# Undirected Hamiltonian Circuit

Shayan Daijavad

December 2024

## 1 Introduction

**Undirected Hamiltonian Circuit**
**Input:** A graph $G$
**Goal:** Yes if $G$ has a cycle which includes each vertex exactly once.

Undirected Hamiltonian Circuit is a graph theory problem involving finding a cycle in a graph that includes each vertex exactly once. It is named after William Rowan Hamilton, a 19th century mathematician who created the Icosian game. In the Icosian game, the goal is to find a Hamiltonian cycle on a dodecahedron. It's also one of the first problems to be proven as NP-complete, as done by Richard Karp [3].

Undirected Hamiltonian Circuit has many applications in a wide variety of fields. These include in routing problems, where we can represent nodes in the graph as locations on a map, such as cities, and the edges are weighted with the physical distance between those locations. In this case, the problem becomes the Traveling Salesperson problem, where we are trying to find a Hamiltonian Circuit of minimum weight on a weighted graph [4]. For example, think of a UPS truck that starts at the UPS facility and needs to deliver packages to a bunch of houses, and then return to the facility. It is of importance to the driver to visit the houses in the least amount of time possible, which is why a routing algorithm like TSP is so valuable.

UHC and TSP also have applications in DNA sequencing [1]. In this problems, we can represent each node as a subsequence of DNA, and a Hamiltonian Circuit represents a full sequence. Sequences are often read in short chunks, so being able to reproduce the original sequence from those chunks is an important problem.

For a famous special case of UHC, see the knight's tour problem [9]. In this problem, the initial setup is a chess board with a knight placed somewhere on it. The goal is to find a "tour", or a set of moves that has the knight visit every single square on the board exactly one time. Although TSP has no polynomial time optimal solution, the knight's tour problem has a linear-time optimal solution that runs in time relative to the size of the board [5].

# 2 NP Complete Proof

To prove that Undirected Hamiltonian Circuit is NP Complete, we must do two things: prove that it is in NP, and prove that it is NP hard. To prove that it is in NP, we simply need to prove that any solution to the problem can be checked in polynomial time. To prove that it is NP hard, we can reduce another NP complete problem to it.

## 2.1 Proving UHC is in NP

Assume the solution to the problem comes in the form of an ordered list of vertices, representative of the cycle. First, we must make sure that the solution is actually a cycle, which is trivially done in polynomial time by checking that the first and last vertex in the list are the same. Next, for every pair of adjacent vertices, we must check that there exists an edge in the graph for said vertices. This can be done in polynomial time by iterating over the list in pairs and checking if an edge with each pair exists in the set of all edges. Finally, we must also check that the list includes every vertex, and includes every vertex exactly once. We can also do this in polynomial time while iterating over the list of vertices by adding new vertices to a set, and if we encounter a vertex that is already in the set, returning false.

## 2.2 Proving UHC is NP Hard

To prove that UHC is NP Hard, we can reduce Directed Hamiltonian Circuit to it. There also exists a very elegant reduction of 3SAT to Directed Hamiltonian Circuit, which I won't cover in full detail, but will show a figure of and give some intuition of.

### 2.2.1 3SAT to Directed Hamiltonian Circuit

For every boolean variable, we create a path of $3m + 1$ nodes where $m$ is the number of clauses in the boolean formula. Set the direction of the edges in the path to face left if the variable was assigned as false, and right if it was signed as true. We connect the end nodes of each path to the end nodes of the next path, directed downward. We also create a start node and an end node. The start node is connected to the end nodes of the first path, and the end node is connected to the end nodes of the last path. The end node is also connected back to the start node.

We also create nodes for each clause. For each variable in the clause (of up to $j$ variables), we add an outgoing edge to the $3j + 1$ node in the path of that variable, and an incoming edge from the $3j$ node in the path. For each complement variable, do the same thing but flip the direction of the edges.

Given this setup, if a Directed Hamiltonian Circuit exists in the graph, then the assignment of the variables satisfies the formula.

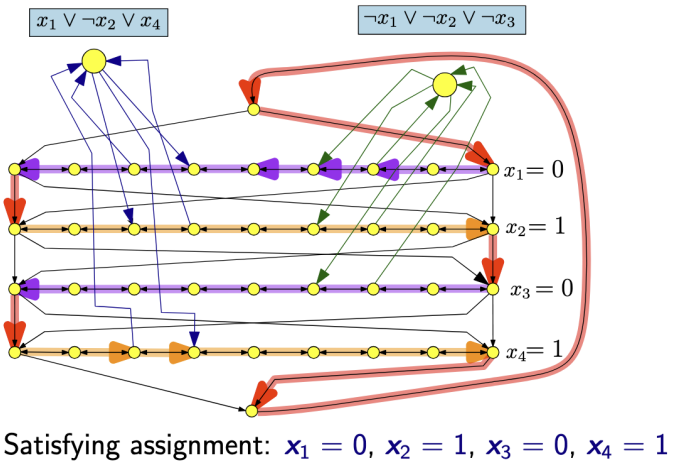


Figure 1: 3 SAT to DHC [6]

### 2.2.2 Directed Hamiltonian Circuit to Undirected Hamiltonian Circuit

The reduction of DHC to UHC is a good deal simpler than 3SAT to UHC. Since this is the primary reduction of the paper, here is its formal definition.

We define a pair of functions $f$ and $h$ as follows. Given an instance $I = H$ of DHC (where $H$ is a directed graph), $f$ will return the instance $f(I) = G$ of UHC. $f$ works as follows: For every vertex $v$ in $H$, replace $v$ with 3 new vertices: $v_{in}$, $v$, and $v_{out}$. For every directed edge $e$ in $H$ from vertex $u$ to vertex $v$, add an edge between $u_{out}$ and $v_{in}$. This is demonstrated in the figure below.

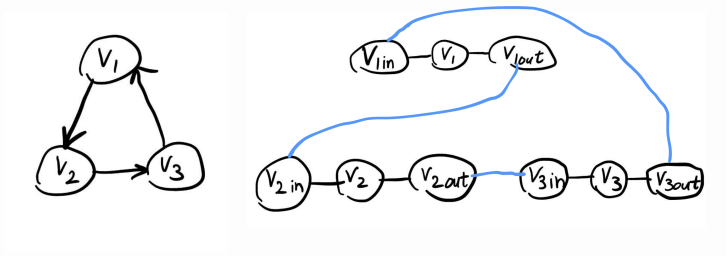


Figure 2: DHC to UHC

The intuition behind this is that we are basically forcing a direction for our circuit to take if it wants to visit every vertex in the graph. Figure 3 is an example of a directed graph without a Hamiltonian cycle, and how that property gets maintained in our UHC instance. As you can see, $v_1$ and $v_{1out}$ cannot be accessed without doubling back over them, which maintains the lack

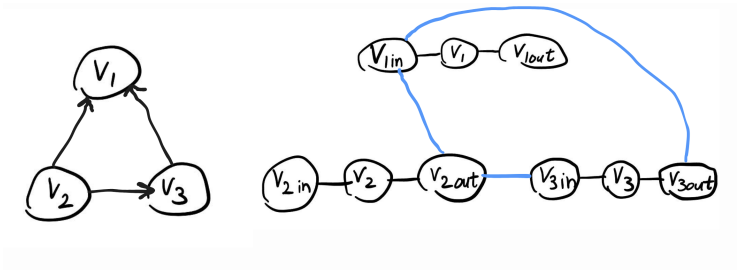


Figure 3: DHC to UHC 2

of Hamiltonian Cycle of the original graph.

Given a solution $T$ to the UHC instance $f(I)$, $h$ will return the solution $h(T) = S$ to DHC. $h$ is simple enough: if $f(I)$ returns True (as in, there is a Hamiltonian cycle), $h$ returns True, and False otherwise. If there's a Hamiltonian Cycle in one graph, there will be a Hamiltonian cycle in the other.

# 3 What can be done?

## 3.1 Approximation Algorithms

Since UHC returns a Yes or No answer, there are no true approximation algorithms for it. However, if we convert the problem into an optimization problem, such as finding the shortest length Hamiltonian circuit, our options for approximation algorithms expand greatly. To be a little more formal, what I am referring to is known as the Traveling Salesperson Problem. It is formally defined as follows:

**Traveling Salesperson Problem**
**Input:** A weighted graph $G$
**Goal:** A Hamiltonian circuit with minimum total weight

The best approximation algorithm for TSP is Christofides' Algorithm [2]. It is a 3/2 approximation; however, it has two conditions:

1. The input graph has to be complete.
2. All the edges of the input graph have to abide by the triangle inequality, i.e., $w(xy) + w(yz) \geq w(xz)$.

These may seem like strict conditions, but they are reasonable for problems taking place in Euclidean space. This is because real world physical distance naturally abides by the triangle inequality, and generally speaking there is at least one way to get between any two points in the real world.

Christofides' Algorithm works as follows [7]:

1. Create a minimum spanning tree $T$ of $G$.

2. Let $O$ be the set of vertices with odd degree in $T$.

3. Find a minimum-weight perfect matching $M$ in the subgraph induced in $G$ by $O$.

4. Combine the edges of $M$ and $T$ to form a connected multigraph $H$ in which each vertex has even degree.

5. Form an Eulerian circuit in $H$.

6. Make the circuit found in the previous step into a Hamiltonian circuit by skipping repeated vertices (shortcutting).

This algorithm has an $O(n^3)$ complexity, which is mainly because the current best algorithm for finding a minimum-weight perfect matching is $O(n^3)$.

The intuition behind the algorithm is that we want to build an Eulerian graph using a Minimum Spanning Tree as its foundation. This is because the Minimum Spanning Tree is a subset of the edges connecting all the vertices together with minimum weight, which would be an optimal solution if it was a circuit. The MST is where we get 2/2 of our 3/2 approximation.

Using the MST, we build an Eulerian graph, which only has the condition that every vertex must have even degree. If we can build such a graph, we are guaranteed to find an Eulerian circuit. An Eulerian circuit is a circuit that visits every edge exactly once, instead of every vertex.

To satisfy this condition, we make use of a very useful lemma called the handshaking lemma [8], which states that in every finite undirected graph, the number of vertices that touch an odd number of edges is even. By finding all the odd vertices and creating a minimum-weight perfect matching between them (which we can do since there are an even number of them), we eliminate all odd degree vertices. The minimum-weight perfect matching contributes another 1/2 to our approximation ratio.

With our Eulerian graph in hand, we can find an Eulerian circuit, and turn it into a Hamiltonian circuit by following the vertices in order and taking shortcuts to skip vertices we've already seen. We can do this because the graph is complete (meaning there is an edge between any two vertices), and we can also do so without affecting our approximation ratio because we know the weight of the shortcut edge is guaranteed to be less than or equal to the sum of the original edges from the Eulerian circuit thanks to the triangle inequality. For example, imagine our Eulerian circuit has the edges (0,1), (1,0), (0,4), and (4,0). Our Hamiltonian circuit will look like 0,1,4,0. We are replacing (1,0) and (0,4) with a shortcut edge, (1,4).

# References

[1] O. K. Berdewad, Laeequr Raheman, and V. A. Jadhav. A hamilton path-based approach to dna sequence analysis. *Tuijin Jishu/Journal of Propulsion Technology*, 44(4), 2023.

[2] Nicos Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. *Carnegie Mellon University*, 3:10, 03 2022.

[3] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

[4] E.L. Lawler. *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley-Interscience series in discrete mathematics and optimization. John Wiley & Sons, 1985.

[5] Shun-Shii Lin and Chung-Liang Wei. Optimal algorithms for constructing knight's tours on arbitrary n×m chessboards. *Discrete Applied Mathematics*, 146(3):219–232, 2005.

[6] University of Illinois Urbana-Champaign. Cs 374 lecture notes: 3sat to hamiltonian cycle reduction, 2021.

[7] Wikipedia contributors. Christofides algorithm — Wikipedia, the free encyclopedia, 2024. [Online; accessed 13-December-2024].

[8] Wikipedia contributors. Handshaking lemma — Wikipedia, the free encyclopedia, 2024. [Online; accessed 13-December-2024].

[9] Wikipedia contributors. Knight's tour — Wikipedia, the free encyclopedia, 2024. [Online; accessed 14-December-2024].