International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2024.0479
e ISSN: 2584-2854
Volume: 02
Issue: 11 November 2024
Page No: 3256-3260

Numerical Data Processing by The Implementation of Trees and Graphs

Dhanashri Korpad[1], Nisha Satpute[2], Nayana Joshi[3], Snehal Kulkarni[4], Komal Walgude[5], Neha Dhadiwal[6]

[1,2,3,4,6]Assistant Professor, Vishwakarma College of Arts, Commerce and Science, Pune, Maharashtra, India.

[5]HMIS Executive, Symbiosis Medical College for Women, Pune, Maharashtra, India.

Email ID: *dhanashrikorpad1991@gmail.com[1], nisha.satpute3@gmail.com[2], nayanajoshi80@gmail.com[3], snehal.lad87@gmail.com[4], kswalgude16@gmail.com[5], neha.dhadiwal07@gmail.com[6]*

Abstract

Trees and Graphs play a vital role in transport and logistics. In tree, decision tree is one of the important, not only implemented for data processing, but also considered for Numerical data analysis. The decision tree is a flow chart-like structure, in which each internal node represents a 'test' on an attribute, which has a node known as root being at the top, which further divides the given data into branches depending upon the conditions. Every branch consists of a rule, and each leaf node is its outcome. A support tool with a tree-like structure that models probable outcomes, cost of resources, utilities, and possible consequences. Decision trees are also used in operations research along with planning logistics. They can help in determining appropriate plans that will help a company achieve its target. In Graph, Dijkstra's algorithm is an admired algorithm used to find the shortest paths between nodes in a graph, which may represent road networks for example. Dijkstra's algorithm finds the shortest path from a given source node to every other node. It can also be used to find the shortest path to a specific destination node, by concluding the algorithm once the shortest path to the destination node is found. It is also commonly used on graphs where the edge weights are in real numbers. A common application of shortest path algorithms is network routing protocols and also support in route optimization for delivery services, ensuring timely and cost-effective deliveries by finding the best paths for transportation.

Keywords: *Data Processing; Dijkstra's Algorithm; Graph; Shortest Path; Tree*

1. Introduction

This paper aims to explore the theoretical foundations and practical applications of shortest path algorithms, particularly Dijkstra's algorithm, while examining how decision trees can enhance decision-making in this context. Through a comprehensive analysis of these topics, one can highlight their significance in contemporary computational problems and identify potential areas for future research and application. The ability to quickly determine the shortest path not only saves resources but also enhances overall efficiency in transportation and communication. As global networks continue to expand, the need for effective pathfinding solutions becomes increasingly vital. Numerical Data Processing is more quantitative in nature than qualitative. Both Decision tree and Dijkstra's algorithm, are one of the perfect algorithms for quantitative analysis.

2. Decision Tree

Decision tree is the process that helps to collect data which gather relevant data related to the incident based on the type which extract the relevant features from the data for that time of day and location. With the help of the decision tree, one can model the data using the labelled incident and then evaluate and deploy with a real time environment, to detect the incident with the help of predictive maintenance and safety monitoring which will identify the potential safety risks or hazards.

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2024.0479
e ISSN: 2584-2854
Volume: 02
Issue: 11 November 2024
Page No: 3256-3260

2.1. Diagram of Decision Tree of PMPML (Pune Mahanagar Parivahan Mahamandal Ltd):

2.1.1.Data Analysis on the Basis of Time Form Saswad to Swargate [1]

Route 1: Saswad - Dive Ghat- Hadapsar- Fatima Nagar- Swargate.
Route 2: Saswad- Bop Dev Ghat- Khadi Machine Chowk- Swargate.

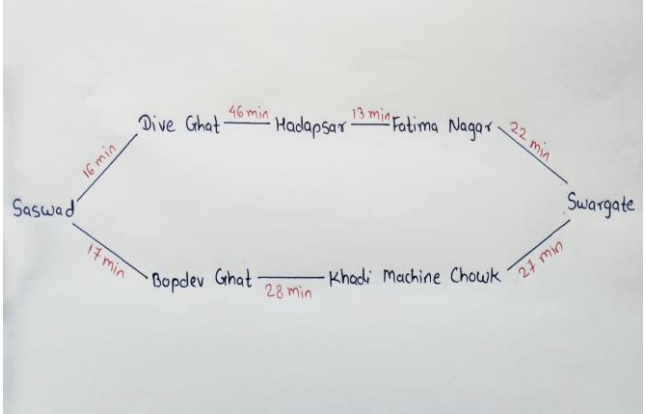

Figure 1 PMPML Bus Route from Saswad to Swargate(Pune)

2.1.2. Total Time Taken

Total Time Taken for Route 1- 1 hr 12 mins
Total Time Taken for Route 2- 1 hr 37 mins
Decision tree supports the partitioning structure i.e., tree model. the algorithm here displays the analysis on time, in the given decision tree source and destination are the two places of pune city, where the source is Saswad town and destination is Swargate pune city. Here the decision tree gives us a brief idea about the branches with specified rule (Time). By analyzing the source and destination, there are two specific paths as mentioned. Implementing the above decision tree in C language, to analyze the Pune Municipal Transport (PMT) bus system, which will help in making decisions based on various factors, such as route selection, bus timing, or passenger demand. A decision tree consists of nodes representing decisions or outcomes based on certain attributes. In our case, Figure 1 shows PMPML Bus Route from Saswad to Swargate(Pune) such as distance, time, and passenger count. Below is a simplified example of how to implement a basic decision tree in C Language, Shown in Figure 2 & 3.

2.2. C Language Implementation of a Simple Decision Tree

Here's an illustrative implementation of a decision tree that decides the best bus route based on hypothetical data [2].

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
// Define a structure for the decision tree node
typedef struct Node {
    char *question;
    struct Node *yes;
    struct Node *no;
    char *result;
} Node;

// Function to create a new tree node
Node* createNode(char *question, char *result)
{
 Node *newNode = (Node *) malloc (sizeof(Node));
    newNode->question = question;
    newNode->yes = NULL;
    newNode->no = NULL;
    newNode->result = result;
    return newNode;
}
// Function to build a sample decision tree for two routes
Node* buildTree() {
    // Create leaf nodes for the routes
    Node *route1 = createNode(NULL, "Take Route 1");
    Node *route2 = createNode(NULL, "Take Route 2");

    // Create decision nodes
Node *timeDecision = createNode("Is travel time less than 30 mins?", NULL);
Node *passengerDecision = createNode("Are there more than 10 passengers?", NULL);

    // Build the tree structure
    timeDecision->yes = route1; // If yes, take Route 1
 timeDecision->no = passengerDecision; // If no, check passenger count
```

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2024.0479
e ISSN: 2584-2854
Volume: 02
Issue: 11 November 2024
Page No: 3256-3260

```
    passengerDecision->yes = route2; // If yes, take Route 2
    passengerDecision->no = route1; // If no, take Route 1 (fallback)
return timeDecision;
}
// Function to traverse the tree and make a decision
void makeDecision(Node *node) {
    if (node->result! = NULL) {
        printf("%s\n", node->result);
        return;
    }
    char answer [4];
    printf("%s (yes/no): ", node->question);
    scanf("%s", answer);
    if (strcmp(answer, "yes") == 0) {
        makeDecision(node->yes);
    } else {
        makeDecision(node->no);
    }
}

// Main function
int main () {
    // Build the decision tree
 Node *decisionTree = buildTree();
 printf("Decision Tree for Pune PMT Bus Routes:\n");
makeDecision(decisionTree);
    return 0;
}
```

```
Decision Tree for Pune PMT Bus Routes:
Is travel time less than 90 mins? (yes/no): yes
Take Route 1
```

Figure 2 Result

```
Decision Tree for Pune PMT Bus Routes:
Is travel time less than 90 mins? (yes/no): no
Are there more than 10 passengers? (yes/no): yes
Take Route 2
```

Figure 3 Result

```
Decision Tree for Pune PMT Bus Routes:
Is travel time less than 90 mins? (yes/no): no
Are there more than 10 passengers? (yes/no): no
Take Route 1
```

Figure 4 Result

3. Dijkstra's Algorithm

Dijkstra's algorithm is designed to find the shortest paths from a starting node to all other nodes in a graph, with non-negative edge weights in kilometres. The algorithm uses a priority queue to explore nodes based on the cumulative cost from the start node [3].

3.1. Graph Representation

As Dijkstra's algorithm is used for directed as well as undirected graphs, which is also implemented in the Pune Metro System. Where all stations are nodes and routes are paths or edges which are connected between two stations. So according to stations, travel time and distance change.

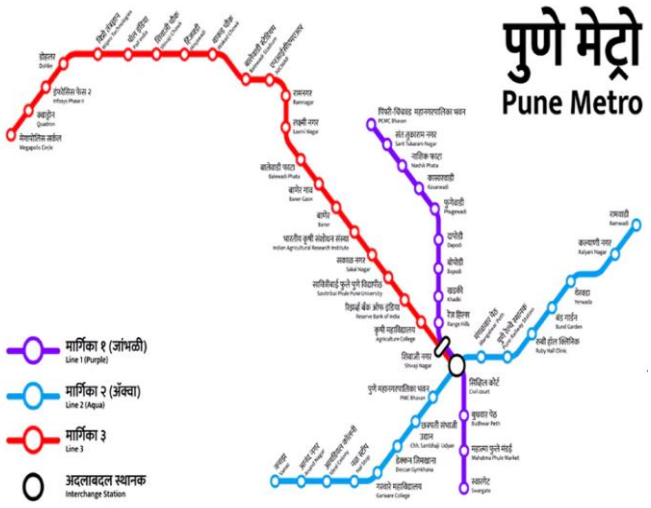

Figure 4 Pune Metro Map [4]

In the context of the Pune Metro, the graph can be represented as follows:

- **Nodes:** Metro stations (e.g., PCMC, Swargate, Shivajinagar).
- **Edges:** Tracks between the stations, with weights representing travel time or distance.

3.1.1. Assuming Some Stations and Travel Times

PCMC: [(Sant Tukaram Nagar, 5), (Kasarwadi, 10)] Sant Tukaram Nagar: [(PCMC, 5), (Pune Junction, 15)] Kasarwadi: [(PCMC, 10), (Pune Junction, 8)] Pune Junction: [(Sant Tukaram Nagar, 15), (Swargate, 20)] Swargate: [(Pune Junction, 20)]

3.2. Dijkstra's Algorithm Implementation

Here's a simplified implementation in C Language to find the shortest path in the Pune Metro network [2].

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2024.0479
e ISSN: 2584-2854
Volume: 02
Issue: 11 November 2024
Page No: 3256-3260

```
#include <stdio.h>
#include <limits.h>
#include <stdbool.h>
#define V 5 // Number of vertices in the graph
// Function to find the vertex with the minimum distance
int minDistance(int dist[], bool sptSet[]) {
   int min = INT_MAX, min_index;
   for (int v = 0; v < V; v++) {
      if (!sptSet[v] && dist[v] <= min) {
         min = dist[v];
         min_index = v;
      }
   }
   return min_index;
}

// Function to implement Dijkstra's algorithm
void dijkstra(int graph[V][V], int src) {
   int dist[V]; // Output array. dist[i] holds the shortest distance from src to j
   bool sptSet[V]; // sptSet[i] will be true if vertex i is included in the shortest path tree

   // Initialize all distances as INFINITE and sptSet[] as false
   for (int i = 0; i < V; i++) {
      dist[i] = INT_MAX;
      sptSet[i] = false;
   }

   // Distance from source to itself is always 0
   dist[src] = 0;

   // Find the shortest path for all vertices
   for (int count = 0; count < V - 1; count++) {
      // Pick the minimum distance vertex from the set of vertices not yet processed
      int u = minDistance(dist, sptSet);
      sptSet[u] = true; // Mark the picked vertex as processed
      // Update dist value of the adjacent vertices of the picked vertex
      for (int v = 0; v < V; v++) {
         // Update dist[v] if and only if it is not in sptSet, there is an edge from u to v,
         // and the total weight of path from src to v through u is smaller than the current value of dist[v]
         if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX && dist[u] + graph[u][v] < dist[v]) {
            dist[v] = dist[u] + graph[u][v];
         }
      }
   }
   // Print the constructed distance array
   printf("Vertex\tDistance from Source\n");
   for (int i = 0; i < V; i++) {
      printf("%d\t\t%d\n", i, dist[i]);
   }
}

// Main function to test the above functions
int main() {
   // Example graph represented as an adjacency matrix
   int graph[V][V] = {
      {0, 5, 0, 0, 10},
      {5, 0, 0, 15, 0},
      {0, 0, 0, 20, 0},
      {0, 15, 20, 0, 25},
      {10, 0, 0, 25, 0}
   };

   dijkstra(graph, 0); // Starting from vertex 0 (PCMC)
return 0;
}
```

```
Vertex  Distance from Source
0       0
1       5
2       40
3       20
4       10
```

Figure 5 Result

4. Disadvantage

Above mentioned code of Decision Tree and Dijkstra Algorithm, are well defined for the implementation in numerical analysis, for finding the shortest route between the source and destination, but there are few dis-advantage of the both algorithms, which are listed below, Shown in Figure 4 & 5.

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2024.0479
e ISSN: 2584-2854
Volume: 02
Issue: 11 November 2024
Page No: 3256-3260

4.1. Decision Tree

- They are largely unstable compared to other decision predictors. A small change in data can result in a major change in the structure of the decision tree
- Use quantitative data only, and ignore qualitative aspects of decision.

4.2. Dijkstra Algorithm

- Non-Negative Weights Only: Dijkstra's algorithm works only with graphs that have non-negative edge weights. If there are negative weights, the algorithm can produce incorrect results [5]
- Single Source: Dijkstra's algorithm finds the shortest path from a single source to all other vertices, which might not be efficient if only the shortest path to a specific destination is needed.

Conclusion

This simple decision tree implementation in C Language can help analyse decision-making processes for the Pune PMT bus system based on user-defined criteria. You can expand the tree with more complex questions and routes as needed.

This implementation of Dijkstra's algorithm in C Language provides a foundational approach to finding the shortest paths in a graph, applicable to scenarios like metro systems. You can modify the graph matrix to represent different metro networks as needed.

References

[1] Time Table. (n.d.). https://pmpml.org/assets/schedule/170866629042bbb3b37009c5fd5bea2b0df59c1b62.pdf
[2] 1st-year-study-materials-vssut/let-us-c-by yashvant- kanetkar.pdf at master msatul1305/1st-year-study-materials-vssut. (n.d.). GitHub. Retrieved October 29,2024, from https://github.com/msatul1305/1st-year-study-materials-vssut-study-materials-vssut/blob/master/ let-us-c-by-yashavant-kanetkar.pdf
[3] Rosen, K. (n.d.). Discrete Mathematics and its applications. (Seventh Edition ed.). Tata McGraw Hill.
[4] (n.d.).https://punemetrorail.org/route-map.aspx
[5] Dijkstra's-Algorithm. (n.d.). https://www.geeksforgeeks.org/why-does-dijkstras-algorithm-fail-on-negative-weights/.