Hamiltonian Path Approach for Genome Assembly And Its challenges

Akshay Krishnan, Aditya Vardhan Reddy, Jagath Surya, Niranjan Prasanth, Kiran P, Muhammed Shajahan

Department of computer science and Engineering

Akshay krishnan (AM.EN.U4AIE21109)

Aditya Vardhan Reddy (AM.EN.U4AIE21171)

Jagath Surya R(AM.EN.U4AIE21152)

Niranjan Prasanth P (AM.EN.U4AIE21148)

Kiran P(AM.EN.U4AIE21175)

Muhammed Shajahan(AM.EN.U4AIE21144)

***Abstract*— Hamiltonian distance is a measure of the dissimilarity between two Hamiltonian cycles in a graph. It is defined as the minimum number of edges that must be traversed in order to transform one cycle into the other. This measure is particularly useful in the field of computational biology, where it can be used to compare different solutions to the Hamiltonian cycle problem, which is to find a cycle that visits every vertex in a graph exactly once.**

One of the main challenges in computing Hamiltonian distance is that it is NP-hard, which means that it is computationally difficult to find an exact solution in a reasonable amount of time. Additionally, there are no known polynomial-time algorithms for approximating the Hamiltonian distance. As a result, researchers have developed a number of heuristic methods for estimating Hamiltonian distance, but these methods often lack accuracy and can produce unreliable results.

Another major challenge is that the Hamiltonian cycle problem is NP-hard for dense graphs which means for graphs with a high number of edges. This makes the problem even more difficult to solve, as the number of possible cycles increases exponentially with the number of edges.

Overall, the Hamiltonian distance is an important measure of the similarity between Hamiltonian cycles, but it remains a challenging problem due to its computational complexity and the lack of accurate approximate algorithms.

I. Introduction

Genome assembly is the procedure for arranging nucleotide sequences into the appropriate order.

The path in a directed or undirected graph that visits each and every vertex of the graph exactly once is known as a Hamiltonian path.

In an overlap graph, each Hamiltonian cycle corresponds to a single genome reconstruction with all the repeats resolved.

II. Literature Review

The Hamiltonian path problem, a fundamental problem in graph theory, is concerned with determining whether a given graph contains a path that visits each vertex exactly once. It was first introduced by mathematician leonhard euler in the 18th century while studying konigsberg Seven Bridges. Finding an exact solution for this problem in a larger graph can be computationally difficult as it is considered to be NP - hard.

There has been much research on the hamiltonian path problem in literature, resulting in various solutions being proposed. One popular strategy is the backtracking method, which involves systematically searching the graph for a path that visits each vertex. Another popular approach is the branch and bound method, which involves narrowing down the search tree to minimize the number of paths that need to be considered

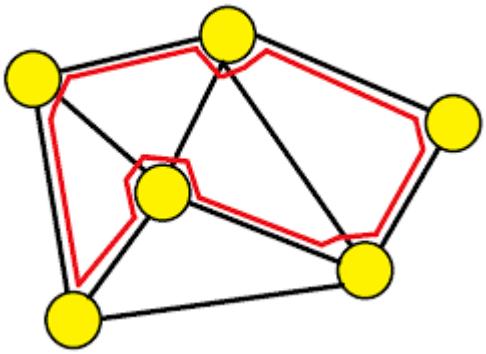
III. Methodology

A. Graph Approach

A graph is a type of data structure that is made up of vertices and edges. Vertices are represented by symbols such as A,B and C, and edges represent connections between these vertices. In this context, the vertices represent reads and edges represent overlaps between those reads. The goal is to use the overlaps to find a path that can be used to reconstruct the genome..

B. Defining the Hamiltonian path Problem

Graph G has a Hamiltonian Path from some vertex s to another vertex t if there is a path that connects the two vertices and touches each vertex in the graph exactly once.

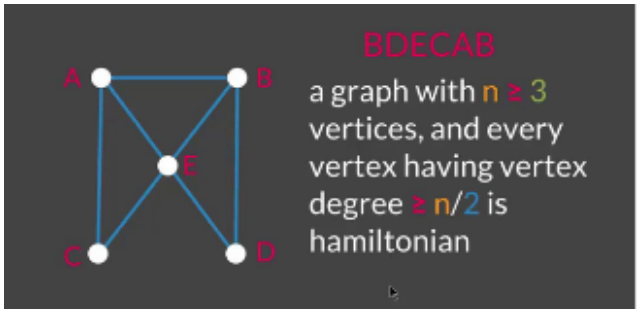


The picture above shows an example of a Hamiltonian Path.The red path shows the hamiltonian path

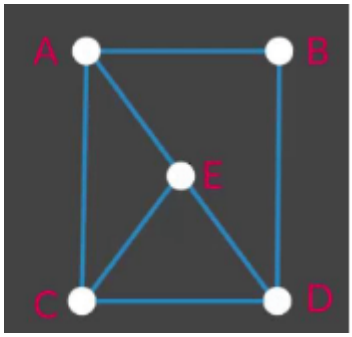
C. Criteria for a Hamiltonian Path to Exist

If a vertex in a graph only has one edge connecting to it, then it is not possible for that graph to have a Hamiltonian circuit. Additionally, there are certain rules that can be used to determine if a graph can have a Hamiltonian circuit, but they do not always guarantee its existence.

Dirac's Theorem: A Graph G with n vertices (n >= 3) is Hamiltonian if every vertex has a degree n/2 or greater.



Ore's Theorem: If for any pair of non-adjacent vertices u and v deg(u) + deg(v) >= n, then graph G contains a Hamilton circuit



Ghouila–Houiri Theorem: A strongly connected simple directed graph with n vertices is Hamiltonian if every vertex has a full degree greater than or equal to n.

Meyniel Theorem : A strongly connected simple directed graph with n vertices is Hamiltonian if the sum of full degrees of every pair of distinct non-adjacent vertices is greater than or equal to {2n-1}

D. Overlap Graph and construction

An overlap graph is a graph representation of the overlaps between sequences in a genome assembly problem. The sequences are represented as nodes in the graph, and edges between nodes indicate overlaps between the sequences. The construction of an overlap graph is a key step in building a Hamiltonian path or cycle, which is a path or cycle that visits every node in the graph exactly once.

To construct an overlap graph, the first step is to identify overlaps between the sequences. This can be done using various methods such as k-mer counting, hashing, or suffix tree construction. Once the overlaps are identified, the sequences are represented as nodes in the graph and edges are added between nodes that have overlapped. The overlaps can be represented as the length of the edge between the nodes, with longer overlaps corresponding to stronger edges.

Once the overlap graph is constructed, a Hamiltonian path or cycle can be found by searching for a path or cycle that visits every node in the graph exactly once. This can be done using algorithms such as Fleury's Algorithm, Hierholzer's Algorithm, or using a backtracking approach.

In summary, an overlap graph is a representation of the overlaps between sequences in a genome assembly problem, and its construction is a key step in building a Hamiltonian path or cycle, which visits every node in the graph exactly once.

Overlaps between reads can be determined by constructing an overlap graph (fig.1)

Node	Suffix	Prefix	Connected Nodes Suffix= Prefix
ATG	TG	AT	TGC,TGG
TGC	GC	TG	GCA,GCG
GTG	TG	GT	TGC,TGG
TGG	GG	TG	GGC
GGC	GC	GG	GCA,GCG
GCA	CA	GC	
GCG	CG	GC	CGT
CGT	GT	CG	GTG

Fig.1: represents the overlaps between reads

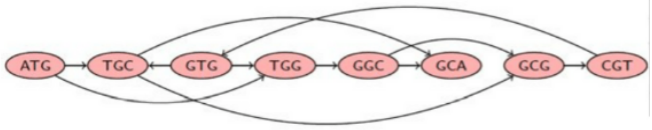


Fig2: Represent the Overlap graph built using the overlap graph.

- To Solve the string reconstruction problem, we are looking for a path in an overlap graph that visits every node exactly once
- Consecutive k-mers in a genome, are linked together to form a genome path and Consecutive k-mers has relations

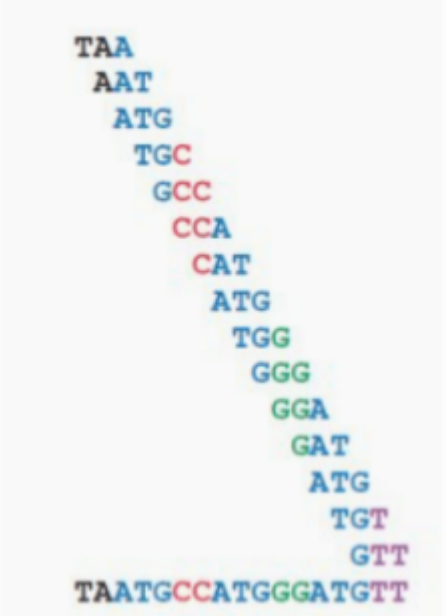


Fig.3: Represents the overlaps in the graph we made and through this, we can solve the string reconstruction problem.

This is our final string created from the group of reads we got using overlap graph method

E. ***Implementation***

The Hamiltonian path problem is implemented in Rosalind "Hamiltonian Path in DAG". The objective of the problem is to check whether the hamiltonian path exists or not. If so, return the hamiltonian path through the graph.

Problem Definition:

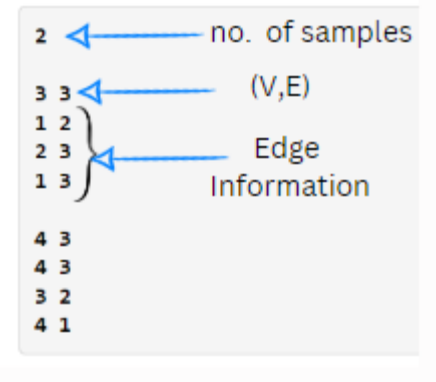
Given: A positive integer k≤20 and k simple directed acyclic graphs in the edge list format with at most 103 vertices each.

Return: For each graph, if it contains a Hamiltonian path output "1" followed by a Hamiltonian path (i.e., a list of vertices), otherwise output "-1".

Details on Dataset:

Dataset contains information about the number of sample graphs, number of vertices and edges for each sample, followed by edge information.

Sample Dataset



Goal:

The aim is to check whether a hamiltonian path exists (if exists return 1 else, return -1), followed by the hamiltonian path.

Sample Output

```
1 1 2 3
-1
```

Explanation:

1. Retrieve information from the dataset:

   Read the dataset using file handling techniques. Using the number of sample graph information, separate the graph information and store it in a list. Using attributes like number of vertices and edges and edge pair information, represent graphs in adjacency list format. Likewise, generate an adjacency list for each and every sample and store it for further operation.

2. Perform Topological Sort

   In computer science, a topological sort or topological ordering of a directed graph is a linear ordering of its vertices such that for every directed edge uv from vertex u to vertex v, u comes before v in the ordering. For instance, the vertices of the graph may represent tasks to be performed, and the edges may represent constraints that one task must be performed before another; in this application, a topological ordering is just a valid sequence for the tasks. Precisely, a topological sort is a graph traversal in which each node v is visited only after all its dependencies are visited. A topological ordering is possible if and only if the graph has no directed cycles, that is, if it is a directed acyclic graph (DAG). Note that a node is visited exactly once.

   ```
   TopologicalSort(Graph G)
   1. FIFO_Queue Q = {vertices with in-degree 0};
   2. LinkedList ll = Ø;
   3. while (Q is not empty) do
   4.    Vertex v = Dequeue(Q);
   5.    insert v into ll;
   6.    for (each vertex u such that (v, u) ∈ E) do
   7.       remove (v, u) from E;
   8.       if (in-degree of u is 0) Enqueue(Q, u);
   9.    end for
   10. end while
   11. if (E ≠ Ø) return "G has cycles";
   12. else return ll;
   ```

3. Check whether a path exists between consecutive elements returned by the topological sort. This can be checked by iterating through various nodes according to the topological traversal obtained and finding whether an edge exists between the consecutive members. If there is no edge between the consecutive members, the hamiltonian path does not exist. Continue the same for all the sample graphs. If path exists return the traversal which gives the hamiltonian path.

IV. Limitations of Hamiltonian Path Approach

NP HARD - NP-hardness (non-deterministic polynomial-time hardness) is a property of a class of problems that are informally at least as hard as the hardest problems in NP.
NP - A problem is class NP (non‑deterministic polynomial time) if a solution can be checked in polynomial time, but no known algorithm can generate the solution in polynomial time

Connected Graphs: It only applies to graphs that are connected, meaning that there is a path between every pair of vertices

Large graphs: it is difficult to find the optimal solution for large graphs, and heuristic algorithm are often used instead

It may not always exist as a hamiltonian path in a graph, even if the graph is connected.

V. Conclusions

The Hamiltonian path problem is the problem of determining whether a Hamiltonian path exists in a given graph. It is a well-known NP-complete problem, meaning that there is no known efficient algorithm to solve it for all possible inputs. However, there are efficient algorithms for solving the problem in certain types of graphs, such as planar graphs and graphs with small degree. Additionally, there are heuristic algorithms that can be used to find approximate solutions in large graphs.

VI. Acknowledgment

We, the members of the group, would like to show our gratitude towards Dr. Manjusha Nair and Athira Sudarshan , for helping us in the field of bioinformatics and putting in a lot of effort to explain the experimental approach in the field of Bioinformatics. We would also like to thank all the other faculty who helped us throughout the semester in the field of bioinformatics.

VII. References

[1] https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1103&context=etd_projects
[2] https://medium.com/@matthewwestmk/calculating-reversal-distance-using-parks-exact-greedy-algorithm-87c62d690eef
[3] Python for Bioinformatics By Sebastian Bassi
[4] Greedy algorithm - Wikipedia.
[5] Hidden_Messages.pdf (cmu.edu
[6] https://rosalind.info/problems/hdag/