International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 01 | Jan - 2025 SJIF Rating: 8.448 ISSN: 2582-3930

A Review of Discrete Mathematics in Artificial Intelligence

Neeta Ravindra Mohite [1], Dr.G.J.Chhajed[2] , Monali Rahul Bhosale3

[1]AI & DS (Computer Engineering) VP's Kamalnayan Bajaj Institute of Engineering and Technology, Baramati.

[2]HOD AI & DS (Computer Engineering) VP's Kamalnayan Bajaj Institute of Engineering and Technology, Baramati.

3Assistant Professor AI &DS (Computer Engineering) VP's Kamalnayan Bajaj Institute of Engineering and Technology, Baramati.

Abstract - The foundation of many Artificial Intelligence (AI) approaches and algorithms is discrete mathematics. Graph theory, combinatorics, and logic are just a few of the discrete mathematics fields that provide substantial contributions to AI. Each of these fields is essential to the development of contemporary AI systems. This section lays the groundwork for a more in-depth examination of particular instances by giving a summary of how discrete mathematics supports the architecture and operation of AI

***Key Words*:** Artificial Intelligence, DiscreteMathematics, GraphTheory, Combinatorics in AI.

1.INTRODUCTION

This chapter provides an introduction to mathematical induction, set theory, and the formalization of mathematical functions. In abstract mathematics, "elementary" does not mean simple; rather, it means the bare minimum of knowledge required to comprehend, even when the subject matter is basic. As a result, some concepts—even those that seem straightforward—may defy conventional wisdom and require correction.[3]
Artificial intelligence (AI) has emerged as a key component of contemporary technology development, permeating every sphere of human existence, from entertainment and security to healthcare and education. Discrete mathematics, a subfield of mathematics that deals with distinct and separable values, is fundamental to AI's operation and development.[7] In order to predict how these interactions will continue to influence AI technologies in the future, this paper attempts to clarify the intricate relationship between discrete mathematics and AI.

2. Application of discrete mathematics

The foundation of algorithms and data structures in computer science is discrete mathematics. The creation of effective computing solutions is made easier by discrete mathematics, which is used in sorting algorithms like quicksort and merge sort as well as graph algorithms like Dijkstra's algorithm and breadth-first search. [2]

The theoretical underpinnings for the creation and evaluation of algorithms and data structuresare found in discrete mathematics. Sorting, searching, optimization, and algorithmic complexity all depend on concepts like graphs, trees, sets, permutations, combinations, and probability theory.[6]

3.Information Theory

1Graph Theory in AI

The architecture of neural networks and other AI models is based on graph theory. In these situations, graphs are nodes that are connected to one another, and information is processed and delivered via these connections. In fields where interactions between data points are modeled and examined, like networking, deep learning, and online search algorithms, this structure is crucial.[5]

2 Set Theory in AI

A group of unique elements is called a set. In contrast to {1,1,3}\{1, 1, 3\}{1,1,3}, which has duplicate items, {1,2,3}\{1, 2, 3\}{1,2,3} is a set. A multiset is a collection that contains repetitions. Curly brace notation is frequently used to express sets.[9]

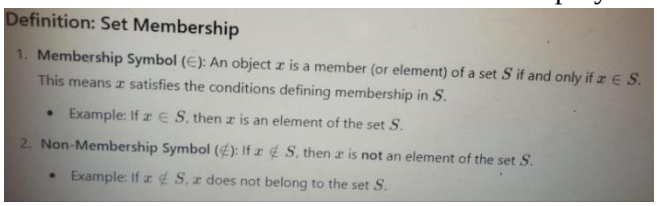
3Tree Theory

The specific properties and uses of trees in domains such as data structures, network architecture, and computer science make them a popular topic in graph theory. Being a linked graph without any cycles is a tree's most basic characteristic. Trees can be further defined and characterized with the use of other attributes and comparable conditions. Comprehending these characteristics is crucial for evaluating trees in diverse settings[1]

4.**Foundations of Sets**

Set theory definitions:

1). Definition: Establish Membership an object is a member of a set if it has the set membership symbol



2) A Set's Cardinality A set's cardinality is a measure of its size. The cardinality of a finite set is the total number of elements that make up the set.[4] The cardinality in a collection |S||S||S| is the notation for SSS. In the event that S= {1,2,3}, for instance, |S|=3. We'll introduce and examine the idea of cardinality for infinite sets

Set Operations

Similar to numbers, sets in mathematics can be worked with through a variety of operations. Set operations enable us to mix and compare sets of elements, just like addition, multiplication, and negation do for integers

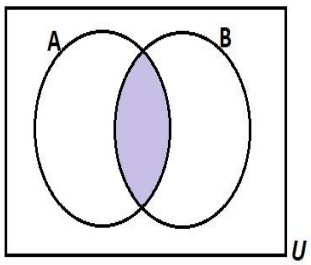


fig: Intersection(A∩B)

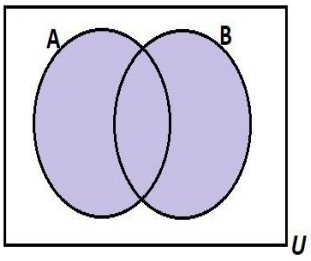


fig: Union(A∪B)

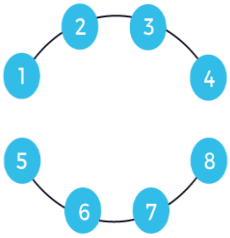


fig: Disjoint

Exploring Graphs

Graph Theory in AI The architecture of neural networks and other AI models is based on graph theory. In these situations, graphs are nodes that are connected to one another, and information is processed and delivered via these connections.[1] In fields where interactions between data points are modeled and examined, like networking, deeplearning, and online search algorithms, this structure is crucial.

Combinatorics and Learning Machines

Machine learning algorithms benefit greatly from combinatorics, especially when it comes to pattern detection and optimization issues. [6]Improving predicted accuracy and performance in machine learning tasks requires understanding the different combinations of data attributes, optimizing resource allocation, and modeling complicated relationships. Reasoning and Automated Reasoning.The basis for automated reasoning and decision-making in AI systems is logic. AI systems are capable of making judgments, proving theorems, and deriving new knowledge from data by using formal logical systems. [3]Expert systems, rule-based reasoning systems, and natural language processing (NLP) all depend on logical frameworks like propositional and predicate logic.

AI Discrete Structures

Discrete structures like sets, graphs, and functions are essential for organizing data and algorithms in addition

to the previously stated domains. [8]These structures facilitate effective algorithm creation, organize computational procedures, and model the relationships between data points. For instance in AI graphs represent relational data, functions translate data to results, and sets define input spaces. The fundamental connection between discrete mathematics and artificial intelligence is highlighted in this section, laying the groundwork for succeeding sections' more detailed examples and applications.[1] If you would want to discuss any of these subjects in.

5.Neural Networks as Graphs in Graph Theory in AI

The fundamental building blocks of machine learning, neural networks, can be efficiently depicted as directed graphs. In these models, neurons are represented by nodes, while the connections between them are represented by edges. [9]The tools needed to describe, analyze, and optimize these networks are provided by graph theory. The fundamentals of graph theory are covered in this section, along with how graph structures can be used to model and analyze neural networks in order to improve their connection, performance, and learning potential.

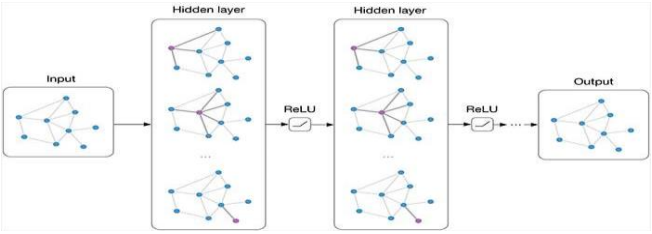


Fig: Neural Networks

AICombinatorics: Optimization Issues:

In AI optimization, combinatorics is crucial, especially when tackling issues like shortest path, which determines the smallest distance between network nodes.[4] Numerous applications of this basic issue can be found in network routing, logistics, and urban planning. Combinatorial techniques can be used to improve the accuracy and efficiency of algorithms in a variety of domains.

a) **Neural Networks with Continuous Time (CTNNs): CTNNs**' advantage is their ability to efficiently handle dynamic and continuous input. Applications of Optimization: Perfect for figuring out the shortest path around complicated problems. Time-Dependent Data: Excellent for real-time optimization and changing surroundings. The main advantages are that they provide effective real-time scheduling, routing, and decisionmaking practices.[3]

b) **Dynamic Response to Changing Input Conditions and Continuous-Time Neural Networks (CTNNs) :** CTNNs can analyze data constantly and dynamically, efficiently adjusting to changing inputs. Real-time optimization is helpful for applications where input data changes over time, allowing for iterative solution refinement. Dynamic Response: Address issues such as the Traveling Salesman Problem (TSP) by continuously modifying network conditions. The CTNN Framework's advantages improves optimization efficiency by producing higherquality solutions and faster convergence[10].

The use of C++ in CTNN research serves to illustrate how theoretical models can be applied in real-world scenarios. A conceptual use of CTNN to address a streamlined Traveling Salesman Problem (TSP) is the example focus. Relevance: Emphasizes how CTNN can be used to solve combinatorial optimization problems in the real world.

Conceptual C++ Code for CTNN Implementation:

#include <iostream> #include <vector> #include <cmath> #include <limits>

using namespace std;

const int NUM_CITIES = 5; // Example: 5 cities const double INF = numeric_limits<double>::infinity();

// Define the distances between cities (distance matrix) double dist[NUM_CITIES][NUM_CITIES] = { {0, 10, 15, 20, 25}, {10, 0, 35, 25, 30}, {15, 35, 0, 30, 5}, {20, 25, 30, 0, 15}, {25, 30, 5, 15, 0} };

// Sigmoid function (used for activation in CTNN) double sigmoid(double x) { return 1.0 / (1.0 + exp(-x)); }

// CTNN model for optimizing TSP (simplified example) class CTNN { public: vector<vector<double>>neuronStates; vector<vector<double>> synapses; CTNN() { // Initialize neuron states and synapses randomly

```
neuronStates.resize(NUM_CITIES, vector<double>(NUM_CITIES, 0.0));
synapses.resize(NUM_CITIES, vector<double>(NUM_CITIES, 0.0));

    for (int i = 0; i< NUM_CITIES; ++i) { for (int j = 0; j < NUM_CITIES; ++j) { if (i != j) { synapses[i][j] = static_cast<double>(rand()) / RAND_MAX; } } } }

  // Update neuron states based on input distances (simplified model) void updateNeuronStates() { for (int i = 0; i< NUM_CITIES; ++i) { for (int j = 0; j < NUM_CITIES; ++j) { if (i != j) { neuronStates[i][j] = sigmoid(neuronStates[i][j] - dist[i][j]); } } } }

  // Calculate the total distance (objective function) double totalDistance() { double total = 0.0; for (int i = 0; i< NUM_CITIES; ++i) { for (int j = 0; j < NUM_CITIES; ++j) { total += neuronStates[i][j] * dist[i][j]; } } return total; }

  // Perform optimization (simplified iteration) void optimize() { double prevDistance = INF; double currentDistance = totalDistance(); // Iteratively optimize the neuron states while (abs(currentDistance - prevDistance) > 1e-5) { prevDistance = currentDistance; updateNeuronStates(); currentDistance = totalDistance(); } }

};

int main () {CTNN ctnn; ctnn.optimize();

cout<< "Optimized Total Distance: " <<ctnn.totalDistance() <<endl; return 0; }
```

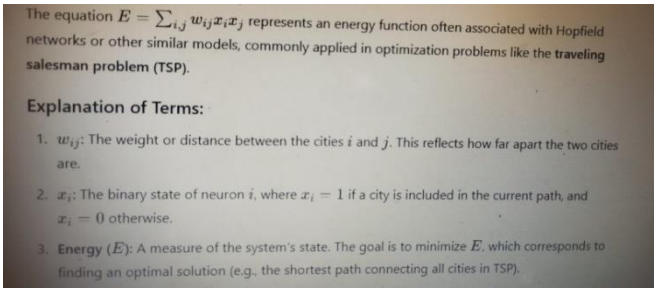
The Traveling Salesman Problem (TSP) :

Overview of the Traveling Salesman Problem (TSP) Definition: The shortest path for a salesman to travel to every city exactly once and then return to the starting point is a classic NP-hard issue. Complexity: The more cities there are in the challenge, the more computationally challenging it becomes. Applications: Used in logistics, urban planning, and network routing. Optimization Need: Techniques like neural networks seek to effectively identify near-optimal solutions for big datasets.

Method of solving the TSP:

1.City Encoding as Neurons: In the TSP, every city is represented by a network neuron. The salesman has a variety of options, which are represented by the states of the neurons. Finding the state configuration that reducestheoverall travel distance is the network's objective. [10]

2. Representation of the Energy Function: The salesman's entire journey distance is represented by the energy function in a Hopfield network. The network develops in a way that minimizes the energy function, which is equivalent to taking the shortest path.


The equation $E = \sum_{i,j} w_{ij} x_i x_j$ represents an energy function often associated with Hopfield networks or other similar models, commonly applied in optimization problems like the traveling salesman problem (TSP).

Explanation of Terms:

1. w_{ij}: The weight or distance between the cities i and j. This reflects how far apart the two cities are.
2. x_i: The binary state of neuron i, where $x_i = 1$ if a city is included in the current path, and $x_i = 0$ otherwise.
3. Energy (E): A measure of the system's state. The goal is to minimize E, which corresponds to finding an optimal solution (e.g., the shortest path connecting all cities in TSP).

As the network progresses toward a solution, the energy drops[2].

3. **Time-Continuous Dynamics:** The CTNNs modify the states of the neurons through continuous-time dynamics. The Hopfield dynamics dictate that the neuron states evolve continuously rather than in discrete time steps: τdxidt = −∂E∂xi\tau \frac{d x_i}{dt} = -\frac{\partial E}{\partial x_i}"τdtdxi=−∂xi∂E" Where xix_ixi is the state of neuron iii, τ\tauτ is a temporal constant, and EEE is the energy function that the network aims to minimize. The network will eventually converge to the energyminimization state,

An Example of Application An analog neural network circuit or an FPGA (Field-Programmable Gate Array) device are two examples of hardware configurations that can be used to create CTNNs for TSP solving. Compared to conventional, entirely digital algorithms, these systems can process big datasets in

parallel and find solutions more quickly by utilizing the continuous nature of CTNNs. Because of their efficiency and versatility, CTNNs are a desirable choice for large-scale, real-time optimization issues.

6.Planar Graphs

A graph is said to be planar if none of its edges cross when it is depicted on a plane. The discrete areas created by the vertices and edges in such a drawing, including the "outside" region, are referred to as faces. Below is a summary of your questions: 2. Two distinct planar graphs that have the same quantity of faces, edges, and vertices Indeed, two distinct planar graphs with the same number of vertices, edges, and faces can be made.[1] Here's an illustration:

• Graph 1: A straightforward triangle with three vertices, three edges, and one face

• Graph 2: A square with two faces, four vertices, and four edges with a single diagonal

There are two distinct planar graphs that have the same number of edges and vertices but different numbers of faces.

Planar graphs with a variable number of faces but the same number of vertices and edges can also be made. As an example:

Graph 1: There is just one face when a complete graph with four vertices (K_4) is depicted as a planar graph.

3. Is It Possible to Draw a Graph Without Any Edges Crossing? If a graph can be drawn on a plane without any edges crossing, then it is said to be planar. Kuratowski's theorem, which asserts that a graph is planar if and only if it does not contain a subgraph that is a subdivision of either K_5 (the full graph with 5 vertices) or $K_{3,3}$ (the complete bipartite graph with 3 vertices in each set), makes this possible for graphs that meet its requirements.[10] A graph is considered non-planar if it contains one of these two minimal nonplanar graphs as a subgraph.

4. Graph Redrawing for Planarity

As mentioned, if you can redraw a graph so that no edges cross, you can make it appear planar even if it doesn't at first. For instance, the graph below may appear non-planar at first, but it may be rebuilt to show that it is planar by avoiding edge crossings.

5.Illustration of a Three-Faced Planar Graph

Take the graph with three sides, including the outer region, as an example.[5] A triangle with a diagonal is among the most basic planar graphs that have precisely three faces. The plane is divided into three areas by its edges and vertices

7.Tree Structures

Spanning tree types include:

Trees with a maximum degree limitation on each vertex are known as bounded-degree spanning trees. helpful when there are connectivity limitations. Spanning Trees with Structural Features: Trees tailored to meet particular requirements, like Bounded Number of Leaves: Trees with a restricted number of ends, frequently for dependability or efficiency.[10]

a) Trees with branching restrictions that minimize intricate connections are known as bounded number of branch vertices.[9]

b) A tree that spans every vertex with the smallest possible total edge weight is known as a Minimum Weight Spanning Tree (MST). essential for designs that are economical. MSTP, or the Minimum Weight Spanning Tree Problem:

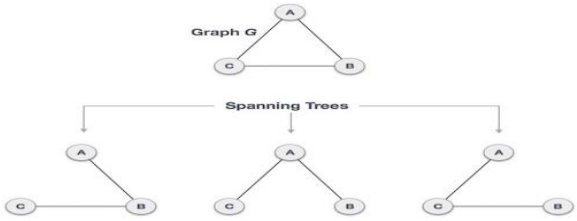


Fig: Spanning Tree

Additional Tree Properties

• Spanning Tree: A spanning tree is a subgraph of a linked graph that is a tree in and of itself and contains every vertex in the original graph. [4]At least one spanning tree exists in every connected graph. n−1n-1n−1 edges make up a spanning tree, where nnn is the number of vertices in the original graph

. • Leaf Nodes: Vertices with a single edge (degree 1) in a tree are known as leaf nodes. The "endpoints" of the tree are the common term for them.[7] In algorithms that investigate tree structures, such as depth-first search (DFS) and breadth-first search (BFS), an understanding of leaf nodes is crucial.

Importance of Spanning Tree Variants

The significance of spanning tree variations is in their ability to provide flexibility in addressing certain

application restrictions and optimization objectives.[10] Relevance to Practice: discovered in domains such as infrastructure construction, transportation, and telecommunications.

8.Hierarchical Structure of Rooted Trees

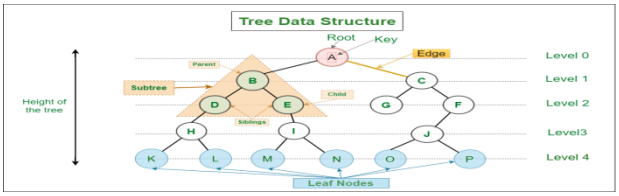


Fig: Hierarchical Structure of Rooted Trees

9.CONCLUSION

The study concludes that highlights discrete mathematics' fundamental importance in artificial intelligence (AI). Important fields including combinatorics, set theory, and graph theory are emphasized for their contributions to the design, development, and optimization of AI algorithms. [4][9]By demonstrating the harmony between theoretical ideas and real-world applications, the research further emphasizes the importance of continuous-time neural networks (CTNNs) in resolving combinatorial challenges such as the Traveling Salesman Problem. Discrete mathematics integration guarantees improvements in AI's effectiveness, scalability, and real-time flexibility

REFERENCES

[1] Chen, L., & Gupta, S.Discrete Mathematics in Artificial Intelligence: A Comprehensive Review Journal Artificial Intelligence and Discrete Mathematics, Vol.7,Issue.2, pp112-130, 2023.

[2] F. Glover, Future Paths for Integer Programming and Links to Artificial Intelligence, Computers and Operations Research, Vol.13, Issue.5, pp.533-549, 1986.

[3] Gendreau, M., Hertz, A., & Laporte, G. A review of heuristics for the vehicle routing problem. Operations Research, Vol.44, Issue.3, pp378-400, 1996.

[4] Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences, Vol.79, Issue.8, pp2554-2558, 1982.

[5] Hopfield JJ., Tank DW., " Neural" computation of decisions in optimization problems Biological Cybernetics, Vol.52, Issue.3, pp141-152, 1985.

[6] Kennedy, M. P., & Chua, L. O. Neural networks for nonlinear programming. IEEE Transactions on Circuits and Systems, Vol.35, Issue.5, pp554-562, 1988.

[7] Laporte, G. The traveling salesman problem: An overview of exact and approximate algorithms. European Journal of Operational Research, Vol.59, Issue.2, pp231-247, 1992.

[8] Patel, R., & Wang, Graph-Theoretic Approaches to Machine Learning: A Survey , Journal of Artificial Intelligence and Discrete Mathematics Vol.5, Issue.3, pp201-220, 2022.

[9] Smith, K., & Lee, T. H. Continuous-time neural networks for combinatorial optimization. IEEE Transactions on Circuits and Systems I: Regular Papers, Vol.55, Issue.8, pp2451-2461, 2008.

[10] Yang, S., et al. On the convergence of a class of continuous time stochastic processes. IEEE Transactions on Automatic Control, Vol 62, Issue.12, pp6534-6549, 2017