So the problem remains unsolved.

Another NP-complete problem is that of deciding whether a graph is Hamiltonian. That is, deciding whether it contains at least one **Hamiltonian cycle** (that is, a cycle visiting each vertex in the graph exactly once and ending where it started). These two problems are somehow related, meaning that finding the solution to one of them is likely to quickly lead to the solution of the other (as this is, in fact, a defining characteristic of all NP-complete problems).

For starters, let us imagine a set of towns linked by roads of known lengths. Can we find the shortest Hamiltonian cycle though them? One way of finding the answer is by using the **exhaustion algorithm**, which compares all the possibilities and picks the shortest one. However, when the number of towns is large, the difficulty of checking each possibility increases exponentially and eventually reaches a point where the problem is intractable even for the most powerful computers available (as of this writing). Because of this *combinatorial explosion*, the only known algorithm that guarantees a solution (namely, the exhaustion algorithm) is rendered useless when it comes to handling large graphs. In the present paper, we shall see whether we can improve upon the present state of affairs, but not before we have first dealt with the second of the above-mentioned problems. Namely, that of deciding whether a graph is Hamiltonian.

2. The Hamiltonian Cycle Problem

So far, we know that certain graphs are always Hamiltonian. These include the **complete graphs**, denoted by K_n, where n is the number of vertices in the graph. In these graphs, each vertex is linked to each other by exactly one edge, so it is perfectly possible to travel from any vertex to any other through just one edge (namely, the one between them). The number of (undirected) edges in a complete graph is

$$m_{\text{sym}} = \tfrac{1}{2}n(n-1), \tag{1}$$

Licensed Under Creative Commons Attribution CC BY

and the number of (undirected) Hamiltonian cycles that can be found in this graph is

$$h_{\text{sym}} = \frac{1}{2}(n-1)! \tag{2}$$

As we can see, the latter number becomes very big very soon as n grows.

A version of this problem considers the possibility that the Hamiltonian cycles in the graph have different weights depending on which direction they are traversed. This happens in **asymmetric graphs** (that is, those where at least one edge weighs differently depending on which direction it is traversed). In such graphs, each edge may be thought of as being made up of two oppositely directed and not necessarily equally weighted **arcs**. In these graphs, the number of arcs is effectively twice that in Equation 1 and the number of distinct Hamiltonian cycles is twice that in Equation 2. That is,

$$m_{\text{asym}} = n(n-1), \tag{3}$$

and

$$h_{\text{asym}} = (n-1)! \tag{4}$$

At first, one might think that solving NP problems involving asymmetric graphs is harder than solving NP problems involving symmetric graphs, but this is not at all the case. In order to see why, we may first consider another type of graphs that are known to be Hamiltonian too. Namely, the **cycle graphs**. That is, those consisting of a single cycle of vertices and edges. These graphs are denoted by C_n (where n is as before) and have as many edges as they have vertices (that is, $m = n$). Clearly, no graph having fewer than n edges can ever be Hamiltonian. So C_n is the smallest simple graph where a Hamiltonian cycle can be found, and K_n is the largest simple graph where the largest number of Hamiltonian cycles can be found (and this number is given by either Equation 2 or Equation 4, depending on whether the graph is symmetric or asymmetric).[1]

Example 1. As an example, let us consider a graph with four vertices A, B, C, D, representing four towns. Let us imagine that only four roads can be built between these towns, and that these roads can only be travelled in one direction. There are many ways to arrange these one-way roads, but not all of these arrangements are Hamiltonian. If we are to have one such arrangement, then the roads must allow travellers to visit each town exactly once and return to the starting point. Let us consider one such route. Namely, $ABCDA$. Its corresponding graph is as in Figure 1.

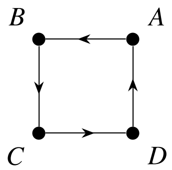


Figure 1: A cycle digraph with four vertices.

Figure 1 represents a directed cycle graph (also called a **cycle digraph**). By definition, this digraph contains just one Hamiltonian cycle (namely, $ABCDA$), which can only be travelled in one direction (because it is only made up of one-way roads).

[1]A **simple graph** is one with no **loops** (joining a vertex to itself) or **multiple edges** (joining the same pair of vertices).

Now, the adjacency matrix of this digraph is as follows.[2]

$$\mathbf{A} = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

One of the first things to strike the eye when looking at this matrix is that it is asymmetric everywhere (with respect to the main diagonal). The second is that the only Hamiltonian cycle there is arranged in such a way that its positive entries always occur on different rows and columns, and never symmetrically about the main diagonal. In fact, should two of these positive entries be symmetric about the main diagonal, they would form a cycle of length 2 (also called a 2-cycle, or **transposition**), which cannot be part of a Hamiltonian cycle (unless, of course, the graph has only two vertices).

The two conditions pointed out in Example 1 are necessary, but not enough to define a (directed) Hamiltonian cycle. The reason is that, in larger graphs, lesser cycles of length more than 2 are often found which must be winnowed out of the true Hamiltonian cycles in these graphs and can be more difficult to spot (by simple inspection of the matrix) than simple transpositions are, because they come in many shapes and sizes, as the following example shows for a not too large graph.

Example 2. Let there be a graph with six vertices (or towns) A, B, C, D, E, F and six arcs (or one-way roads) between them arranged in such a way that the corresponding adjacency matrix is as follows.

$$\mathbf{A} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

Here, the two conditions mentioned in Example 1 are both satisfied (because the number 1 occurs exactly once in each row and column, and never symmetrically about the main diagonal). However, this graph is non-Hamiltonian, because it has two lesser cycles of length 3 (namely, $ABCA$ and $DEFD$), rather than just one cycle through all vertices.

[2] The **adjacency matrix** of a digraph with n vertices is the $n \times n$ matrix in which the entry in row i and column j is the number of arcs from vertex i to vertex j.

Licensed Under Creative Commons Attribution CC BY

As we have seen in Example 2, large simple graphs may contain lesser cycles whose lengths can go from 2 to $n - 2$ (where n is the number of vertices in the graph), and the only lesser cycles that are easy to spot (by simple inspection of the adjacency matrix) are transpositions (because they are symmetric about the main diagonal).

Fortunately, there is one way of telling whether the above digraphs are Hamiltonian or not, and it is as simple as telling whether they are strongly connected or not. (A digraph is **strongly connected** if there is a path from any vertex to any other.) The reason is that graphs (such as that in Example 2) which have lesser cycles and exactly one positive entry in each row and column are always disconnected, because each vertex has only one in-degree and only one out-degree, and these are both employed in making up the lesser cycles, so they cannot be used for linking these lesser cycles together. So, an obviously necessary (but not sufficient) condition for a digraph to be Hamiltonian is that it is strongly connected.

We can know whether a digraph is strongly connected by first finding the matrix

$$\mathbf{B} = \mathbf{A} + \mathbf{A}^2 + \ldots + \mathbf{A}^{n-1} = \sum_{i=1}^{n-1} \mathbf{A}^i, \tag{5}$$

where n is the number of vertices in the graph and $\mathbf{A}$ is the adjacency matrix, and then seeing whether all the non-diagonal entries of B are positive. If this is the case, the digraph fulfils one of the conditions to be Hamiltonian. Otherwise, the digraph is non-Hamiltonian.

In the case of Example 1, the matrix $\mathbf{B}$ is as follows.

$$\mathbf{B} = \mathbf{A} + \mathbf{A}^2 + \mathbf{A}^3 = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \end{array}.$$

This matrix is of the required form (because each of its non-diagonal entries is positive). Hence, the adjacency matrix $\mathbf{A}$ in Example 1 belongs to a digraph that might be Hamiltonian. In fact, it is Hamiltonian, because it is a cycle digraph.

In the case of Example 2, the matrix $\mathbf{B}$ is as follows.

$$\mathbf{B} = \mathbf{A} + \mathbf{A}^2 + \mathbf{A}^3 + \mathbf{A}^4 + \mathbf{A}^5 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 1 & 2 & 2 & 0 & 0 & 0 \\ 2 & 1 & 2 & 0 & 0 & 0 \\ 2 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 2 \\ 0 & 0 & 0 & 2 & 1 & 2 \\ 0 & 0 & 0 & 2 & 2 & 1 \end{bmatrix} \end{array}.$$

This matrix is not of the required form (because some of its non-diagonal entries are noughts). Hence, the adjacency matrix $\mathbf{A}$ in Example 2 belongs to a non-Hamiltonian digraph. Now we are in a position to tell whether a digraph is Hamiltonian.

Theorem 1. *A digraph is Hamiltonian if and only if both of the following conditions hold.*

1. *Its adjacency matrix has a set of ones covering each row and column exactly once.*
2. *The subdigraph corresponding to this set of ones is strongly connected.*

Corollary 1.1. *The entries corresponding to a Hamiltonian cycle in the adjacency matrix of a Hamiltonian graph occur exactly once in each row and column.*

Proof of Theorem 1. Let us assume that there is a Hamiltonian cycle made of lesser cycles. Then, some of the vertices in these lesser cycles would either have more than one in-degree or more than one out-degree. Otherwise, they could not be joined together to form a greater, Hamiltonian cycle. However, by definition, the vertices in a (directed) cycle have exactly one in-degree and exactly one out-degree. But this is a contradiction. Hence, a Hamiltonian cycle cannot be made up of lesser cycles.

Also, if a digraph had a Hamiltonian cycle and either some of the rows or columns in the adjacency matrix of this digraph were entirely made of noughts or entries not in the cycle, then the corresponding vertices could not be joined, which is a contradiction, because a Hamiltonian cycle must go through all the vertices in the digraph. Hence, a Hamiltonian cycle visits each row and column of the graph's adjacency matrix exactly once.

Theorem 1 is easy to verify for small graphs (by inspection of the adjacency matrix). For bigger ones, it is convenient to label each entry in the adjacency matrix with subscripts indicating its position (that is, its row and column), as in the following example.

Example 3. Let the following be the adjacency matrix **A** of an asymmetric graph with 12 vertices $(A, B, \ldots, L)$ and 12 arcs.

$$\mathbf{A} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \end{array} \begin{array}{c} \begin{array}{cccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L \end{array} \\ \left[\begin{array}{cccccccccccc}
0_{1,1} & 1_{1,2} & 0_{1,3} & 0_{1,4} & 0_{1,5} & 0_{1,6} & 0_{1,7} & 0_{1,8} & 0_{1,9} & 0_{1,10} & 0_{1,11} & 0_{1,12} \\
0_{2,1} & 0_{2,2} & 0_{2,3} & 0_{2,4} & 0_{2,5} & 0_{2,6} & 0_{2,7} & 0_{2,8} & 0_{2,9} & 0_{2,10} & 1_{2,11} & 0_{2,12} \\
1_{3,1} & 0_{3,2} & 0_{3,3} & 0_{3,4} & 0_{3,5} & 0_{3,6} & 0_{3,7} & 0_{3,8} & 0_{3,9} & 0_{3,10} & 0_{3,11} & 0_{3,12} \\
1_{4,1} & 0_{4,2} & 0_{4,3} & 0_{4,4} & 0_{4,5} & 1_{4,6} & 0_{4,7} & 0_{4,8} & 0_{4,9} & 1_{4,10} & 1_{4,11} & 0_{4,12} \\
0_{5,1} & 0_{5,2} & 0_{5,3} & 1_{5,4} & 0_{5,5} & 0_{5,6} & 0_{5,7} & 0_{5,8} & 0_{5,9} & 0_{5,10} & 0_{5,11} & 0_{5,12} \\
0_{6,1} & 0_{6,2} & 0_{6,3} & 0_{6,4} & 0_{6,5} & 0_{6,6} & 0_{6,7} & 0_{6,8} & 0_{6,9} & 0_{6,10} & 1_{6,11} & 0_{6,12} \\
0_{7,1} & 0_{7,2} & 0_{7,3} & 0_{7,4} & 0_{7,5} & 0_{7,6} & 0_{7,7} & 0_{7,8} & 0_{7,9} & 0_{7,10} & 0_{7,11} & 0_{7,12} \\
0_{8,1} & 0_{8,2} & 0_{8,3} & 0_{8,4} & 0_{8,5} & 0_{8,6} & 0_{8,7} & 0_{8,8} & 0_{8,9} & 0_{8,10} & 0_{8,11} & 0_{8,12} \\
0_{9,1} & 0_{9,2} & 0_{9,3} & 0_{9,4} & 0_{9,5} & 0_{9,6} & 0_{9,7} & 0_{9,8} & 0_{9,9} & 0_{9,10} & 0_{9,11} & 0_{9,12} \\
0_{10,1} & 0_{10,2} & 0_{10,3} & 1_{10,4} & 0_{10,5} & 0_{10,6} & 1_{10,7} & 0_{10,8} & 0_{10,9} & 0_{10,10} & 0_{10,11} & 0_{10,12} \\
0_{11,1} & 0_{11,2} & 0_{11,3} & 0_{11,4} & 0_{11,5} & 0_{11,6} & 0_{11,7} & 0_{11,8} & 0_{11,9} & 0_{11,10} & 0_{11,11} & 0_{11,12} \\
0_{12,1} & 0_{12,2} & 0_{12,3} & 0_{12,4} & 0_{12,5} & 1_{12,6} & 0_{12,7} & 0_{12,8} & 0_{12,9} & 0_{12,10} & 0_{12,11} & 0_{12,12}
\end{array}\right] \end{array}$$

This adjacency matrix has 12 (randomly assigned) positive entries. We can see at once that this matrix corresponds to a non-Hamiltonian graph, because it violates the conditions of Theorem 1. For example, entries CA and DA are both positive, but lie in the same column, so they cannot belong to the same Hamiltonian cycle, and since there are only 12 positive entries, there can be no Hamiltonian cycle in this digraph. Also, entries DJ and JD are symmetric about the main diagonal, since their subscripts (10, 4 and 4, 10) are reversed, so they form a lesser 2-cycle. Again, this violates Condition 2 of Theorem 1. Also, several rows and columns in this matrix are entirely made of noughts, which, again, is not allowed by Condition 1 of Theorem 1.

Example 4. As a further example, the following adjacency matrix has been enlarged from the previous one by randomly adding a number of positive entries (by means of a pair of 12-sided dice) until all rows

Licensed Under Creative Commons Attribution CC BY

and columns had at least one positive entry. (The last entry to be added was that in row 11 and column 1.)

$$\mathbf{A} = \begin{array}{c|cccccccccccc|} & A & B & C & D & E & F & G & H & I & J & K & L \\ \hline A & 0_{1,1} & 1_{1,2} & 0_{1,3} & 0_{1,4} & 1_{1,5} & 0_{1,6} & 0_{1,7} & 0_{1,8} & 0_{1,9} & 0_{1,10} & 0_{1,11} & 0_{1,12} \\ B & 0_{2,1} & 0_{2,2} & 1_{2,3} & 0_{2,4} & 0_{2,5} & 0_{2,6} & 1_{2,7} & 0_{2,8} & 0_{2,9} & 0_{2,10} & 1_{2,11} & 0_{2,12} \\ C & 1_{3,1} & 0_{3,2} & 0_{3,3} & 0_{3,4} & 0_{3,5} & 0_{3,6} & 0_{3,7} & 0_{3,8} & 1_{3,9} & 1_{3,10} & 0_{3,11} & 0_{3,12} \\ D & 1_{4,1} & 0_{4,2} & 0_{4,3} & 0_{4,4} & 0_{4,5} & 1_{4,6} & 0_{4,7} & 0_{4,8} & 1_{4,9} & 1_{4,10} & 1_{4,11} & 0_{4,12} \\ E & 0_{5,1} & 1_{5,2} & 0_{5,3} & 1_{5,4} & 0_{5,5} & 1_{5,6} & 0_{5,7} & 1_{5,8} & 1_{5,9} & 0_{5,10} & 0_{5,11} & 0_{5,12} \\ F & 1_{6,1} & 0_{6,2} & 1_{6,3} & 1_{6,4} & 0_{6,5} & 0_{6,6} & 0_{6,7} & 1_{6,8} & 0_{6,9} & 0_{6,10} & 1_{6,11} & 1_{6,12} \\ G & 0_{7,1} & 0_{7,2} & 0_{7,3} & 0_{7,4} & 1_{7,5} & 1_{7,6} & 0_{7,7} & 0_{7,8} & 0_{7,9} & 0_{7,10} & 0_{7,11} & 1_{7,12} \\ H & 1_{8,1} & 0_{8,2} & 1_{8,3} & 0_{8,4} & 1_{8,5} & 0_{8,6} & 1_{8,7} & 0_{8,8} & 1_{8,9} & 0_{8,10} & 0_{8,11} & 0_{8,12} \\ I & 0_{9,1} & 0_{9,2} & 1_{9,3} & 0_{9,4} & 0_{9,5} & 1_{9,6} & 0_{9,7} & 0_{9,8} & 0_{9,9} & 1_{9,10} & 0_{9,11} & 1_{9,12} \\ J & 0_{10,1} & 0_{10,2} & 0_{10,3} & 1_{10,4} & 0_{10,5} & 1_{10,6} & 0_{10,7} & 0_{10,8} & 1_{10,9} & 0_{10,10} & 0_{10,11} & 1_{10,12} \\ K & 1_{11,1} & 0_{11,2} & 0_{11,3} & 0_{11,4} & 0_{11,5} & 0_{11,6} & 0_{11,7} & 0_{11,8} & 0_{11,9} & 0_{11,10} & 0_{11,11} & 0_{11,12} \\ L & 0_{12,1} & 0_{12,2} & 1_{12,3} & 1_{12,4} & 0_{12,5} & 1_{12,6} & 0_{12,7} & 0_{12,8} & 0_{12,9} & 1_{12,10} & 0_{12,11} & 0_{12,12} \end{array}$$

In order to see whether this new graph is Hamiltonian, we start by listing all positive entries in its adjacency matrix, or rather, their subscripts, which are represented here as if they were two-dimensional coordinates (i, j) indicating the entry's row and column. Thus,

$(1,2)$, $(1,5)$, $(2,3)$, $(2,7)$, $(2,11)$, $(3,1)$, $(3,9)$, $(3,10)$, $(4,1)$, $(4,6)$, $(4,9)$, $(4,10)$, $(4,11)$, $(5,2)$, $(5,4)$, $(5,6)$, $(5,8)$, $(5,9)$, $(6,1)$, $(6,3)$, $(6,4)$, $(6,8)$, $(6,11)$, $(6,12)$, $(7,5)$, $(7,6)$, $(7,12)$, $(8,1)$, $(8,3)$, $(8,5)$, $(8,7)$, $(8,9)$, $(9,3)$, $(9,6)$, $(9,10)$, $(9,12)$, $(10,4)$, $(10,6)$, $(10,9)$, $(10,12)$, $(11,1)$, $(12,3)$, $(12,4)$, $(12,6)$, $(12,10)$.

In all, there are 45 positive entries in the above adjacency matrix, so the graph has 45 arcs, and some of these are transpositions. Namely,

$(3,9), (9,3)$; $(4,6), (6,4)$; $(4,10), (10,4)$; $(5,8), (8,5)$; $(6,12), (12,6)$; $(9,10), (10,9)$; $(10,12), (12,10)$.

It is important to note that none of these transpositions can form part of the same Hamiltonian cycle (by Condition 2 of Theorem 1).

We shall now arrange the above entries in a table where the numbers on the left-hand side of the vertical line are just row subscripts, and the numbers on the right-hand side of this line are the column subscripts of the positive entries in the given adjacency matrix. Thus,

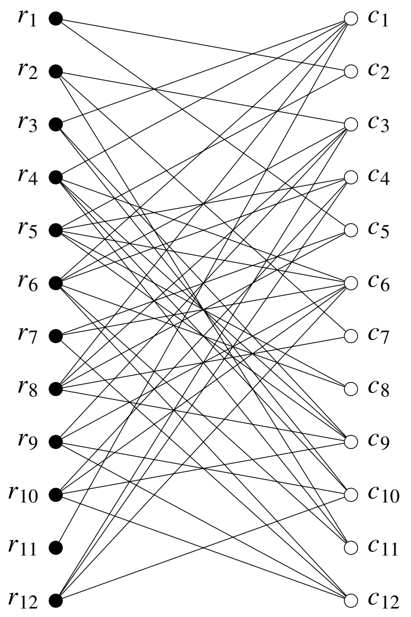
1	2	5				
2	3	7	11			
3	1	9	10			
4	1	6	9	10	11	
5	2	4	6	8	9	
6	1	3	4	8	11	12
7	5	6	12			
8	1	3	5	7	9	
9	3	6	10	12		
10	4	6	9	12		
11	1					
12	3	4	6	10		

At once, we see that the only entry in row 11 is that incident to column 1 of the matrix. So, if there is a Hamiltonian cycle in this graph, it must necessarily include this entry, which we have labelled $(11,1)$. This automatically excludes all other entries in column 1 of the matrix (by Condition 1 of Theorem 1). That is, all those of the form $(i,1)$. So we can now cross all the 1s on the right-hand side off the table. At this point, we may try to find a complete sequence of numbers from 1 to 12 in the table's columns either **heuristically** (that is, by trial and error) or by using the **maximum matching algorithm** (Edmonds, 1965). This algorithm ensures that a **maximum matching** (that is, one with the greatest possible number of edges) is achieved, and if this maximum matching is also a **perfect matching** (that is, one covering all the vertices in the graph), then (and only then) Condition 1 of Theorem 1 will hold. One such perfect matching (found heuristically by the author) is as follows.

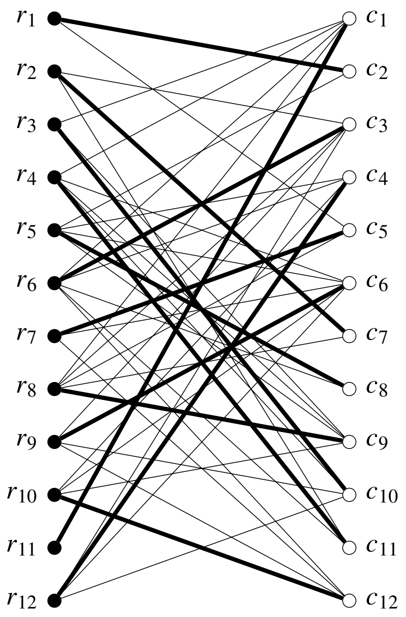
$$(1,2),\ (2,7),\ (7,5),\ (5,8),\ (8,9),\ (9,6),\ (6,3),\ (3,10),\ (10,12),\ (12,4),\ (4,11),\ (11,1).$$

This could also have been found by the maximum matching algorithm, whose starting point (in this case) is the bipartite graph in Figure 2a, whose black and white points represent, respectively, the rows and columns of the adjacency matrix for the original graph in the present example.[3]

Applying the maximum matching algorithm to the bipartite graph in Example 4, a perfect matching (coincidentally equal to the one above) is found that includes none of the above-listed transpositions. This matching is shown in Figure 2b and includes the arcs r_1c_2, r_2c_7, r_7c_5, r_5c_8, r_8c_9, r_9c_6, r_6c_3, r_3c_{10}, $r_{10}c_{12}$, $r_{12}c_4$, r_4c_{11}, $r_{11}c_1$. A cycle can be read off these arcs (namely, 1, 2, 7, 5, 8, 9, 6, 3, 10, 12, 4, 11, 1, or, using letters, $ABGEHIFCJLDKA$) which may (or may not) be Hamiltonian.



(a) Bipartite graph connecting the rows r_i and columns c_j of each positive entry in the adjacency matrix for the original graph in Example 4.



(b) A perfect matching (shown as thick lines) for the bipartite graph in Figure 2a.

Figure 2

So a perfect matching exists (namely, that in Figure 2b), and therefore, Condition 1 of Theorem 1 holds. Now, we need to check whether Condition 2 of Theorem 1 holds too. In order to see whether it

[3]A **bipartite graph** is one that can be split into two sets whose vertices are connected only to vertices in the other set.

Licensed Under Creative Commons Attribution CC BY

does, we write down the adjacency matrix corresponding to this matching. Namely,

$$\mathbf{A} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \end{array} \!\!\! \begin{array}{c} \begin{array}{cccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L \end{array} \\ \left[\begin{array}{cccccccccccc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \end{array}.$$

Then (applying Equation 5), we find that the matrix **B** is as follows.

$$\mathbf{B} = \sum_{i=1}^{11} \mathbf{A}^i = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \end{array} \!\!\! \begin{array}{c} \begin{array}{cccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L \end{array} \\ \left[\begin{array}{cccccccccccc} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{array}\right] \end{array}.$$

Clearly, all non-diagonal entries in **B** are positive. Hence, the digraph is strongly connected, and so, Condition 2 of Theorem 1 holds. So both conditions hold, and thus we know that the perfect matching in Figure 2b corresponds to a Hamiltonian cycle. Hence, the digraph in this example is Hamiltonian. (See Figure 3.)

Note that the perfect matching in Figure 2b may not be the only one, and so there may be more Hamiltonian cycles to be found in Figure 3, but the presence of just this one is enough to prove that the digraph in Figure 3 is Hamiltonian.

Note also that not until the $r_{11}c_1$ arc was added (and recall that this was the last arc to be randomly added to the digraph in Example 3 to form the digraph in the present example) did the digraph in the present example have any chance of being Hamiltonian.

Finally, note also that had the perfect matching in Figure 2b not been strongly connected, then other perfect matchings should have been checked for connectivity until the eventual exhaustion of all possibilities, or, alternatively, until Ore's Theorem is satisfied, whichever solves the problem first.

Example 4 ended up pointing out a difficulty that may arise when looking for Hamiltonian cycles in a graph. Namely, the possibility that the perfect matching we come up with (if any) is not strongly connected. In this case, we will have to seek a different perfect matching to check for strong connectivity, and here we enter an enumeration problem that is considered to be #P-complete. Namely, that of listing all perfect matchings in a bipartite graph. To this we shall devote the next section, but not before concluding the present one with an algorithm (suitable to be implemented as a computer program) that can solve (in

Licensed Under Creative Commons Attribution CC BY

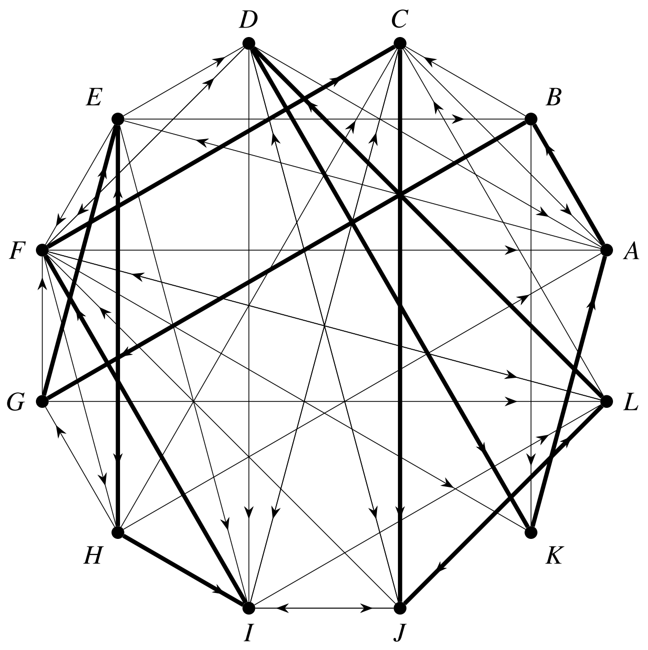


Figure 3: A digraph with (at least) one Hamiltonian cycle (shown as thick lines).

polynomial time) larger problems than those in the above examples, but not so large as to reach the worst case scenario, because long before this point is reached, the graph will be known to be Hamiltonian by Ore's Theorem. (See Algorithm 1.)

3. Perfect matching enumeration in bipartite graphs

It is easy to tell whether a bipartite graph has a perfect matching (by the maximum matching algorithm). However, once one is found, how can we know whether the graph has more, and if so, which ones? To answer these questions, we may use a **perfect matching enumeration algorithm**, such as Murty's (1968), Fukuda and Matsui's (1994) or Uno's (2001), which are based on the fact that there is a perfect matching for every distinct alternating cycle in a bipartite graph (Uno, 1997:2), where an **alternating cycle** is one that starts at a vertex in a given bipartite graph and returns to this same vertex after traversing the arcs between this and some other vertices in an alternating fashion (that is, an arc *in* the matching, then an arc *not in* the matching, and so on).

Now, the number of distinct perfect matchings in complete bipartite graphs (denoted by $K_{n,n}$) with $2n$ vertices is $n!$. So, the cited algorithms require super-polynomial time to enumerate all of these. Fortunately, the problems we are dealing with in this paper are never this big, because the main diagonal is always empty (since the graphs considered do not include loops). Hence, the graphs we are dealing with here are, at most, regular bipartite graphs of degree $n-1$, and the number of distinct perfect matchings in such graphs is exactly given by

$$p = 2(n-1)! - (n-1), \qquad (6)$$

with $(n-1)!$ of these perfect matchings being Hamiltonian (by Equation 4).

But the worst case scenario we may ever encounter is yet smaller than this, because if we keep randomly adding arcs to a bipartite graph (as we did in Example 4), the worst case scenario we may ever end up with is one in which all the vertices in the first set have degree $n-1$, but one, which would have degree 1. That is, after casting the dice a number of times, one vertex may never be lucky enough

Licensed Under Creative Commons Attribution CC BY

Algorithm 1 An algorithm for solving the Hamiltonian cycle problem

```
START   with either a directed or undirected graph with $m$ edges and $n$ vertices.
STEP 1  Check whether this graph satisfies Ore's Theorem.
            If so, STOP. The graph is Hamiltonian.
            If not, go to STEP 2.
STEP 2  Build the adjacency matrix $\mathbf{A}$ of the given graph.
STEP 3  Is any row or column in the graph entirely made of noughts?
            If so, STOP. The graph is non-Hamiltonian.
            If not, go to STEP 4.
STEP 4  Label each positive entry in $\mathbf{A}$ with subscripts $(i, j)$ indicating its row and column.
STEP 5  Build a table of labels with those indicating the rows on the left-hand side, and those indicating
        the columns they are incident to on the right-hand side of a vertical line.
STEP 6  Find a maximum matching either in the table of labels or its corresponding bipartite graph
        (either by the maximum matching algorithm or otherwise).
STEP 7  Is this maximum matching a perfect matching?
            If so, go to STEP 8.
            If not, STOP. The graph is non-Hamiltonian.
STEP 8  Find all the perfect matchings in the bipartite graph mentioned in STEP 6 (by a perfect matching
        enumeration algorithm) and write down the adjacency matrix of each of these matchings.
STEP 9  Is any of these adjacency matrices strongly connected?
            If so, STOP. The graph is Hamiltonian.
            If not, STOP. The graph is non-Hamiltonian.
```

to be matched except at the very last possible step, when all other vertices have already been assigned the greatest possible number of arcs. The graph cannot be Hamiltonian until this particular vertex is assigned an arc, but then, when it is, a number of perfect matchings will suddenly arise in the graph—all including this particular arc—and this number is exactly given by

$$p = \frac{2(n-1)! - (n-1)}{n-1} = 2(n-2)! - 1, \tag{7}$$

with $(n-2)!$ of these perfect matchings being Hamiltonian.

The chances of such an event ever happening are very small. So, in most cases, the problem of enumerating all the perfect matchings in a bipartite graph can be solved in polynomial time (because the graphs we are dealing with are highly trimmed, and their adjacency matrices, sparse), and this can be done either by any of the cited methods or by building a special tree like the one in the following example.

Example 5. How many perfect matchings can be found in a bipartite graph such as that in Figure 2a? In order to answer this question, we take the perfect matching in Figure 2b and construct a tree whose root can be any vertex (of small degree) in the graph, say r_1, and whose first branches are the arc incident to this vertex and in the matching (on the right-hand side) and the arc (or arcs) incident from this vertex and not in the matching (on the left-hand side), and whose other vertices follow suit in a anticlockwise direction. It is to be noted that whenever we climb up or down this tree, we encounter an alternating sequence of arcs in, and not in the matching. We do this systematically, repeating vertices if needed, until the leaves on the left-hand side can all be linked (from below) to the leaves on the right-hand side, except, of course, those that have degree 1 in the original graph. (See Figure 4.)

Then, climbing up the branches on the right-hand side, and down the branches on the left-hand side of the tree in Figure 4, we write down any sequence that starts at a particular vertex and ends at the next occurrence of the same vertex (if any). The list of distinct sequences thus identified is the required list of

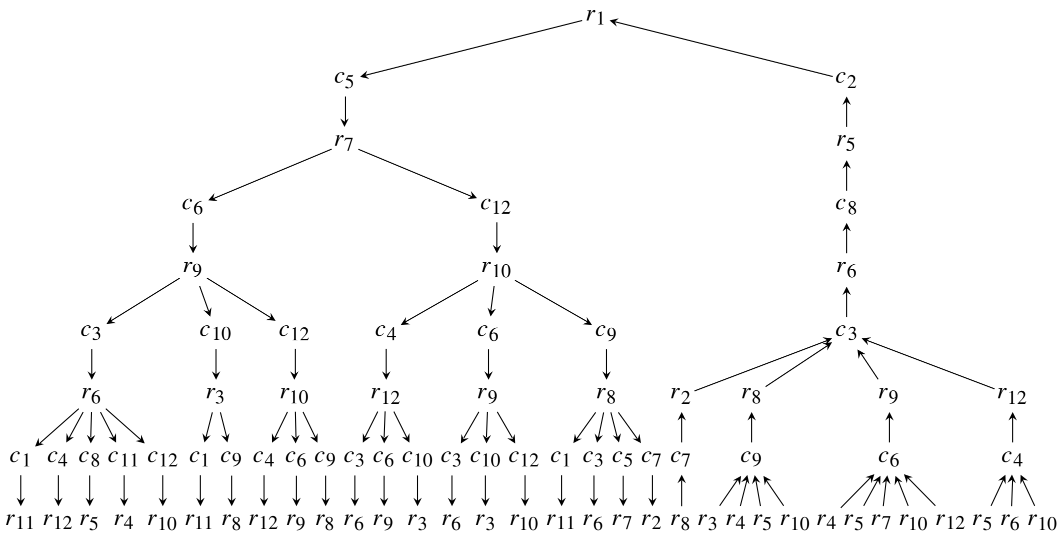

r_1
c_5 c_2
r_7 r_5
c_6 c_{12} c_8
r_9 r_{10} r_6
c_3 c_{10} c_{12} c_4 c_6 c_9 c_3
r_6 r_3 r_{10} r_{12} r_9 r_8 r_2 r_8 r_9 r_{12}
c_1 c_4 c_8 c_{11} c_{12} c_1 c_9 c_4 c_6 c_9 c_3 c_6 c_{10} c_3 c_{10} c_{12} c_1 c_3 c_5 c_7 c_7 c_9 c_6 c_4
r_{11} r_{12} r_5 r_4 r_{10} r_{11} r_8 r_{12} r_9 r_8 r_6 r_9 r_3 r_6 r_3 r_{10} r_{11} r_6 r_7 r_2 r_8 r_3 r_4 r_5 r_{10} r_4 r_5 r_7 r_{10} r_{12} r_5 r_6 r_{10}

Figure 4: A directed alternating tree for the graph in Figure 2b.

alternating cycles in the graph, each of which generates a different perfect matching other than that in Figure 2b. Table 1 lists all such cycles (starting with the vertices of lesser row index).

When checked for strong connectivity, some of the perfect matchings in Table 1 are found to be Hamiltonian, and others, non-Hamiltonian. Thus, we know that the graph in Figure 2a has a total of 16 Hamiltonian cycles. Namely, the 15 ones in Table 1 plus the one in Figure 2b.

It is important to note that the time complexity of a depth-first search for all repeated items in a rooted path graph (like that in Figure 4) is, at most, polynomial, which is exactly what we need to write an efficient algorithm for listing all perfect matchings in sparse bipartite graphs. (See Algorithm 2.)

4. The travelling salesman problem

The travelling salesman problem seeks to find the best path through all the vertices either in a symmetric or an asymmetric complete graph and then return to where it started. Finding the solution to this problem is easy now that we know to tell whether a graph is Hamiltonian (by means of Algorithm 1). One way to proceed is as follows.

4.1. The all-at-once approach

This procedure applies the Hungarian algorithm (Kuhn, 1955) to the weight (or cost) matrix of the given problem. We will first illustrate this procedure with an example, and then write up a general algorithm for it.

Example 6. Let there be a complete, asymmetric graph with 12 vertices ($A, B, ..., L$), whose table **A** of (randomly generated) arc weights is as follows. (A random number generator was used to construct this table, limiting its output to no more than two digits.)

Table 1: List of alternating cycles generated by the perfect matching in Figure 2b.

	alternating cycle	strongly connected	Hamiltonian cycle
1	$r_1c_5r_7c_6r_9c_3r_6c_8r_5c_2r_1$	yes	$AEBGFHICJLDKA$
2	$r_1c_5r_7c_6r_9c_{10}r_3c_9r_8c_3r_6c_8r_5c_2r_1$	yes	$AEBGFHCIJLDKA$
3	$r_1c_5r_7c_6r_9c_{10}r_3c_9r_8c_7r_2c_3r_6c_8r_5c_2r_1$	no	
4	$r_1c_5r_7c_6r_9c_{12}r_{10}c_4r_{12}c_3r_6c_8r_5c_2r_1$	yes	$AEBGFHCJILDKA$
5	$r_1c_5r_7c_6r_9c_{12}r_{10}c_9r_8c_3r_6c_8r_5c_2r_1$	yes	$AEBGFHCJILDKA$
6	$r_1c_5r_7c_6r_9c_{12}r_{10}c_9r_8c_7r_2c_3r_6c_8r_5c_2r_1$	no	
7	$r_1c_5r_7c_{12}r_{10}c_4r_{12}c_3r_6c_8r_5c_2r_1$	no	
8	$r_1c_5r_7c_{12}r_{10}c_4r_{12}c_6r_9c_3r_6c_8r_5c_2r_1$	yes	$AEBGLFHICJDKA$
9	$r_1c_5r_7c_{12}r_{10}c_4r_{12}c_{10}r_3c_9r_8c_3r_6c_8r_5c_2r_1$	no	
10	$r_1c_5r_7c_{12}r_{10}c_6r_9c_3r_6c_8r_5c_2r_1$	no	
11	$r_1c_5r_7c_{12}r_{10}c_6r_9c_{10}r_3c_9r_8c_3r_6c_8r_5c_2r_1$	no	
12	$r_1c_5r_7c_{12}r_{10}c_9r_8c_3r_6c_8r_5c_2r_1$	no	
13	$r_1c_5r_7c_{12}r_{10}c_9r_8c_7r_2c_3r_6c_8r_5c_2r_1$	yes	$AEBCJIFHGLDKA$
14	$r_4c_6r_9c_3r_6c_{11}r_4$	yes	$ABGEHICJLDFKA$
15	$r_4c_9r_8c_3r_6c_{11}r_4$	yes	$ABGEHCJLDIFKA$
16	$r_5c_4r_{12}c_3r_6c_8r_5$	no	
17	$r_5c_6r_9c_3r_6c_8r_5$	yes	$ABGEFHICJLDKA$
18	$r_5c_9r_8c_3r_6c_8r_5$	yes	$ABGEIFHCJLDKA$
19	$r_6c_4r_{12}c_3r_6$	no	
20	$r_6c_4r_{12}c_6r_9c_3r_6$	yes	$ABGEHICJLFDKA$
21	$r_6c_{12}r_{10}c_4r_{12}c_3r_6$	yes	$ABGEHIFLCJDKA$
22	$r_6c_{12}r_{10}c_6r_9c_3r_6$	no	
23	$r_6c_{12}r_{10}c_9r_8c_3r_6$	yes	$ABGEHCJIFLDKA$
24	$r_7c_{12}r_{10}c_9r_8c_5r_7$	no	
25	$r_9c_6r_{10}c_{12}r_9$	yes	$ABGEHIFCJLDKA$
26	$r_9c_{12}r_{10}c_4r_{12}c_6r_9$	yes	$ABGEHILFCJDKA$
27	$r_9c_{12}r_{10}c_6r_9$	no	

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}
r_1		86	34	71	39	97	38	67	41	20	94	36
r_2	1		61	89	91	33	50	22	51	39	23	44
r_3	90	40		40	90	27	52	55	77	21	94	6
r_4	60	84	19		19	45	71	36	12	37	27	13
r_5	67	59	36	12		82	9	86	75	94	30	44
r_6	66	93	49	16	87		5	52	56	25	87	10
r_7	77	29	66	87	74	85		35	22	49	46	28
r_8	48	81	6	22	8	3	79		25	47	85	81
r_9	56	92	42	79	56	19	70	83		16	49	43
r_{10}	60	64	11	41	77	27	68	29	61		32	37
r_{11}	80	15	67	18	11	22	45	34	80	7		41
r_{12}	50	90	64	32	78	11	13	95	45	44	9	

Licensed Under Creative Commons Attribution CC BY

Algorithm 2 An algorithm for listing all perfect matchings in sparse bipartite graphs

START with a perfect matching found (either by the maximum matching algorithm or otherwise) in either a directed or undirected bipartite graph with n vertices.

STEP 1 For each component of the graph (should there be many), construct a tree whose root can be any vertex in the graph (preferably one of low degree), and whose branches are, on one side, the arc incident to this vertex and in the matching, and on the other (or others), the arc (or arcs) incident from this vertex and not in the matching. All other vertices must follow suit in a strictly alternating path fashion, being repeated, if necessary, until the leaves on one side of the tree link up (from below) with those on the other side (or sides), with the exception (of course) of those vertices (if any) that are also leaves in the original graph.

STEP 2 Perform a depth-first search for repeating vertices, starting at any vertex and climbing up the branches on one side, and (if needed) down the branches on the other side (or sides) of the root.

STEP 3 Can a vertex be found twice in a depth-first search of the above-mentioned alternating tree?
If so, go to STEP 4.
If not, go to STEP 5.

STEP 4 Write down the sequence of vertices between and including this repeating vertex. This sequence is an alternating cycle, which (as a rule) must be written starting and ending with the smallest row index (so as to easily spot and avoid repetitions). Then, go back to STEP 3.

STEP 5 Make a list of all the different alternating cycles in the graph. Is this list empty?
If so, STOP. The given perfect matching is the only one in the graph.
If not, go to STEP 6.

STEP 6 For each alternating cycle in the latter list, add the arcs in the given perfect matching that are not in this cycle to form a new perfect matching.

STEP 7 Make a list of all the different perfect matchings thus found, adding the given perfect matching (that served to generate all others), and STOP. This is the required list.

In order to find the lightest Hamiltonian cycle in this matrix, we systematically apply the Hungarian algorithm to it. Thus, the first step is to label each row with, and reduce it by the lowest cost in the row. This gives the following cost matrix.

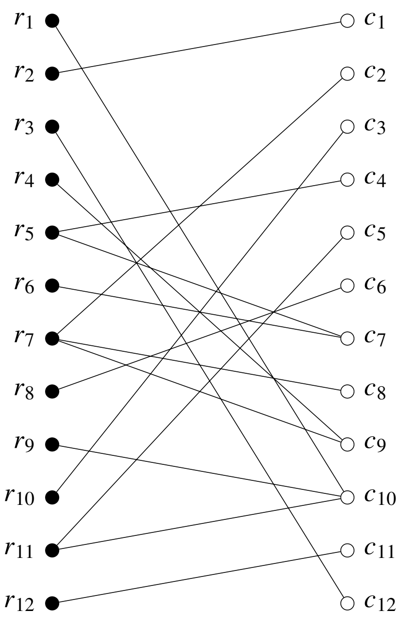
	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	
r_1		66	14	51	19	77	18	47	21	0	74	16	20
r_2	0		60	88	90	32	49	21	50	38	22	43	1
r_3	84	34		34	84	21	46	49	71	15	88	0	6
r_4	48	72	7		7	33	59	24	0	25	15	1	12
r_5	58	50	27	3		73	0	77	66	85	21	35	9
r_6	61	88	44	11	82		0	47	51	20	82	5	5
r_7	55	7	44	65	52	63		13	0	27	24	6	22
r_8	45	78	3	19	5	0	76		22	44	82	78	3
r_9	40	76	26	63	40	3	54	67		0	33	27	16
r_{10}	49	53	0	30	66	16	57	18	50		21	26	11
r_{11}	73	8	60	11	4	15	38	27	73	0		34	7
r_{12}	41	81	55	23	69	2	4	86	36	35	0		9

Then, we label each column with, and reduce it by the lowest cost in the column. Thus,

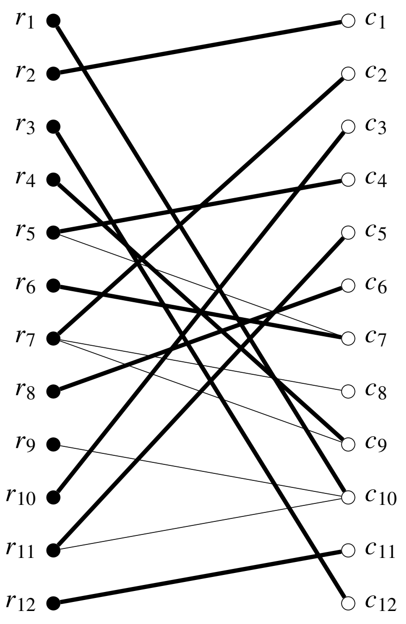
Licensed Under Creative Commons Attribution CC BY

$$
\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c|c}
 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} & \\
\hline
r_1 & & 59 & 14 & 48 & 15 & 77 & 18 & 34 & 21 & 0 & 74 & 16 & 20 \\
\hline
r_2 & 0 & & 60 & 85 & 86 & 32 & 49 & 8 & 50 & 38 & 22 & 43 & 1 \\
\hline
r_3 & 84 & 27 & & 31 & 80 & 21 & 46 & 36 & 71 & 15 & 88 & 0 & 6 \\
\hline
r_4 & 48 & 65 & 7 & & 3 & 33 & 59 & 11 & 0 & 25 & 15 & 1 & 12 \\
\hline
r_5 & 58 & 43 & 27 & 0 & & 73 & 0 & 64 & 66 & 85 & 21 & 35 & 9 \\
\hline
r_6 & 61 & 81 & 44 & 8 & 78 & & 0 & 34 & 51 & 20 & 82 & 5 & 5 \\
\hline
r_7 & 55 & 0 & 44 & 62 & 48 & 63 & & 0 & 0 & 27 & 24 & 6 & 22 \\
\hline
r_8 & 45 & 71 & 3 & 16 & 1 & 0 & 76 & & 22 & 44 & 82 & 78 & 3 \\
\hline
r_9 & 40 & 69 & 26 & 60 & 36 & 3 & 54 & 54 & & 0 & 33 & 27 & 16 \\
\hline
r_{10} & 49 & 46 & 0 & 27 & 62 & 16 & 57 & 5 & 50 & & 21 & 26 & 11 \\
\hline
r_{11} & 73 & 1 & 60 & 8 & 0 & 15 & 38 & 14 & 73 & 0 & & 34 & 7 \\
\hline
r_{12} & 41 & 74 & 55 & 20 & 65 & 2 & 4 & 73 & 36 & 35 & 0 & & 9 \\
\hline
 & 0 & 7 & 0 & 3 & 4 & 0 & 0 & 13 & 0 & 0 & 0 & 0 &
\end{array}
$$

This is the first revised cost matrix of the method. Its corresponding **partial graph** (which includes only the arcs whose current cost is nought) is shown in Figure 5a, and a maximum matching (found either by inspection or by the maximum matching algorithm) for the graph in Figure 5a is shown in Figure 5b.



(a) First partial graph (of the first revised cost matrix of the original graph).



(b) A maximum matching for the graph in Figure 5a.

Figure 5

Following the instructions of the maximum matching algorithm, the only alternating path to be found in Figure 5b is $r_9c_{10}r_1$. No breakthrough is achieved in Figure 5b, so we highlight rows 9 and 1 and column 10. Thus,

Licensed Under Creative Commons Attribution CC BY

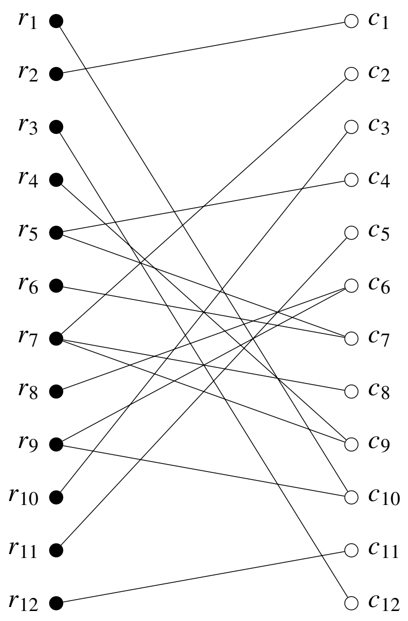
	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	
r_1		59	14	48	15	77	18	34	21	0	74	16	20
r_2	0		60	85	86	32	49	8	50	38	22	43	1
r_3	84	27		31	80	21	46	36	71	15	88	0	6
r_4	48	65	7		3	33	59	11	0	25	15	1	12
r_5	58	43	27	0		73	0	64	66	85	21	35	9
r_6	61	81	44	8	78		0	34	51	20	82	5	5
r_7	55	0	44	62	48	63		0	0	27	24	6	22
r_8	45	71	3	16	1	0	76		22	44	82	78	3
r_9	40	69	26	60	36	3	54	54		0	33	27	16
r_{10}	49	46	0	27	62	16	57	5	50		21	26	11
r_{11}	73	1	60	8	0	15	38	14	73	0		34	7
r_{12}	41	74	55	20	65	2	4	73	36	35	0		9
	0	7	0	3	4	0	0	13	0	0	0	0	

Following the instructions of the Hungarian algorithm, we have $\delta = 3$, so we reduce each value in the shaded rows by $\delta = 3$ and increase each value in the shaded column by the same amount $\delta = 3$, adjusting the labels accordingly. Thus,

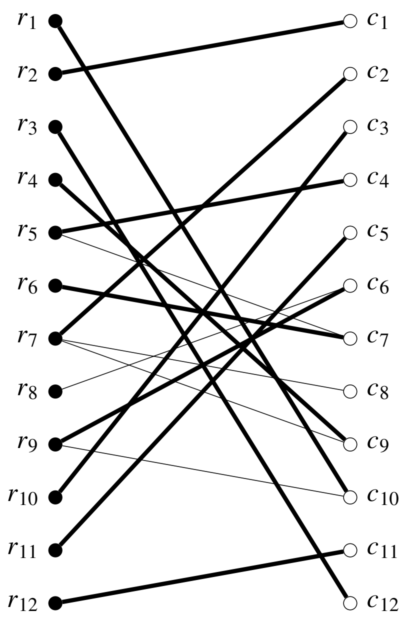
	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	
r_1		56	11	45	12	74	15	31	18	0	71	13	23
r_2	0		60	85	86	32	49	8	50	41	22	43	1
r_3	84	27		31	80	21	46	36	71	18	88	0	6
r_4	48	65	7		3	33	59	11	0	28	15	1	12
r_5	58	43	27	0		73	0	64	66	88	21	35	9
r_6	61	81	44	8	78		0	34	51	23	82	5	5
r_7	55	0	44	62	48	63		0	0	30	24	6	22
r_8	45	71	3	16	1	0	76		22	47	82	78	3
r_9	37	66	23	57	33	0	51	51		0	30	24	19
r_{10}	49	46	0	27	62	16	57	5	50		21	26	11
r_{11}	73	1	60	8	0	15	38	14	73	3		34	7
r_{12}	41	74	55	20	65	2	4	73	36	38	0		9
	0	7	0	3	4	0	0	13	0	−3	0	0	

This is the second revised cost matrix. Its corresponding partial graph is shown in Figure 6a, and a maximum matching for the graph in Figure 6a is shown in Figure 6b.

Figure 6b achieves no breakthrough, and the only alternating path there is $r_8c_6r_9c_{10}r_1$. So we revise the current cost matrix by first highlighting rows 1, 8, and 9 and columns 6 and 10. Thus,



(a) Second partial graph.



(b) A maximum matching for the graph in Figure 6a.

Figure 6

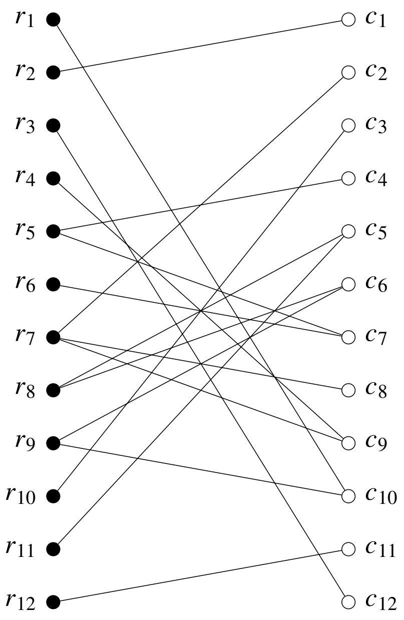
	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	
r_1		56	11	45	12	74	15	31	18	0	71	13	23
r_2	0		60	85	86	32	49	8	50	41	22	43	1
r_3	84	27		31	80	21	46	36	71	18	88	0	6
r_4	48	65	7		3	33	59	11	0	28	15	1	12
r_5	58	43	27	0		73	0	64	66	88	21	35	9
r_6	61	81	44	8	78		0	34	51	23	82	5	5
r_7	55	0	44	62	48	63		0	0	30	24	6	22
r_8	45	71	3	16	1	0	76		22	47	82	78	3
r_9	37	66	23	57	33	0	51	51		0	30	24	19
r_{10}	49	46	0	27	62	16	57	5	50		21	26	11
r_{11}	73	1	60	8	0	15	38	14	73	3		34	7
r_{12}	41	74	55	20	65	2	4	73	36	38	0		9
	0	7	0	3	4	0	0	13	0	−3	0	0	

Here, $\delta = 1$, so we reduce each shadowed row by $\delta = 1$ and increase each shadowed column by $\delta = 1$, adjusting the labels accordingly. Thus,

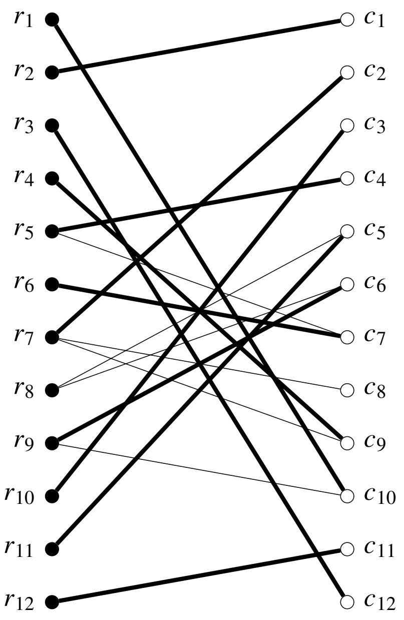
Licensed Under Creative Commons Attribution CC BY

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	
r_1		55	10	44	11	74	14	30	17	0	70	12	24
r_2	0		60	85	86	33	49	8	50	42	22	43	1
r_3	84	27		31	80	22	46	36	71	19	88	0	6
r_4	48	65	7		3	34	59	11	0	29	15	1	12
r_5	58	43	27	0		74	0	64	66	89	21	35	9
r_6	61	81	44	8	78		0	34	51	24	82	5	5
r_7	55	0	44	62	48	64		0	0	31	24	6	22
r_8	44	70	2	15	0	0	75		21	47	81	77	4
r_9	36	65	22	56	32	0	50	50		0	29	23	20
r_{10}	49	46	0	27	62	17	57	5	50		21	26	11
r_{11}	73	1	60	8	0	16	38	14	73	4		34	7
r_{12}	41	74	55	20	65	3	4	73	36	39	0		9
	0	7	0	3	4	−1	0	13	0	−4	0	0	

This is the third revised cost matrix. Its partial graph is shown in Figure 7a, and a maximum matching for the graph in Figure 7a is shown in Figure 7b.



(a) Third partial graph.



(b) A maximum matching for the graph in Figure 7a.

Figure 7

No breakthrough is achieved in Figure 7b. Two alternating paths there are $r_8c_5r_{11}$ and $r_8c_6r_9c_{10}r_1$.

Using both of them, we go on to highlight the corresponding rows and columns in the fourth revised cost matrix of the original graph. Namely, rows 1, 8, 9, and 11, and columns 5, 6, and 10. Thus,

Licensed Under Creative Commons Attribution CC BY

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	
r_1		55	10	44	11	74	14	30	17	0	70	12	24
r_2	0		60	85	86	33	49	8	50	42	22	43	1
r_3	84	27		31	80	22	46	36	71	19	88	0	6
r_4	48	65	7		3	34	59	11	0	29	15	1	12
r_5	58	43	27	0		74	0	64	66	89	21	35	9
r_6	61	81	44	8	78		0	34	51	24	82	5	5
r_7	55	0	44	62	48	64		0	0	31	24	6	22
r_8	44	70	2	15	0	0	75		21	47	81	77	4
r_9	36	65	22	56	32	0	50	50		0	29	23	20
r_{10}	49	46	0	27	62	17	57	5	50		21	26	11
r_{11}	73	1	60	8	0	16	38	14	73	4		34	7
r_{12}	41	74	55	20	65	3	4	73	36	39	0		9
	0	7	0	3	4	−1	0	13	0	−4	0	0	

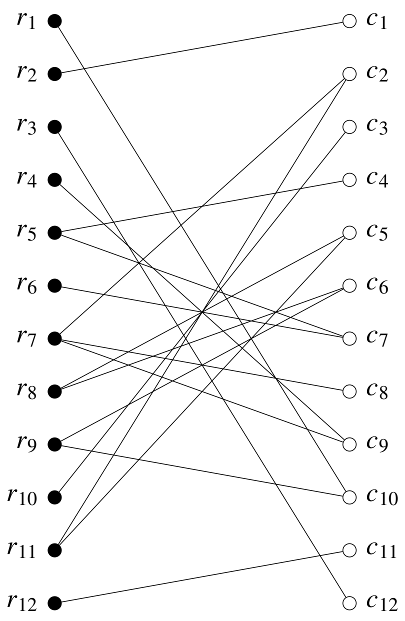
Here, $\delta = 1$. So, reducing each shaded row by $\delta = 1$, increasing each shaded column by $\delta = 1$, and adjusting the labels accordingly, we have

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	
r_1		54	9	43	11	74	13	29	16	0	69	11	25
r_2	0		60	85	87	34	49	8	50	43	22	43	1
r_3	84	27		31	81	23	46	36	71	20	88	0	6
r_4	48	65	7		4	35	59	11	0	30	15	1	12
r_5	58	43	27	0		75	0	64	66	90	21	35	9
r_6	61	81	44	8	79		0	34	51	25	82	5	5
r_7	55	0	44	62	49	65		0	0	32	24	6	22
r_8	43	69	1	14	0	0	74		20	47	80	76	5
r_9	35	64	21	55	32	0	49	49		0	28	22	21
r_{10}	49	46	0	27	63	18	57	5	50		21	26	11
r_{11}	72	0	59	7	0	16	37	13	72	4		33	8
r_{12}	41	74	55	20	66	4	4	73	36	40	0		9
	0	7	0	3	3	−2	0	13	0	−5	0	0	

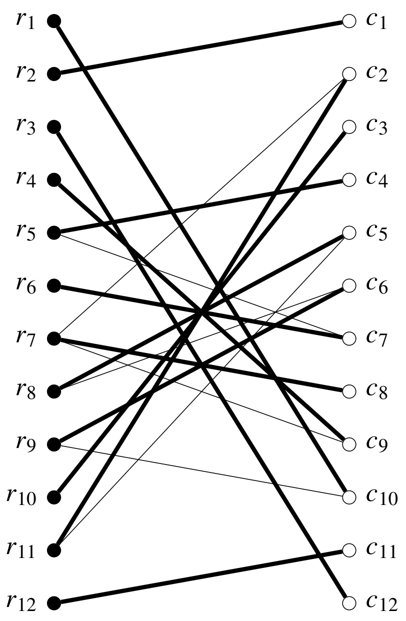
This is the fourth revised cost matrix. Its corresponding partial graph is shown in Figure 8a, and a maximum matching for the graph in Figure 8a is shown in Figure 8b.

At this point, we see that only one perfect matching can be found in Figure 8b (namely, r_1c_{10}, $r_{10}c_3$, r_3c_{12}, $r_{12}c_{11}$, $r_{11}c_2$, r_2c_1; r_4c_9, r_9c_6, r_6c_7, r_7c_8, r_8c_5, r_5c_4). It fulfils Condition 1, but not Condition 2 of Theorem 1 (as it is made of lesser cycles). So we reject this perfect matching (whose total weight, 153, is a lower bound for the problem in hand) and ask the Hungarian algorithm to find the next perfect matching. We do this by adding the lowest non-zero cost in the current matrix (namely, 1), which occurs in arcs r_8c_3 and r_4c_{12}. No alternative perfect matching can be found in the resulting bipartite graph. So we add the next lowest non-zero cost in the current matrix (namely, 4), which occurs in arcs r_4c_5, $r_{11}c_{10}$, $r_{12}c_6$, and $r_{12}c_7$. Still no alternative perfect matching can be found in the resulting bipartite graph. So we add the next lowest non-zero cost in the current matrix (namely, 5), which occurs in arcs r_6c_{12} and $r_{10}c_8$. (All of these steps are shown in the tables below.)

Licensed Under Creative Commons Attribution CC BY


r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9 r_{10} r_{11} r_{12}
c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8 c_9 c_{10} c_{11} c_{12}

(a) Fourth partial graph.


r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9 r_{10} r_{11} r_{12}
c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8 c_9 c_{10} c_{11} c_{12}

(b) A perfect matching for the graph in Figure 8a.

Figure 8

1	10		
2	1		
3	12		
4	9		
5	4	7	
6	7		
7	2	8	9
8	5	6	
9	6	10	
10	3		
11	2	5	
12	11		

1	10		
2	1		
3	12		
4	9	12	
5	4	7	
6	7		
7	2	8	9
8	5	6	3
9	6	10	
10	3		
11	2	5	
12	11		

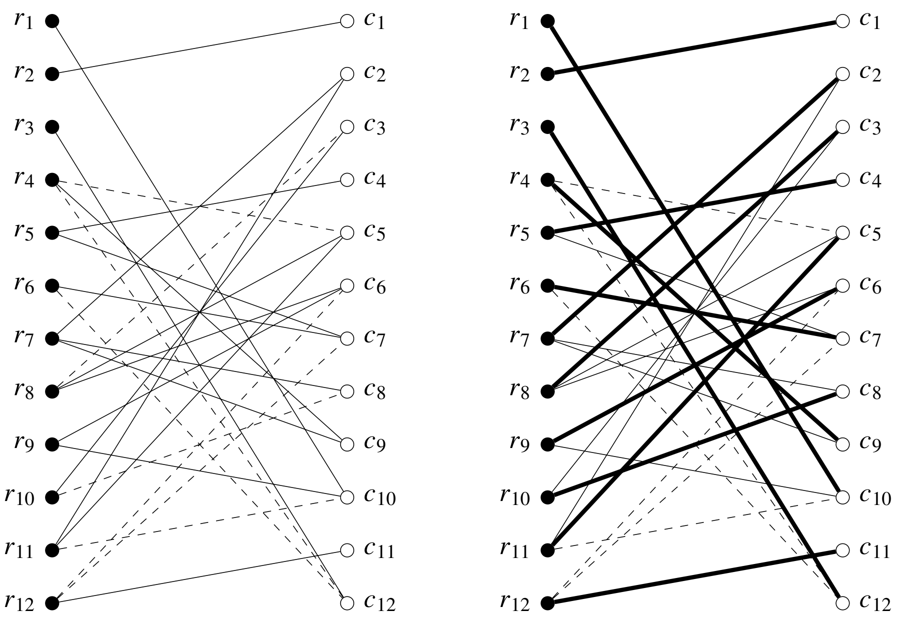
1	10		
2	1		
3	12		
4	9	12	5
5	4	7	
6	7		
7	2	8	9
8	5	6	3
9	6	10	
10	3		
11	2	5	10
12	11	6	7

1	10		
2	1		
3	12		
4	9	12	5
5	4	7	
6	7	12	
7	2	8	9
8	5	6	3
9	6	10	
10	3	8	
11	2	5	10
12	11	6	7

This time, two alternative perfect matchings can be found in the resulting graph (either by inspection or by a perfect matching enumeration algorithm), and only one of these fulfils all the conditions of Theorem 1. Namely, r_1c_{10}, $r_{10}c_8$, r_8c_3, r_3c_{12}, $r_{12}c_{11}$, $r_{11}c_5$, r_5c_4, r_4c_9, r_9c_6, r_6c_7, r_7c_2, r_2c_1, which can be written more concisely as 1, 10, 8, 3, 12, 11, 5, 4, 9, 6, 7, 2, 1, or, using letters, as $AJHCLKEDIFGBA$. (See Figure 9b.)

For ease of reference, the last revised cost matrix is repeated below with all noughts shaded in light grey, and all added arcs shaded in darker grey.

Licensed Under Creative Commons Attribution CC BY



(a) Expanded fourth partial graph (with added arcs shown as dashed lines).

(b) A perfect matching for the graph in Figure 9a.

Figure 9

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	
r_1		54	9	43	11	74	13	29	16	0	69	11	25
r_2	0		60	85	87	34	49	8	50	43	22	43	1
r_3	84	27		31	81	23	46	36	71	20	88	0	6
r_4	48	65	7		4	35	59	11	0	30	15	1	12
r_5	58	43	27	0		75	0	64	66	90	21	35	9
r_6	61	81	44	8	79		0	34	51	25	82	5	5
r_7	55	0	44	62	49	65		0	0	32	24	6	22
r_8	43	69	1	14	0	0	74		20	47	80	76	5
r_9	35	64	21	55	32	0	49	49		0	28	22	21
r_{10}	49	46	0	27	63	18	57	5	50		21	26	11
r_{11}	72	0	59	7	0	16	37	13	72	4		33	8
r_{12}	41	74	55	20	66	4	4	73	36	40	0		9
	0	7	0	3	3	−2	0	13	0	−5	0	0	

The total cost of this Hamiltonian cycle can be read off the original cost matrix as

$$w_{\min} = 20 + 29 + 6 + 6 + 9 + 11 + 12 + 12 + 19 + 5 + 29 + 1 = 159. \quad (8)$$

Licensed Under Creative Commons Attribution CC BY

Alternatively, this can also be read off the labels of the last revised cost matrix plus the non-zero costs (namely, 1 and 5) of the extra arcs used in the new matching (namely, r_8c_3 and $r_{10}c_8$). Thus,

$$\begin{aligned} w_{\min} &= 25+1+6+12+9+5+22+5+21+11+8+9 && \text{(row labels)} \\ &\quad +0+7+0+3+3-2+0+13+0-5+0+0 && \text{(column labels)} \\ &\quad +1+5 && \text{(added weights)} \\ &= 159. \end{aligned} \tag{9}$$

The present work allows us to be absolutely certain that no lighter Hamiltonian cycle can be found in the original graph other than that in Figure 9b, and no better solution exists other than that in Equations 8 and 9.

An algorithm for the travelling salesman problem is now formally stated. (See Algorithm 3.)

Algorithm 3 An algorithm for the travelling salesman problem

```
START   with a complete (either symmetric or asymmetric) graph.
STEP 1  Apply the Hungarian algorithm until a perfect matching is found.
STEP 2  Is this perfect matching strongly connected?
            If so, STOP. This is the solution (or one of the solutions) to the problem in hand.
            If not, go to STEP 3.
STEP 3  Can other perfect matchings be found (by means of a perfect matching enumeration algorithm
        or otherwise) in the current partial graph?
            If so, go to STEP 4 .
            If not, go to STEP 5.
STEP 4  Is any of these perfect matchings strongly connected?
            If so, find the lightest of these and STOP. This is the solution to the problem in hand.
            If not, go to STEP 5.
STEP 5  Add the next lightest, non-zero arc (or arcs) in the last cost matrix to the last partial graph.
STEP 6  Can any new perfect matchings be found (by means of a perfect matching enumeration algorithm
        or otherwise) in the expanded partial graph?
            If so, go to STEP 4.
            If not, go to STEP 5.
```

This algorithm can be adapted to find the 'heaviest' (or 'costliest') Hamiltonian cycle in a graph. The only thing to do is to change the sign of all (non-zero) weights in the matrix of the graph, so that all positive entries now become negative, and then, increase all entries by an arbitrary amount, so that all negative entries become positive again. What we have done here is to reverse the weights in the matrix, so that the light ones become heavy, and the heavy ones, light. Then, we simply apply Algorithm 3 to find the lightest Hamiltonian cycle in the reversed matrix, which is exactly the same as the heaviest Hamiltonian cycle in the original graph.

4.2. The arc-adding approach

There is another way of tackling the travelling salesman problem. This time, it is an approximation, for we cannot be sure that the solution it comes up with is the best one, though it often is. The advantage of this new method is that it is often faster than the approach in the previous subsection, even though certainty is sacrificed in the trade-off.

It starts by deleting all arcs in the original graph but the n lightest ones, and then it checks whether this trimmed graph is Hamiltonian or not (by means of Algorithm 1). If it turns out to be Hamiltonian, then this cycle is the solution to the problem in hand. If it turns out not to be Hamiltonian, then we start adding arcs in strict order of increasing weight, checking (for each newly added arc) whether the conditions of Theorem 1 are satisfied. When they are satisfied—and they will eventually be, then several promising candidates may arise all at once, of which we are to take only the ones to *repeat*, *omit*, and *reverse* no single number from 1 to n (for $n > 2$) in any (i, j) pair of subscript components. Then, we will choose the lightest of all those that pass the strong connectivity test. This may be the solution (or one of the solutions) to the problem in hand.

5. Conclusion

The travelling salesman problem is not restricted to just distances between points in a plane, or in space, or on a line. In fact, the number of space dimensions involved is irrelevant, as is the concept of *distance* itself, which is replaced with that of *weight* (or *cost*). To see why, imagine a set of points in three-dimensional space and think not of the distances between them, but of the cost of covering these distances. This cost may be different depending on the direction in which we travel. For example, more effort is required to travel from a town in a valley to a town on a mountain than the other way round, since we are going uphill in the first case, and downhill in the second. The distance is the same, but not so the cost. This means that asymmetric graphs are far more common in real life than symmetric ones, and both are equally easy to solve—the latter being just a special case of the former.

In this paper, we have seen how to do so. The methods employed involve no new algorithms, just a combination of existing ones. First, we found two conditions that are necessary and sufficient to define a Hamiltonian cycle, and used this information to find a way of checking whether a graph is Hamiltonian. Then, we found a way of finding the lightest Hamiltonian cycle in sparse Hamiltonian graphs, such as that obtained from deleting all arcs from the complete graph of a travelling salesman problem except the lightest among those that are strictly necessary for the graph to be Hamiltonian.

It is to be noted that the algorithms cited in this paper (namely, the maximum matching algorithm, the Hungarian algorithm, and, when applied to sparse bipartite graphs, the perfect matching enumeration algorithm) can all be applied in polynomial time, meaning that the algorithms we came up with (namely, Algorithms 1 to 3), which are based on these, can also be applied in polynomial time (when applied to sparse graphs).

The algorithms in this paper are both based on a theorem, which, for lack of a better name, we might call 'The Hycle Theorem', which is short for 'Hamiltonian Cycle Theorem'. This theorem requires the adjacency matrix of any Hamiltonian cycle in a graph to have exactly one positive entry in each row and column and to be strongly connected. Together, these requirements are necessary and sufficient to define a Hamiltonian cycle. In fact, the Hycle Theorem can be rephrased as saying that there is a Hamiltonian cycle for every strongly connected, perfect matching that can be found in a graph.

References

Edmonds, Jack (1965), 'Paths, trees, and flowers', *Canadian Journal of mathematics* **17**, 449–467.
Fukuda, Komei and Tomomi Matsui (1994), 'Finding all the perfect matchings in bipartite graphs', *Applied Mathematics Letters* **7**(1), 15–18.
Kuhn, Harold William (1955), 'The Hungarian method for the assignment problem', *Naval research logistics quarterly* **2**(1-2), 83–97.
Murty, Katta G (1968), 'An algorithm for ranking all the assignments in order of increasing cost', *Operations research* **16**(3), 682–687.
Uno, Takeaki (1997), Algorithms for enumerating all perfect, maximum and maximal matchings in bipartite graphs, *in* 'International Symposium on Algorithms and Computation', Springer, pp. 92–101.
Uno, Takeaki (2001), A fast algorithm for enumerating bipartite perfect matchings, *in* 'International Symposium on Algorithms and Computation', Springer, pp. 367–379.