Metaheuristic approach to the Hamiltonian Path

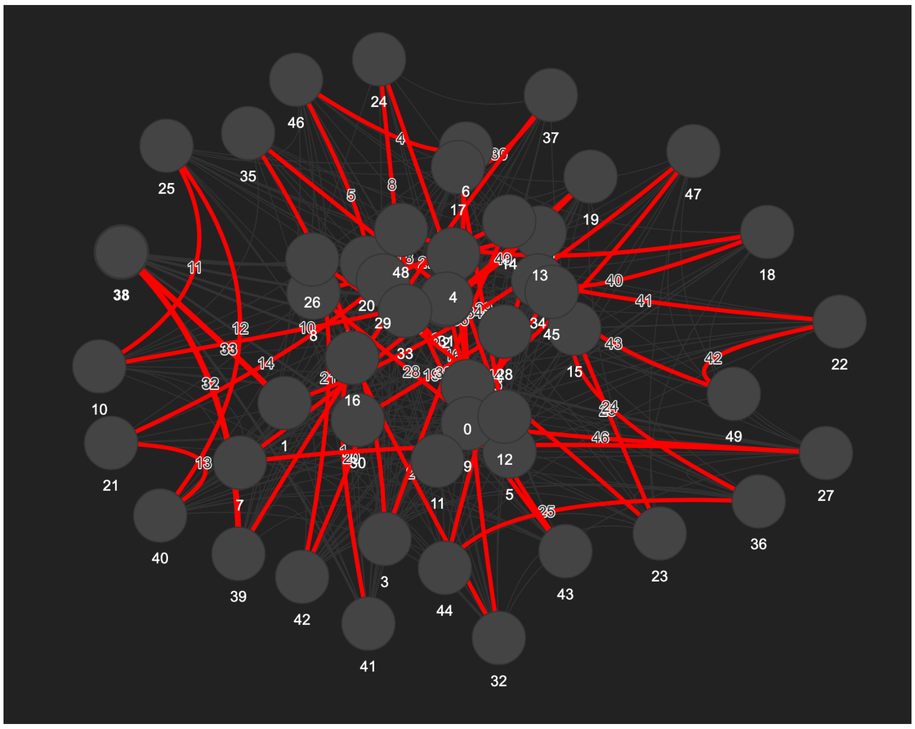

Figure 1: *Visualised with Pyvis, depicts the correct Hamiltonian path through the artificially generated, 50 node graph.*

Module: Algorithms and Combinatorial Thinking 2025-2026

Written by:
Christos Christophoros Mitsakopoulos

Supervised by: Dr. Franck Delaplace

December 23, 2025

Contents

1 Abstract

Given a graph with a set of vertices, V, a path, G, is described as a Hamiltonian path, given that it passes through each vertex of a graph exactly once. Importantly, there is no efficient polynomial-time algorithm known to solve it for general graphs. As V increases, the computational time nedded to solve it by brute force grows factorially. As such, this necessitates the use of metaheuristics for larger instances (=graphs). Importantly, the Hamiltonian path is critical for de novo genome assembly, as arranging reads in an acyclic graph – one seeks to maximise the reads used within the a single path, to create a contiguous sequence. In this report a variety of metaheuristic approaches are tested against each other and a smaller implementation of a brute force search. The approaches include: *Simulated Annealing*, *Tabu Search* and a *Genetic Algorithm* (appropriated from class), compared against a simple brute force approach for demonstration purposes.

Find all relevant results / as well as reproduce the experiment using the `christos_-mitsakopoulos.ipynb` – Jupyter Notebook, clone the repository at the following address for additional proof of work: https://github.com/cmitsakopoulos/Delaplace_coursework

2 Test Environment: Erdős-Rényi Random Graph

In order to benchmark the metaheuristic approaches in this report, an Erdős-Rényi (ER) $G(n, p)$ model was used. Where: n is the number of vertices / nodes of the ER graph, while p the probability an edge exists between two distinct nodes in that graph. Increasing the parameter p influences the edge-density of this unstructured graph type, making it less challenging to identify a Hamiltonian path. This phenomenon became apparent when testing the configuration within my `christos_mitsakopoulos.ipynb` testing environment; for instance, a p of ≈ 0.3 saw all metaheuristic approaches converge to a perfect "score" – zero broken edges (=nodes uncovered). An expected outcome given the abundance of edges between any node in the graph, and by effect multiple Hamiltonian paths for the metaheuristic algorithms to identify.

As you might observe in the Jupyter Notebook, the solution of each algorithm is encoded as a permutation vector $S = [v_1, v_2, v_3, ..., v_n]$. Expectedly, the constraint here is that each v_i (where $i \in n$), must be present only once in S – such that it represents a Hamiltonian Path.