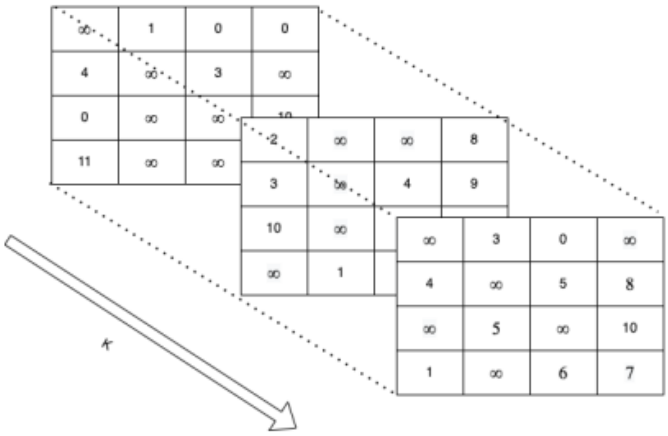
DEPARTMENT OF CIVIL ENGINEERING
INDIAN INSTITUTE OF
TECHNOLOGY
MADRAS
CHENNAI-600 036

Journey Planner Algorithm



A Report

Submitted by

Sayash Raaj Hiraou

May, 2023

Acknowledgments

I would like to acknowledge and give my thanks to Professor Bhargava Rama Chilukuri, Department of Civil Engineering, IIT Madras and Professor Pramesh Kumar, Department of Civil Engineering, IIT Madras.
I thank the Department of Civil Engineering, IIT Madras for providing the opportunity to work on the project.

Abstract

Journey planning deals with identifying the optimal route to minimise travel time without compromising on the pre-decided time to be spent at each location and involves a significant time investment when done manually. Existing TSP methods focus on heuristics and networks without time constraints. The Journey Planning problem statement focuses on finding the optimal route on a network with time-dependent links.

The problem is formulated as a graph routing problem where train stations are modelled as nodes and trains connecting subsequent stations are modelled as links, with the objective of minimising the travel time. Two novel methods are proposed: Using a 3-Dimensional network and a time-expanded network.

An integer programming approach has been proposed, including bounds and constraints imposed on the decision variables. A modified version of the 62-year-old Miller, Tucker, and Zemlin constraints for a 3-Dimensional matrix representation of a network has been developed independently. A time-expanded network representation for train data has been developed and used to find the shortest tour with the given constraints. The present work aims to develop an efficient algorithm to automate the process of journey planning, formulating it as a graph routing problem with additional constraints mimicking real-life situations and objectives. The algorithm has been tested on sample and real-world networks and has been found to perform well.

KEYWORDS: Graph, Routing, Integer Programming, Algorithms, Journey Planner, Modified TSP, Time-Expanded Graph Network, Minimise Travel Time

TABLE OF CONTENTS

CHAPTER 4 DISCUSSION AND CONCLUSIONS

LIST OF TABLES

LIST OF FIGURES

Glossary

The following are some of the commonly used terms in this report:

- **Nodes**- as referred to in graphs, analogous to locations to visit using the Journey Planner

- **Link**- Connectivity between a pair of nodes with a start time and travel cost

- **Tour**- A directed path connecting more than one node

Abbreviations

- **TSP**- Travelling Salesman Problem
- **GA**- Genetic Algorithm
- **3D**- 3-Dimensional
- **MTZ**- Miller, Tucker, Zemlin
- **DFJ**- Dantzig, Fulkerson, Johnson
- **IP**- Integer Programming
- **MDA-** Multiple Destination Approach

Notations

- ∞ - Infinity, or a large value
- **n** - Number of nodes
- **i, j** - Indices of respective nodes
- **k** - Number of links between each pair of nodes
- $\mathbf{C_{ijk}}$ - Cost of travel of link *ijk*
- $\mathbf{Start_{ijk}}$ - Start time of link *ijk*
- $\mathbf{Buffer_i}$ - Time to spend at a node/place
- $\mathbf{x_{ijk}}$ - Binary, 0 if the link is not selected, 1 otherwise
- $\mathbf{u_i}$ - Rank of node
- $\mathbf{t_i}$ - Exit time of a node. Example- arrive at node 2 at 1 pm, buffer time is 3 hours. This means we are ready to leave the node at 1 pm + 3 hours = 4 pm. Hence, t_2 = 4 pm

Chapter 1

Literature Review

Journey planning requires deciding a route to follow while minimising the travel time. The route has to be planned such that it does not compromise spending time at a place.

Manually, this task can take a significant amount of time. Deciding the order of locations to visit, keeping the timings of trains leaving from each location to the next and optimising for spending the least amount of time travelling is a tedious task. The difficulty grows with the number of locations to plan for.

Additionally, certain constraints have to be considered that mimic real-life conditions, including but not limited to-

- visiting all locations without skipping any
- visiting each location not more than once
- taking into account a waiting period for the link/train from one location to another

1.1 Existing Algorithmic Solutions

Algorithmic solutions for the problem exist, including but not limited to the naïve brute-force solution. The brute force solution computes all the possible paths that can be taken using the input provided and decides on the path which satisfies the objective. This is a time and resource-intensive solution. The running time of the solution grows exponentially with the input provided and becomes infeasible after a certain number of locations are provided as input (refer to Table 1).

The brute force solution is usually used as a baseline for comparing the efficiency of algorithms. With a time complexity of the order of the factorial of the number of nodes and 10^9 operations per second, we find the estimated solution time as follows

Table 1.1 Brute force solution estimated runtime

Number of Nodes	Estimated Solution Time
10	10 seconds
20	77.14 years
25	491.81 million years
30	8.41 x 10^15 years
50	9.64 x 10^47 years

There are multiple algorithmic solutions, some of them discussed and referenced in this report.

1.2 Potential Solutions

To solve this problem, we looked at multiple potential solutions. From the literature review, we shortlisted multiple candidate approaches which seemed feasible and aligned with the problem statement.

The approaches include-

A. Heuristic algorithms to solve the problem in the least amount of time, but with a trade-off for the accuracy of the final solution
B. Classic cost minimisation algorithms and their variants, most of which are resource-intensive
C. Integer programming uses mathematical formulations of each aspect of the problem to find a constrained solution using the input provided
D. Genetic Algorithms, using multiple simulated generations of pathfinding iterations and building upon a probabilistic approach
E. Biomimicry, similar to genetic algorithms but with more focus on mimicking biological systems like ants using trail pheromones

The problems with the mentioned approaches

- Heuristic algorithms have an approximate solution and not the most optimal one
- GAs might not converge, or might converge at a local minima
- Biomimicry has the same pitfalls as GAs

Out of the approaches reviewed, a combination of B. and C. aligns best with our goal of building an accurate and fast algorithm to solve what brute force takes exponentially long to solve.

1.3 The Travelling Salesman Problem

TSP, the Travelling Salesman Problem, is one of the oldest routing problems in literature, dating back to at least 1932- Menger, (1932). The problem is simple- solving for a minimum-cost tour visiting each node once and returning to the source node. The Journey Planner problem shares some characteristics with TSP.

It would be possible to formulate the Journey Planner problem as a constrained variation of TSP. However, classical algorithmic approaches would not work due to the exponential time complexity mentioned in Table 1.

Learning from the past and current methods used to solve TSP, starting from the classic TSP papers: MTZ (Miller, Tucker and Zemlin, 1960) and DFJ (Dantzig, Fulkerson and Johnson, 1954), multiple approaches towards formulating the problem and its constraints were studied. Several books have been referred to for formulating the problem and the constraints to use.

1.4 The TSP Variant

An integer programming approach has been proposed which includes mathematically formulating the objective and constraints, solving for the minimum cost tour. The cost is defined as the time taken to travel on the network.

Current integer programming approaches for the classic TSP include formulating the constraints primarily in two ways-

- MTZ constraints
- DFJ constraints

Choosing the ones to use depends on our formulation of the problem statement, and thus research was done on both methods.

The base formulation of the problem includes

- Restricting selection of one incoming link from a node
- Restricting selection of one outgoing link from a node

$$\min \sum_{i \neq j}^{n} C_{ij}.x_{ij} \tag{1.1}$$

$$\forall\, i \leq n \quad \sum_{j=1}^{n} x_{ij} = 1 \tag{1.2}$$

$$\forall\, j \leq n \quad \sum_{i=1}^{n} x_{ij} = 1 \tag{1.3}$$

defining x_{ij} as a binary variable-

- 1 if the link is chosen

- 0 if the link is not chosen

$$\forall\, i,j \leq n \quad x_{ij} \in \{0,1\} \qquad (1.4)$$

The base constraints defined are for the classic TSP with a 2D representation of the network. Our problem includes multiple time-based links between each pair of nodes and hence requires a 3D representation of the network as shown in Figure 1

Exploring the integer programming formulation of the classic TSP and using examples with pre-computed solutions to verify the correctness, a visual representation of the final tour has been developed using a script run post computation of the optimal x_{ij} matrix solution.

Table 1.2 Classic TSP Problem, 17 nodes

∞	3	5	48	48	8	8	5	5	3	3	0	3	5	8	8	5
3	∞	3	48	48	8	8	5	5	0	0	3	0	3	8	8	5
5	3	∞	72	72	48	48	24	24	3	3	5	3	0	48	48	24
48	48	74	∞	0	6	6	12	12	48	48	48	48	74	6	6	12
48	48	74	0	∞	6	6	12	12	48	48	48	48	74	6	6	12
8	8	50	6	6	∞	0	8	8	8	8	8	8	50	0	0	8
8	8	50	6	6	0	∞	8	8	8	8	8	8	50	0	0	8
5	5	26	12	12	8	8	∞	0	5	5	5	5	26	8	8	0
5	5	26	12	12	8	8	0	∞	5	5	5	5	26	8	8	0
3	0	3	50	50	8	8	5	5	∞	0	3	0	3	8	8	5
3	0	3	48	48	8	8	5	5	0	∞	3	0	3	8	8	5
0	3	5	51	51	8	8	5	5	3	3	∞	3	5	8	8	5
3	0	3	48	48	8	8	5	5	0	0	3	∞	3	8	8	5
5	3	0	72	72	48	48	24	24	3	3	5	3	∞	48	48	24
8	8	50	6	6	0	0	8	8	8	8	8	8	50	∞	0	8
8	8	50	6	6	0	0	8	8	8	8	8	8	50	0	∞	8
5	5	26	12	12	8	8	0	0	5	5	5	5	26	8	8	∞

Table 1.3 Classic TSP Problem, 17 nodes Solved

1	0
2	7
3	3
4	13
5	12
6	11
7	10
8	16
9	15
10	6
11	5
12	1
13	4
14	2
15	9
16	8
17	14

The ranks of the nodes denotes the order of visiting each node in the optimal complete tour of the Travelling Salesman Problem

1.4.1 The MTZ constraints

The MTZ constraints mathematically define a one-line constraint to avoid subtours in the network such that all nodes are visited and none are left from the final tour.

$$\forall\, i \neq j \neq 1, \quad u_i - u_j + nx_{ij} \leq n - 1 \tag{1.5}$$

Where u is a decision variable denoting the rank, or order, of the node visited in the tour, with a range of [0,n-1] using 0-based indexing. The basic principle behind the constraint is the comparison of ranks between the current node and the next node to be visited, depending if a particular link xij is selected or not.

If a node is selected,

$$u_i-u_j+n<=n-1 \tag{1.6}$$

Simplified,

$$u_i+1<=u_j \tag{1.7}$$

Thus, the rank of the next node to visit, node j, has to be at least one more than the rank of the current node to denote positive route progress.

If u_j is less than u_i, node j has been visited before the current node and cannot be selected as the next node in the tour

If a node is not selected,

$$u_i-u_j+0<=n-1 \tag{1.8}$$

Simplified,

$$u_i-u_j<=n-1 \tag{1.9}$$

Which is true for all values of u, as the bounds of u lie between [0,n-1] and so does the difference between any two values in the range.

The value of n-1 can be considered as the smartly chosen value of Big M used in integer programming to model the indicator constraints.

1.4.2 The DFJ constraints

The DFJ constraints aim to eliminate subtours like the MTZ constraints do, but require exponentially higher amounts of computation when precalculating all constraints.

$$\sum_{i,j \in S}^{n} x_{ij} \leq |S| - 1, \ \forall S \subset V, \ 2 \leq |S| \leq n - 1 \tag{1.10}$$

Where S is the set of nodes in the current tour to be verified as a complete tour and not a subtour, and |S| being the cardinality of the set.

There are 2 interpretations of the DFJ constraints-

- The number of arcs between nodes in the subset should be less than the number of nodes in that subset.
- The number of arcs that connect a node from the subset to a node outside of the subset should be at least 2.

1.5 Introducing the problem

The algorithm is based on a variation of the Travelling Salesman Problem for a journey planner, which is NP-complete. The source is the same as the destination, as the tour has to be completed when the user is back at home or at the starting point.

The input locations have to be visited exactly once, like in TSP. Each pair of nodes can have multiple time-dependent links connecting them, each with its own start time and travel time predefined.

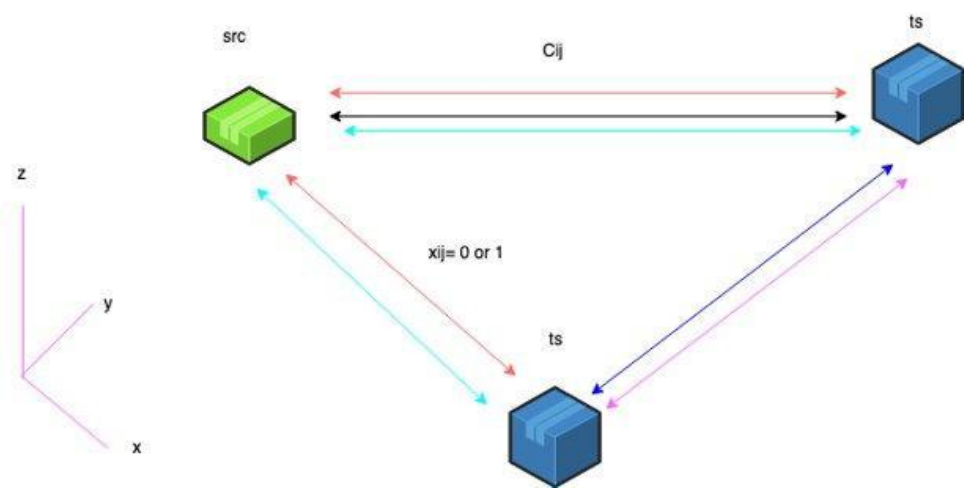
Formally, the problem includes the following:

- Number of nodes as **n**
- Node 1 is set as source and destination
- Number of links between a pair of nodes as **k**
- The time taken to travel on a link in a Cost Matrix $\mathbf{C_{ijk}}$
- The Start time of each link Start Matrix $\mathbf{Start_{ijk}}$
    - Example - Arrive at node x at 1 pm, but trains start from 3 pm, 5 pm, etc.
    - There would be a waiting period involved (explained in the document)
- Time decided to spend at a place as $\mathbf{Buffer_i}$

The cost matrix is 3-Dimensional, with each pair of nodes possibly having more than one link connecting them, each with its independent start time and travel cost.

Shown below is a visualisation of an example, and how a third dimension is added to the network representation containing the set of links joining each pair of nodes

Figure 1.1 3D representation of time-based links connecting nodes in the network

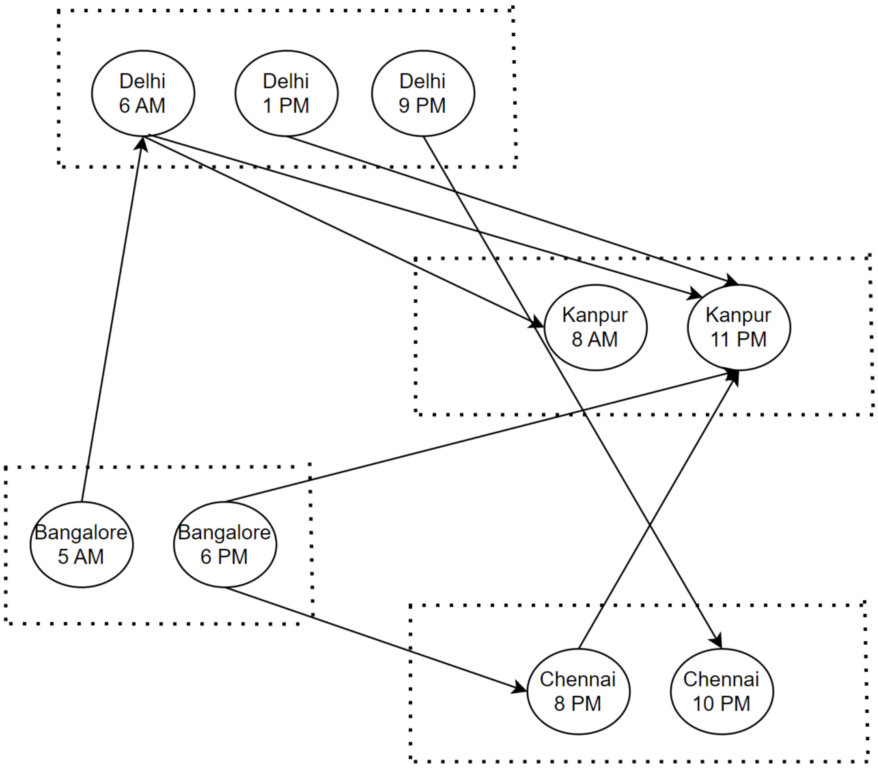


The tour has the provision for waiting or staying at a node for a predefined amount of time. This mimics real-life journeys where a person stays at a place for a certain number of hours or days before moving to the next location or returning back to the start location.

For the second method using a time-expanded transit network, the problem formally includes the following:

- Time-dependent nodes, with each node corresponding to a place and time
- Directed links between pairs of nodes
- The time taken to travel on a link in a Cost Matrix $\mathbf{C}_{\mathbf{ijk}}$
- Time decided to spend at a place as $\mathbf{Buffer}_{\mathbf{i}}$

Figure 1.2 TETN representation of time-based links connecting nodes in the network



This method flattens the graph into 2-Dimensions, avoiding the use of a 3-dimensional cost matrix with each node now corresponding to a geographic place and a specific time.

Chapter 2

3-Dimensional Formulation and Case Study

2.1 Solutions to additions to the base problem

The Journey Planner requires more constraints than the classic TSP does. This is primarily due to the introduction of multiple time-dependent links between each pair of nodes.

The decision variables are:

- x, binary as defined before
- u, for MTZ constraints

The introduction of a new decision variable t, which denotes the time at which the tour is ready to depart from a node, subject to the availability of trains according to the links' start times. This brings its own set of changes, including but not limited to, optimising for the variable t for the destination instead of the collective sum of the path costs as in usual minimum cost routing problems.

A detailed explanation of the decision variable t is as follows:

t_i , corresponding to the exit time of node i, denotes the time when the tour is ready to exit the node before waiting for the connecting train's start times.

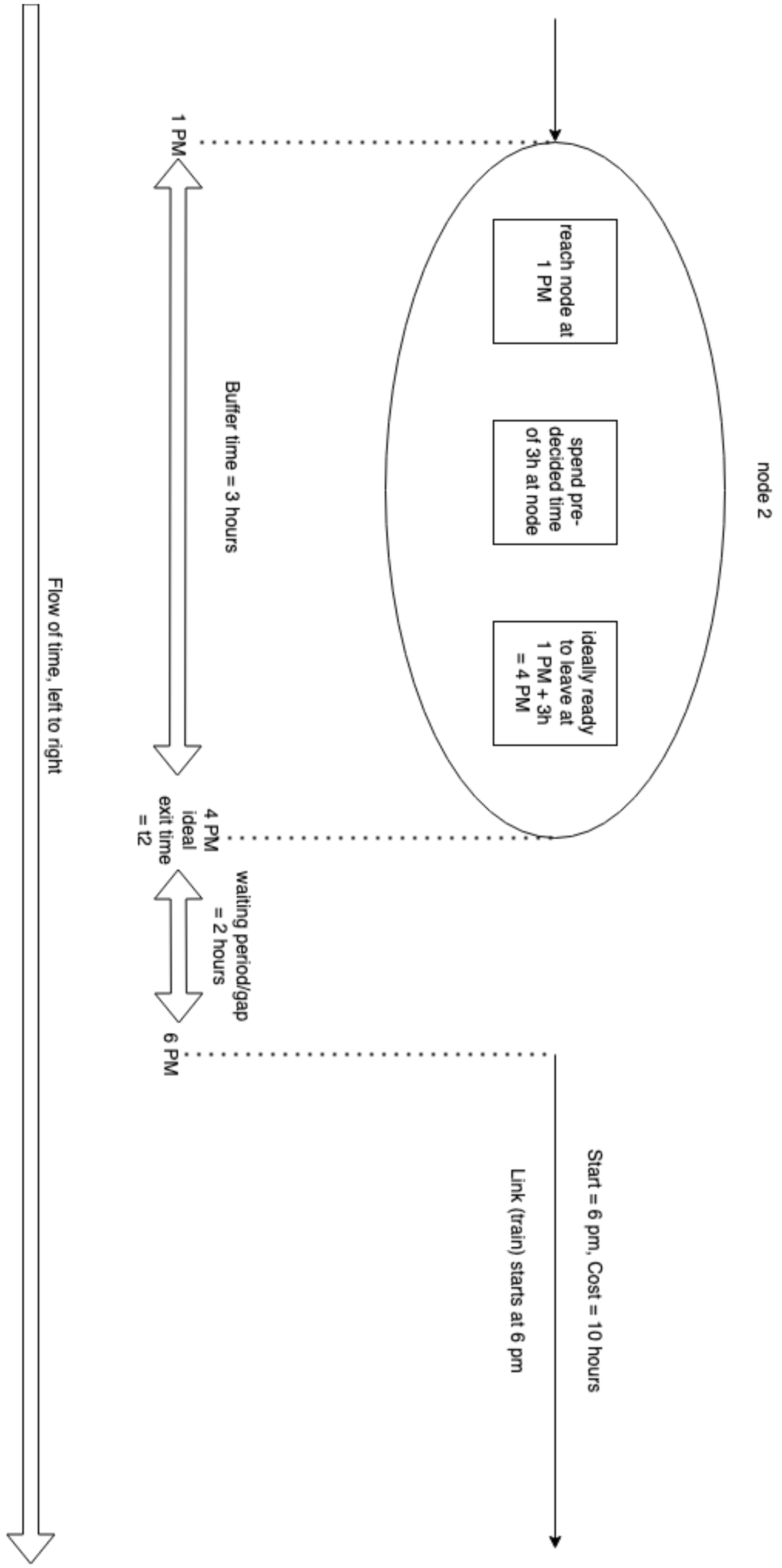
- Eg- The tour arrives at node 2 at 1 pm, and the buffer time is 3 hours. This means the tour is ready to leave the node at 1 pm + 3 hours = 4 pm. Hence, t_2 = 4 pm

- The tour may have to wait after t_i , eg- The next train starts at 6 pm, but t_2 is 4 pm.

  This gives a waiting time of 6 pm - 4 pm = 2 hours.

A diagram representing the idea is shown below

Figure 2.1 Decision variable ***t*** explained



2.2 MTZ Reformulated

The MTZ constraints published in 1960 and perhaps the most sophisticated mathematical formulation of the subtour elimination constraints for TSP, are used in integer programming problems relating to the topic frequently as they were originally formulated.

But the original MTZ constraints are suitable for the 2-Dimensional representation of a network. For the Journey Planner algorithm, the constraints have been modified for a 3-Dimensional network representation to eliminate subtours, 62 years after they were formulated.

2.3 Iterative process

Multiple iterations for finding the perfect formulation for the problem have been discussed with their drawbacks, if any, in the table below

Table 2.1 Iterations, changes and results

Sno	Changes	Result
1	Classic TSP formulated	Tested against n=17, successful
2	Added route visualisation	Displays route in ascending rank, successful
3	Variable t introduced, new time constraints added	Tested on dummy values, inconsistent results
4	The Objective function changed	Consistent results, rank of nodes not continuous, objective function requires refining
5	The Objective function changed, t accommodates n+1 nodes, 1 extra to hold t for source on concluding tour	rank of nodes not consistent, objective function successful
6	62-year-old MTZ constraints modified for 3D binary matrix	ranks successful, time constraints require refactoring
7	Refined the function to compare and find time constraints	Tests successful

2.4 Formulation

The objective function

Minimise **t**[destination]

The consolidated list of constraints-

- Outdegree of each node = 1

  For all nodes i,

$$\Sigma\, x_{ijk} = 1 \text{ (summed over j, k)} \tag{2.1}$$

- Indegree of each node = 1

  For all nodes j,

$$\Sigma\, x_{ijk} = 1 \text{ (summed over i, k)} \tag{2.2}$$

- These constraints cover the constraint of selecting only 1 out of k links between a pair of nodes as well
- MTZ constraint with u

Original for 2D networks-

$$u_i + x_{ijk} <= u_j + (n-1)*(1-x_{ijk}) \tag{2.3}$$

Modified for 3D networks-

$$u_i - u_j - (n-1) <= -n*\Sigma\, x_{ijk} \text{ (summed over k)} \tag{2.4}$$

$$u_1=0 \tag{2.5}$$

Here big M = n-1, as the upper bound on variable u = number of nodes = n

Custom experimental constraints with t, (M=∞/10000)

- Setting an upper bound on t_i

$$t_i <= Start_{ijk}*x_{ijk} + M*(1-x_{ijk}) \tag{2.6}$$

for x_{ijk}=0, if link is not chosen

$$t_i <= 0 + M = M \quad (2.7)$$

for x_{ijk}=1, if link is chosen

$$t_i <= Start_{ijk} + 0 \quad (2.8)$$

- Main constraint for t, inspired by MTZ constraint for u

$$\max(t_i, Start_{ijk}) + Cost_{ijk} * x_{ijk} <= t_j + M*(1-x_{ijk}) \quad (2.9)$$

for x_{ijk}=0, if link is not chosen

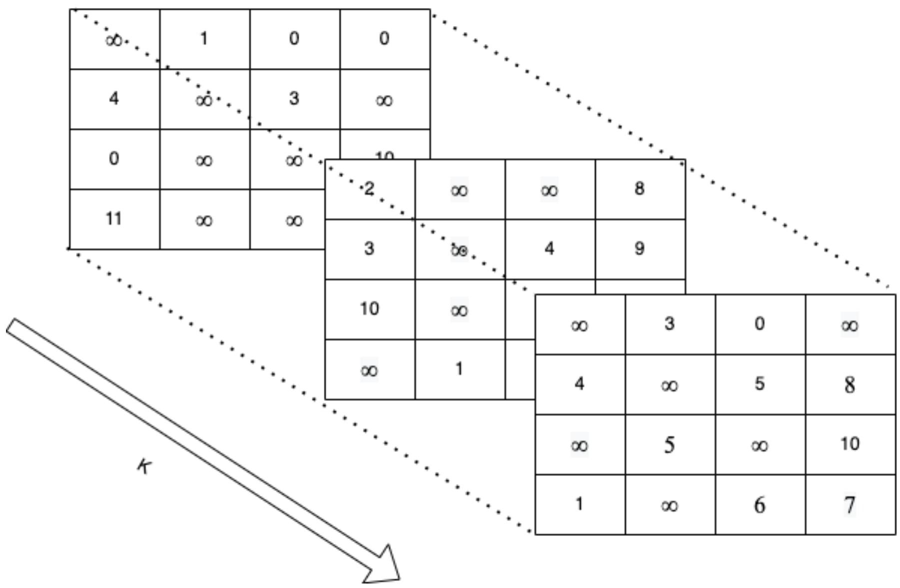
$$\max(t_i, Start_{ijk}) + 0 <= t_j + M \quad (2.10)$$

for x_{ijk}=1, if link is chosen

$$\max(t_i, Start_{ijk}) + Cost_{ijk} <= t_j + 0 \quad (2.11)$$

The 3-Dimensional matrix has varying elements for each time-based link, and thus all constraints have been designed keeping the extra dimensions in mind.

Figure 2.2 3-Dimension matrix visualised



2.5 Results and Conclusion

From the inputs Start Matrix and Cost Matrix, we find the optimal route solution in the form of a binary matrix X, which has a unit entry if the link ijk is chosen and zero otherwise.

The decision variables t and u are as discussed earlier, and denote the time decision variable and the rank of each node in the optimal route respectively.

Example 1

n = 2, k = 2

Table 2.2 Example 1

	K	1			2	
Example 1	Start Matrix	∞	6		∞	8
		13	∞		∞	∞
	Cost Matrix	∞	1		∞	100
		1	∞		∞	∞
	x	0	1		0	0
		1	0		0	0
	t	Node	t value			
		1	0			
		2	7			
		1	14			
	u	Node	u value			
		1	0			
		2	1			

Example 2

n = 3, k = 2

Table 2.3   Example 2

	K	1				2		
Example 2	Start Matrix	∞	0	∞		∞	∞	∞
		∞	∞	6		∞	∞	8
		1000	∞	∞		∞	∞	∞
	Cost Matrix	∞	7	∞		∞	∞	∞
		∞	∞	1		∞	∞	100
		1000	∞	∞		1000	∞	∞
	x	0	1	0		0	0	0
		0	0	0		0	0	1
	t	Node	t value					
		1	0					
		2	8					
		3	108					
		1	2000					
	u	Node	u value					
		1	0					
		2	1					
		3	2					

Example 3

n = 3, k = 2

Subtour resilience check

Table 2.4 Example 3

	K	1				2		
Example 3	Start Matrix	∞	0	∞		∞	∞	∞
		8	∞	6		∞	∞	8
		1000	∞	∞		∞	∞	∞
	Cost Matrix	∞	7	∞		∞	∞	∞
		0	∞	1		∞	∞	100
		1000	∞	∞		∞	∞	∞
	x	0	1	0		0	0	0
		0	0	0		0	0	1
		1	0	0		0	0	0
	t	Node	t value					
		1	0					
		2	8					
		3	108					
		1	2000					
	u	Node	u value					
		1	0					
		2	1					
		3	2					

Example 4

n = 4, k = 3

Table 2.5 Example 4

K	1					2					3			
Start Matrix	∞	1	0	0		∞	∞	∞	∞		∞	∞	∞	∞
	4	∞	3	∞		∞	∞	5	∞		∞	∞	∞	∞
	0	∞	∞	10		∞	∞	∞	10		∞	∞	∞	8
	11	∞	∞	∞		11	∞	∞	∞		∞	∞	∞	∞
Cost Matrix	∞	3	0	0		∞	∞	∞	∞		∞	∞	∞	∞
	0	∞	0	∞		∞	∞	5	∞		∞	∞	∞	∞
	0	∞	∞	1		∞	∞	∞	2		∞	∞	∞	0
	0	∞	∞	∞		100	∞	∞	∞		∞	∞	∞	∞
x	0	1	0	0		0	0	0	0		0	0	0	0
	0	0	0	0		0	0	1	0		0	0	0	0
	0	0	0	1		0	0	0	0		0	0	0	0
	1	0	0	0		0	0	0	0		0	0	0	0

t	Node	t value
	1	0
	2	5
	3	10
	4	11
	1	11
u	Node	u value
	1	0
	2	1
	3	2
	4	3

Testing and verifying the binary solution matrix for x, the decision variable for exit time t, and the decision variable from the modified MTZ constraints for the rank of nodes u, we find the solutions to be exact and error-free. Each example had an increasing level of complexity for the algorithm to solve:

Example 1 started with a base problem involving two nodes to check the algorithm's ability to return the tour to the original node

Example 2 increased the number of nodes by introducing an intermediate node for the directed tour

Example 3 introduced one of the most important obstacles for the algorithm to overcome- the introduction of subtours. The algorithm responded positively to the obstacle and generated a complete tour, passing the test.

Example 4 introduced multiple subtours and increased the complexity of the problem by introducing a four-node system with three maximum possible time-based links between each pair of nodes. The algorithm produced an optimal, complete tour, prohibiting subtour generation in the solution.

The algorithm works as intended and produces the optimal solution for all test cases provided with increasing computational complexity and potential subtour generation, which shows the algorithm's resilience to favouring subtours for a more optimal solution. Instead, the subtour solutions are barred from generation using the modified MTZ constraints for a 3-Dimensional matrix we have built from the original 1963 MTZ constraints. The time constraints provide the optimal solution using the original constraints introduced in the project and work flawlessly.

Chapter 3

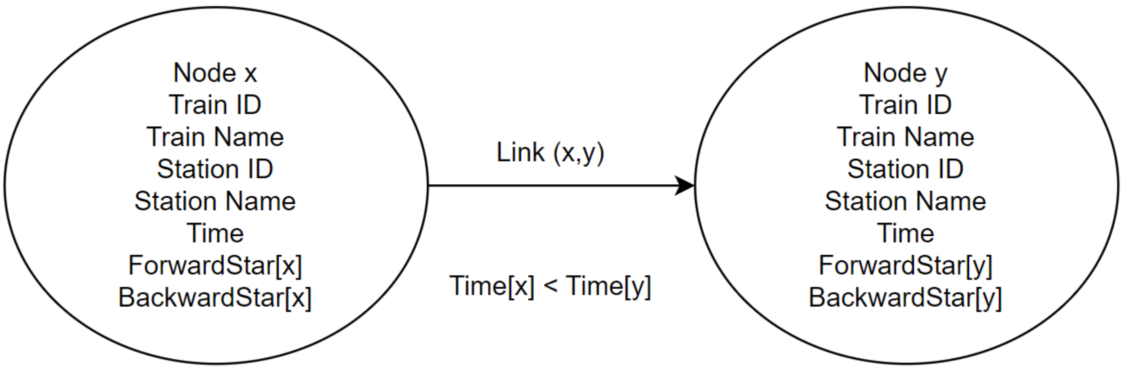
Time-Expanded Transit Network and Case Study

3.1 Expanding the Network through Time

The Journey Planner problem statement remains the same, and the Time-Expanded Transit Network (TETN) solves the problem in an entirely different manner when compared to the 3-Dimensional formulation of the solution.

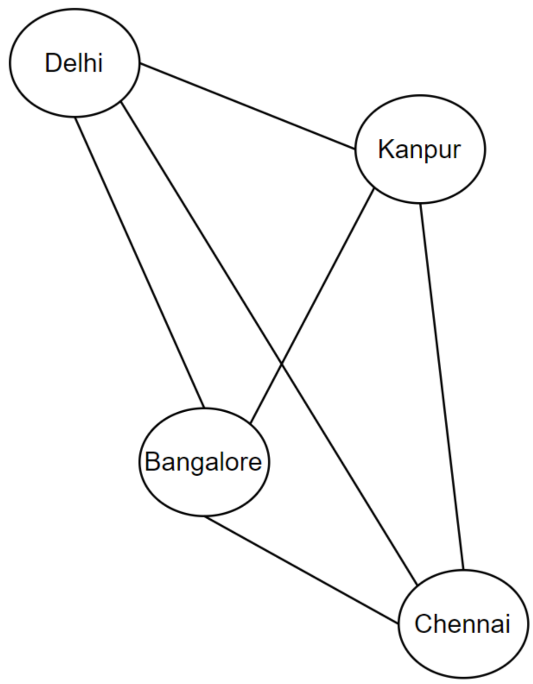
The TETN for journey planning has nodes corresponding to both a physical location (example: Chennai) and a time (example: 10:05 AM). The nodes are connected through directed links with time constraints wherein the link is directed from a node having a lower time value to a higher time value. An example of linking a pair of nodes is given below:

Figure 3.1 Linking Nodes



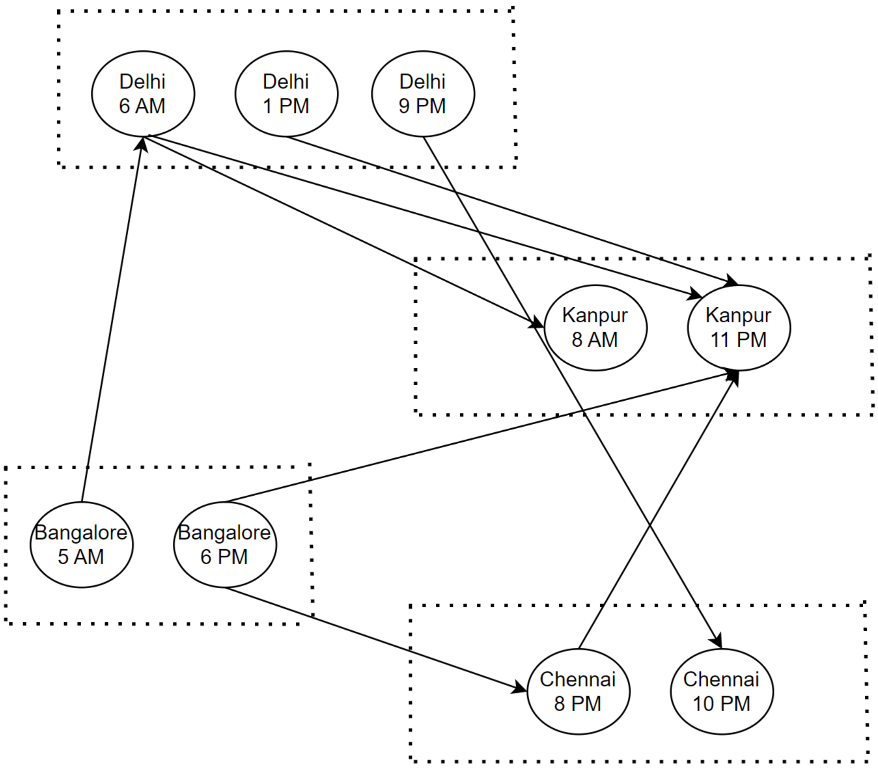
An ordinary network for a select number of nodes would look like the following:

Figure 3.2 Ordinary network representation



The TETN for the same ordinary network looks like

Figure 3.3 TETN representation



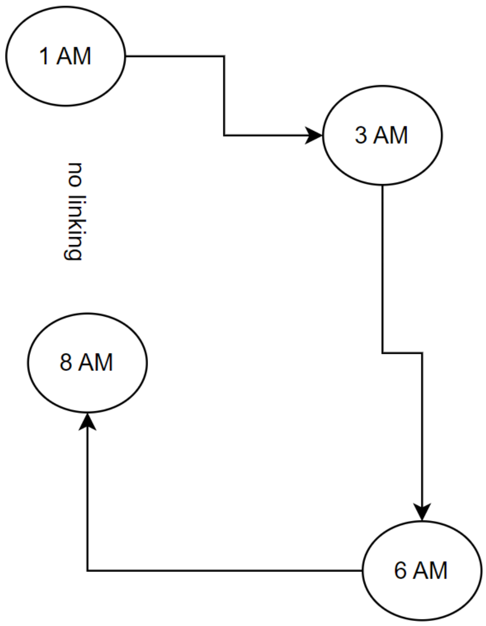
The number of nodes and links increases resulting in a more elaborate network when compared to its ordinary counterpart and the one used in the 3-Dimensional solution approach. The increase in size is offset by the decrease in the number of constraints required to run the algorithm. Each spatial location which used to be represented as a single node is now a set of nodes corresponding to the same spatial location but with different temporal values.

The decision variables are the same as the original TSP, thus eliminating the need for the decision variable ***t*** introduced in the previous chapter. The decision variables are:

- x, binary as defined before
- u, for MTZ constraints

This is because the graph itself holds spatial and temporal information through nodes corresponding to both a physical location and a specific time, and directed links inherently moving towards the positive flow of time. Another property of the network is that it is acyclic. This means that a path emerging from a node cannot enter the same node again as the links from the origin node only connect to nodes with a time value greater than the time value of the origin node.

Figure 3.4 Preventing cycle formation



3.2 Iterative Process

Building the model for the solution has been an iterative process. The major iterations the project has gone through are as follows:

1. Building a sample network

Starting with smaller example networks and visualising to manually check for inconsistencies, the custom example networks were created.

2. Using real-world train data for Indian trains

Using data scraped from the internet and consolidated into a single database, a large network of interconnected train routes was turned into a graph representation. The graph consists of the following final outputs:

- Nodes: a list of all nodes in the graph

- Links: a list of all links in the graph
- Forward Star: for each node, a list of nodes to which the node is connected to
- Backward Star: for each node, a list of nodes which connect to the node

Thus, a database of train schedules has been turned into a TETN

3. Building a model for the classic TSP

An IP optimisation model was built to solve the classic TSP and tested on the example TETN mentioned earlier. Suitable constraints and a cost minimization objective have been set to solve the classic TSP.

4. Building a model for multiple destinations

The TETN has multiple possible destination nodes where the tour returns to. That is, the same physical location as the origin node but with a higher time value.

Thus, the classic TSP approach needs to be modified. In this iteration of the solution, each destination was chosen one at a time and the tour with the lowest travel cost would be chosen as the optimal tour. Referred to in the report as Multiple Destination Approach (MDA) 1.

This solution works well and gives the optimal tour, but takes more time to output the solution as each node is picked from the set of possible destination nodes and the optimal route calculated for it.

5. Multiple destinations revisited

An improvement in the algorithm is done by using the possible destination nods as a set instead of using the nodes individually. This decreases the time taken to arrive at the solution.

6. Major improvement in the algorithm

Extending the use of the set of possible destination nodes, the set of nodes of origin nodes and the set of intermediate nodes to visit mandatorily is used as is, similar to the previous iteration for possible destinations.

3.3 Formulation

The objective function

$$\text{Minimise} \sum_{i,j \in Nodes} C_{ij}x_{ij} \tag{3.1}$$

The consolidated list of constraints:

- Mandatory exit from origin set

$\forall\, o^t \in \{origin\}$

$$\sum_{t \in time\{origin\}} \sum_{j \in FS(o^t)} x_{o^t j} \geq 1 \tag{3.2}$$

- Mandatory exit from intermediate destinations

$\forall\, intermediate\ destinations,\ p^t \in \{intermediate\ destination\}$

$$\sum_{t \in time\{intermediate\ destinations\}} \sum_{i \in BS(p^t)} x_{ip^t} \geq 1 \tag{3.3}$$

- Flow conservation

$\forall\, i \notin \{origin,\ possible\ final\ destinations\}$

$$\sum_{j \in FS(i)} x_{ij} - \sum_{j \in BS(i)} x_{ji} = 0 \tag{3.4}$$

- MTZ constraints to avoid subtour generation

$\forall\, i, j:\ x_{ij} = 1$

$$u_j - u_i \geq 1 \tag{3.5}$$

- Barring entry into origin set

$\forall\, o^t \in\ \{origin\}$

$$\sum_{t \in\, time\{origin\}} \sum_{i \in BS(\, o^t\,)} x_{io^t} \;=\; 0 \tag{3.6}$$

- Barring exit from final destination set

$\forall\, d^t \in\ \{possible\ final\ destinations\}$

$$\sum_{t \in\, time\{possible\ final\ destinations\}} \sum_{j \in FS(\, d^t\,)} x_{d^t j} \;=\; 0 \tag{3.7}$$

- Mandatory entry into final destination set

$\forall\, d^t \in\ \{possible\ final\ destinations\}$

$$\sum_{t \in\, time\{possible\ final\ destinations\}} \sum_{i \in BS(\, d^t\,)} x_{id^t} \;\geq\; 1 \tag{3.8}$$

3.4 Creating the network

The database containing the schedule of Indian trains is enormous. This makes manually checking the network infeasible and resource-intensive. Thus, each step of network creation is planned keeping in mind the smallest details and potential roadblocks that could cause problems in the network.

Details of the database:

237,449 rows, corresponding to nearly 474,898 nodes (each row generates 2 nodes: arrival and departure of a train), and a maximum of 112,763,817,753 links (${}^{n}C_{2}$, where n is the number of nodes) if all nodes are connected, forming a dense network.

An extremely high upper bound on the number of nodes and links could increase the compute time to an extent where it becomes infeasible to obtain a solution considering the resources required to compute it.

3.4.1 Pruning the network

A pruning algorithm is proposed specifically for the Journey Planner and is described as follows:

Let the universal set of train stations contain a certain number of train stations. A majority of the trains would only pass through a single or none of the stations. Such trains are not relevant for solving the problem as they do not connect two or more stations present in the universal set.

Each train in the database is checked for the number of stations present in the universal set it stops at. The train is not included in the network if it does not stop at more than one station present in the universal set of stations.

This prunes the network, decreasing the number of nodes and links significantly and helping find the optimal tour quicker.

3.5 Results and conclusion

The Journey Planner Algorithm, a solution to our problem statement modelled as a variant of the Travelling Salesman Problem, creates a TETN from the railway network database and minimises the total time taken to travel to and from the same spatial location. This is demonstrated by examples MDA 1, MDA 2 and MDA 3 (a), (b).

The algorithm can be used to find the shortest route between two places with the added constraint of travelling via a specific station or location, as is demonstrated by examples MDA 3 (c), (d) in the MDA 3 approach below.

3.5.1 MDA 1

Runtime: 8 seconds

Nodes chosen are in order, top to bottom.

Origin = Chennai at 22:00

Intermediate destinations = Vijayawada at 10:15, Warangal at 19:43

The solution for the optimal tour visits the locations in the following order:

Chennai (origin), Vijayawada, Warangal, Chennai (returns to origin). Refer to Table 4.1 for the detailed tour

Table 3.1 MDA 1 Extended Solution

Node ID	Station ID	Train Name	Time	Station Name
1251	12621	TAMIL NADU SF EXPRESS	22:00	CHENNAI CENTRAL
1252	12621	TAMIL NADU SF EXPRESS	4:10	VIJAYAWADA JUNCTION
1345	12625	KERALA EXPRESS	10:15	VIJAYAWADA JUNCTION
1346	12625	KERALA EXPRESS	13:00	WARANGAL
39	2193	TIRUNELVELI - JABALPUR SF FARE SPECIAL	19:43	WARANGAL
1289	12622	TAMIL NADU SF EXPRESS	20:50	WARANGAL
1290	12622	TAMIL NADU SF EXPRESS	0:15	VIJAYAWADA JUNCTION
1291	12622	TAMIL NADU SF EXPRESS	0:25	VIJAYAWADA JUNCTION
1292	12622	TAMIL NADU SF EXPRESS	7:10	CHENNAI CENTRAL
279	12296	SANGHAMITRA SF EXPRESS	13:30	CHENNAI CENTRAL

3.5.2 MDA 2

Runtime: 13 seconds

Nodes chosen are in order, top to bottom.

Origin = Chennai at 170700

Intermediate destinations = Vijayawada at 209700, Warangal at 247800

The solution for the optimal tour visits the locations in the following order:

Chennai (origin), Vijayawada, Warangal, Chennai (returns to origin). Refer to Table 4.2 for the detailed tour

Table 3.2 MDA 2 Extended Solution

Node ID	Station ID	Train Name	Time	Station Name
5117	22648	THIRUVANANATH APURAM CENTRAL - KORBA SF EXPRESS	170700	CHENNAI CENTRAL
5118	22648	THIRUVANANATH APURAM CENTRAL - KORBA SF EXPRESS	178500	GUDUR JUNCTION
4909	22646	AHILYA NAGARI SF EXPRESS	178620	GUDUR JUNCTION
4910	22646	AHILYA NAGARI SF EXPRESS	180120	NELLORE
1343	12625	KERALA EXPRESS	195780	NELLORE
1344	12625	KERALA EXPRESS	208800	VIJAYAWADA JUNCTION
1345	12625	KERALA EXPRESS	209700	VIJAYAWADA JUNCTION
1346	12625	KERALA EXPRESS	219600	WARANGAL
1289	12622	TAMIL NADU SF EXPRESS	247800	WARANGAL
1290	12622	TAMIL NADU SF EXPRESS	260100	VIJAYAWADA JUNCTION
1291	12622	TAMIL NADU SF EXPRESS	260700	VIJAYAWADA JUNCTION
1292	12622	TAMIL NADU SF EXPRESS	285000	CHENNAI CENTRAL

3.5.3 MDA 3

a)

Origin = New Delhi

Intermediate destinations = Warangal

Final Destion = New Delhi

Runtime: 1 minute 43 seconds

For the complete tour with timings, refer Appendix I, Table A1.1

b)

Origin = Chennai

Intermediate destinations = Vijayawada

Final Destion = Chennai

Runtime: 1 minute 43 seconds

For the complete tour with timings, refer Appendix I, Table A1.2

c)

Origin = New Delhi

Intermediate destinations = Bhopal, Jhansi, Warangal, Vijayawada

Final Destination = Tiruchchirappalli

Runtime: 12 seconds

For the complete tour with timings, refer Appendix I, Table A1.3

d)

Origin = Kanpur

Intermediate Nodes = Bhopal, Jhansi, Warangal, Vijayawada

Final Destination = Tiruchchirappalli

Runtime: 13 seconds

For the complete tour with timings, refer Appendix I, Table A1.4

Chapter 4

Discussion and Conclusions

4.1 Highlights

The two novel solutions proposed approach the problem statement differently and provide the optimal tour as the output. The 3-Dimensional approach for the problem statement uses a cost matrix which builds upon the classic square matrix of size equal to the number of nodes in the networks by adding a third: the number of time-dependent links between two nodes in the network. This adds another dimension to the network representation.

A modification of the MTZ constraints introduced in 1960 has been used for the 3-Dimensional cost matrix and works as intended, preventing the formation of subtours in the optimal solution and providing the rank of each node in the network.

The development of an algorithm for pruning the network shows a significant improvement in the time taken to arrive at the optimal solution. The pruning algorithm, as described before, helps make the network smaller by decreasing the number of nodes and links which do not contribute towards solving the problem.

The TETN approach for the problem statement builds upon the idea of using time-dependent links between two nodes but goes one step further: flattening the network back into two dimensions, wherein the nodes hold both spatial and temporal values and the links are directed towards the positive flow of time, thereby eliminating the possibility of cycles forming in the network.

Each approach has gone through multiple iterations which have been described in their respective chapters, improving the runtime, accuracy and aligning with the final desired output from a solution for the Journey Planning problem statement.

4.2 Limitations and Future Scope

The constraints for the time-expanded transit network approach are designed such that the set of origin and possible final destination nodes are mutually exclusive with reference to time if the sets correspond to the same physical location. This can be improved further by modifying the constraints to accommodate using a single set to replace the two sets described.

The solution can be extended to find the optimal route based on different metrics instead of finding using the shortest path approach. Some of these include finding the most monetarily cost-efficient path by minimising the monetary cost of travelling on a link and finding the most reliable path using the variance for each train based on past data.

Adding or modifying the objective function and the constraints to a small extent can help solve various related problems using the network built using the data for the schedule for trains in India.

4.3 Conclusion

Exploring two novel methods for solving the Journey Planning problem statement, we find that the variant of the Travelling Salesman Problem can be approached in multiple ways, each with its method of network representation, set of constraints and models.

In this report, we explored a 3-Dimensional network representation and a time-expanded network representation for the problem statement and explained in detail the components of the solutions and how they compare against the baseline naïve brute-force solution.

Testing against real-world data on Indian train schedules, the solution works well and gives consistent results. This is a promising endeavour and can be extended to solving problems with more complex objective functions.

The time taken to solve the Journey Planning problem for multiple test cases has been reduced drastically when compared to classic graph algorithms and their runtimes for the original travelling salesman problem, making it possible to solve it within a realistic time frame for a large network as demonstrated.

From drawing inspiration from the tried and tested MTZ constraints from 1960 and modifying them for a 3-Dimensional cost matrix, to converting actual train data into a time-expanded network and pruning it, the project covers everything down to the minuscule details on how to model the solutions for the journey planning algorithm.

The algorithm works as intended and produces the optimal solution for all test cases provided and shows the algorithm's resilience to favouring subtours for a more optimal solution. Instead, the subtour solutions are barred from generation using the modified MTZ constraints for a 3-Dimensional matrix we have built from the original 1963 MTZ constraints, and the original MTZ constraints work well for the time-expanded network representation.

References

1 **C. E. Miller**, **A. W. Tucker**, and **R. A. Zemlin** (1960) Integer Programming Formulation of Traveling Salesman Problems. *J. ACM* 7, 326–329

2 **Dantzig, G., Fulkerson, R.,** & **Johnson, S**. (1954). Solution of a Large-Scale Traveling-Salesman Problem. *Journal of the Operations Research Society of America*, 2(4), 393–410

3 **Menger, K.** (1932) Das Botenproblem. *Ergebnisse eines Mathematischen Kolloquiums*, 11-12

4 **Larrañaga, P., Kuijpers, C., Murga, R.** et al. (1999) Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. *Artificial Intelligence Review* 13, 129–170

5 **Dorigo, M**. and **Gambardella, L.C.** (1997) Ant colonies for the travelling salesman problem, *Biosystems* 43, 73-81

6 **Gavish, B.** and **Graves, S.C.** (1978) The Travelling Salesman Problem and Related Problems, *Massachusetts Institute of Technology Operations Research Center Working Paper;OR 078-78*, 4-11

7 **Christofides, N.** (1976) Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem. *Oper. Res. Forum* 3, 20

8 **Pataki, G.** (2003) Teaching Integer Programming Formulations Using the Traveling Salesman Problem, *SIAM Review* 45:1, 116-123

9 **K. Ilavarasi** and **K. S. Joseph** (2014) Variants of travelling salesman problem: A survey. *International Conference on Information Communication and Embedded Systems (ICICES2014)*, 1-7

10 **Ahuja, R.K., J. Orlin** and **T. Magnanti** *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Upper Saddle River, 1993

11 **Hillier F.S.** and **J.L. Gerald** *Introduction to Operations Research*, McGraw-Hill, New York, 2001

12 **Matthew R. Silver** and **Olivier L. de Weck** (2007) *Time-expanded decision networks: A framework for designing evolvable complex systems*, AIAA-2006-6964, 11th AIAA/ISSMO Multidisciplinary Analysis Optim Conf, Portsmouth, VA

APPENDIX 1

TETN DETAILED OPTIMAL SOLUTION

Table A1.1 MDA 3 Solution A

Node ID	Station ID	Train Name	Time	Station Name
1176	12616	GRAND TRUNK	153600	NEW DELHI
1177	12616	GRAND TRUNK	160800	MATHURA JUNCTION
1178	12616	GRAND TRUNK	161100	MATHURA JUNCTION
1179	12616	GRAND TRUNK	163140	RAJA KI MANDI
1180	12616	GRAND TRUNK	163260	RAJA KI MANDI
1181	12616	GRAND TRUNK	164100	AGRA CANTT.
1182	12616	GRAND TRUNK	164400	AGRA CANTT.
1183	12616	GRAND TRUNK	167280	DHOLPUR JUNCTION
1184	12616	GRAND TRUNK	167400	DHOLPUR JUNCTION
1185	12616	GRAND TRUNK	168900	MORENA
1186	12616	GRAND TRUNK	169020	MORENA
1187	12616	GRAND TRUNK	170760	GWALIOR JUNCTION
1188	12616	GRAND TRUNK	171060	GWALIOR JUNCTION
1189	12616	GRAND TRUNK	176100	JHANSI JUNCTION
1190	12616	GRAND TRUNK	176820	JHANSI JUNCTION
1191	12616	GRAND TRUNK	184500	BINA JUNCTION
1192	12616	GRAND TRUNK	184800	BINA JUNCTION
1193	12616	GRAND TRUNK	186720	GANJ BASODA
1194	12616	GRAND TRUNK	186840	GANJ BASODA
1195	12616	GRAND TRUNK	188580	VIDISHA
1196	12616	GRAND TRUNK	188700	VIDISHA
1197	12616	GRAND TRUNK	192000	BHOPAL JUNCTION
1198	12616	GRAND TRUNK	192300	BHOPAL JUNCTION
1199	12616	GRAND TRUNK	193080	BHOPAL HABIBGANJ
1200	12616	GRAND TRUNK	193200	BHOPAL HABIBGANJ
1201	12616	GRAND TRUNK	196680	HOSHANGABAD
1202	12616	GRAND TRUNK	196800	HOSHANGABAD
1203	12616	GRAND TRUNK	198900	ITARSI JUNCTION

1204	12616	GRAND TRUNK	199500	ITARSI JUNCTION
1205	12616	GRAND TRUNK	203280	GHORADONGRI
1206	12616	GRAND TRUNK	203400	GHORADONGRI
1207	12616	GRAND TRUNK	206100	BETUL
1208	12616	GRAND TRUNK	206280	BETUL
1209	12616	GRAND TRUNK	207720	AMLA JUNCTION
1210	12616	GRAND TRUNK	207900	AMLA JUNCTION
1211	12616	GRAND TRUNK	211260	PANDHURNA
1212	12616	GRAND TRUNK	211380	PANDHURNA
1213	12616	GRAND TRUNK	212340	NARKHER JUNCTION
1214	12616	GRAND TRUNK	212460	NARKHER JUNCTION
1215	12616	GRAND TRUNK	217200	NAGPUR JUNCTION
1216	12616	GRAND TRUNK	217800	NAGPUR JUNCTION
1217	12616	GRAND TRUNK	221520	SEWAGRAM JUNCTION
1218	12616	GRAND TRUNK	221640	SEWAGRAM JUNCTION
1219	12616	GRAND TRUNK	223320	HINGANGHAT
1220	12616	GRAND TRUNK	223440	HINGANGHAT
1221	12616	GRAND TRUNK	227280	CHANDRAPUR
1222	12616	GRAND TRUNK	227460	CHANDRAPUR
1223	12616	GRAND TRUNK	230700	BALHARSHAH JUNCTION
1224	12616	GRAND TRUNK	231300	BALHARSHAH JUNCTION
1225	12616	GRAND TRUNK	233640	SIRPUR KAGHAZNAGAR
1226	12616	GRAND TRUNK	233700	SIRPUR KAGHAZNAGAR
1227	12616	GRAND TRUNK	235860	BELAMPALLI
1228	12616	GRAND TRUNK	235920	BELAMPALLI
1229	12616	GRAND TRUNK	236880	MANCHERIAL
1230	12616	GRAND TRUNK	236940	MANCHERIAL
1231	12616	GRAND TRUNK	237540	RAMAGUNDAM
1232	12616	GRAND TRUNK	237600	RAMAGUNDAM
1233	12616	GRAND TRUNK	243000	WARANGAL
5265	16787	TEN JAMMU EXP	243900	WARANGAL
5266	16787	TEN JAMMU EXP	248700	RAMGUNDAM
5267	16787	TEN JAMMU EXP	248760	RAMGUNDAM
5268	16787	TEN JAMMU EXP	257400	BALHARSHAH
1963	12687	MADURAI - DEHRADUN SF EXPRESS	261600	BALHARSHAH JUNCTION

1964	12687	MADURAI - DEHRADUN SF EXPRESS	268500	SEWAGRAM JUNCTION
2023	12687-Slip	MADURAI - CHANDIGARH SF EXPRESS	268560	SEWAGRAM JUNCTION
2024	12687-Slip	MADURAI - CHANDIGARH SF EXPRESS	272400	NAGPUR JUNCTION
2025	12687-Slip	MADURAI - CHANDIGARH SF EXPRESS	273000	NAGPUR JUNCTION
2026	12687-Slip	MADURAI - CHANDIGARH SF EXPRESS	294000	BHOPAL JUNCTION
1969	12687	MADURAI - DEHRADUN SF EXPRESS	294600	BHOPAL JUNCTION
1970	12687	MADURAI - DEHRADUN SF EXPRESS	309600	JHANSI JUNCTION
1971	12687	MADURAI - DEHRADUN SF EXPRESS	310320	JHANSI JUNCTION
1972	12687	MADURAI - DEHRADUN SF EXPRESS	315000	GWALIOR JUNCTION
1973	12687	MADURAI - DEHRADUN SF EXPRESS	315300	GWALIOR JUNCTION
1974	12687	MADURAI - DEHRADUN SF EXPRESS	321900	AGRA CANTT.
1975	12687	MADURAI - DEHRADUN SF EXPRESS	322200	AGRA CANTT.
1976	12687	MADURAI - DEHRADUN SF EXPRESS	333600	HAZRAT NIZAMUDDIN
3957	16687	NAVYUG EXPRESS	335400	HAZRAT NIZAMUDDIN
3958	16687	NAVYUG EXPRESS	337200	NEW DELHI

Table A1.2 MDA 3 Solution B

Node ID	Station ID	Train Name	Time	Station Name
164	12295	SANGHAMITRA SF EXPRESS	143100	CHENNAI CENTRAL
165	12295	SANGHAMITRA SF EXPRESS	151380	GUDUR JUNCTION
2497	12969	COIMBATORE - JAIPUR EXPRESS	157920	GUDUR JUNCTION
2498	12969	COIMBATORE - JAIPUR EXPRESS	159420	NELLORE
2361	12967	CHENNAI-JAIPUR SF EXPRESS	159480	NELLORE
2362	12967	CHENNAI-JAIPUR SF EXPRESS	173400	VIJAYAWADA JUNCTION
2275	12760	CHARMINAR SF EXPRESS	177000	VIJAYAWADA JUNCTION
2276	12760	CHARMINAR SF EXPRESS	178620	TENALI JUNCTION
2277	12760	CHARMINAR SF EXPRESS	178680	TENALI JUNCTION
2278	12760	CHARMINAR SF EXPRESS	181200	CHIRALA
2279	12760	CHARMINAR SF EXPRESS	181260	CHIRALA
2280	12760	CHARMINAR SF EXPRESS	183840	ONGOLE
2281	12760	CHARMINAR SF EXPRESS	183900	ONGOLE
2282	12760	CHARMINAR SF EXPRESS	186540	KAVALI
2283	12760	CHARMINAR SF EXPRESS	186600	KAVALI
2284	12760	CHARMINAR SF EXPRESS	188580	NELLORE
2285	12760	CHARMINAR SF EXPRESS	188640	NELLORE
2286	12760	CHARMINAR SF EXPRESS	193080	GUDUR JUNCTION
2287	12760	CHARMINAR SF EXPRESS	193200	GUDUR JUNCTION
2288	12760	CHARMINAR SF EXPRESS	194580	NAYADUPETA
2289	12760	CHARMINAR SF EXPRESS	194700	NAYADUPETA
2290	12760	CHARMINAR SF EXPRESS	195780	SULLURUPETA
2291	12760	CHARMINAR SF EXPRESS	195900	SULLURUPETA
2292	12760	CHARMINAR SF EXPRESS	202500	CHENNAI CENTRAL

Table A1.3 MDA 3 Solution C

Node ID	Station ID	Train Name	Time	Station Name
1273	12622	TAMIL NADU SF EXPRESS	167400	NEW DELHI
1274	12622	TAMIL NADU SF EXPRESS	176400	AGRA CANTT.
1275	12622	TAMIL NADU SF EXPRESS	176580	AGRA CANTT.
1276	12622	TAMIL NADU SF EXPRESS	181920	GWALIOR JUNCTION
1277	12622	TAMIL NADU SF EXPRESS	182100	GWALIOR JUNCTION
1278	12622	TAMIL NADU SF EXPRESS	187080	JHANSI JUNCTION
1279	12622	TAMIL NADU SF EXPRESS	187800	JHANSI JUNCTION
1280	12622	TAMIL NADU SF EXPRESS	201000	BHOPAL JUNCTION
1281	12622	TAMIL NADU SF EXPRESS	201600	BHOPAL JUNCTION
1282	12622	TAMIL NADU SF EXPRESS	207600	ITARSI JUNCTION
1283	12622	TAMIL NADU SF EXPRESS	207900	ITARSI JUNCTION
1284	12622	TAMIL NADU SF EXPRESS	223500	NAGPUR JUNCTION
1285	12622	TAMIL NADU SF EXPRESS	223800	NAGPUR JUNCTION
1286	12622	TAMIL NADU SF EXPRESS	235800	BALHARSHAH JUNCTION
1287	12622	TAMIL NADU SF EXPRESS	236400	BALHARSHAH JUNCTION
1288	12622	TAMIL NADU SF EXPRESS	247680	WARANGAL
1289	12622	TAMIL NADU SF EXPRESS	247800	WARANGAL
1290	12622	TAMIL NADU SF EXPRESS	260100	VIJAYAWADA JUNCTION
2475	12968	JAIPUR-CHENNAI SF EXPRESS	267600	VIJAYAWADA JUNCTION
2476	12968	JAIPUR-CHENNAI SF EXPRESS	278400	NELLORE
2615	12970	JAIPUR - COIMBATORE EXPRESS	278460	NELLORE
2616	12970	JAIPUR - COIMBATORE EXPRESS	282660	GUDUR JUNCTION

2741	15120	MANDUADIH - RAMESWARAM WEEKLY EXPRESS	287700	GUDUR JUNCTION
2742	15120	MANDUADIH - RAMESWARAM WEEKLY EXPRESS	297300	CHENNAI EGMORE
2743	15120	MANDUADIH - RAMESWARAM WEEKLY EXPRESS	298200	CHENNAI EGMORE
2744	15120	MANDUADIH - RAMESWARAM WEEKLY EXPRESS	301380	CHENGALPATTU JUNCTION
2745	15120	MANDUADIH - RAMESWARAM WEEKLY EXPRESS	301500	CHENGALPATTU JUNCTION
2746	15120	MANDUADIH - RAMESWARAM WEEKLY EXPRESS	306600	VILLUPURAM JUNCTION
2747	15120	MANDUADIH - RAMESWARAM WEEKLY EXPRESS	306900	VILLUPURAM JUNCTION
2748	15120	MANDUADIH - RAMESWARAM WEEKLY EXPRESS	309840	CUDDALORE PORT JUNCTION
2749	15120	MANDUADIH - RAMESWARAM WEEKLY EXPRESS	309900	CUDDALORE PORT JUNCTION
2750	15120	MANDUADIH - RAMESWARAM WEEKLY EXPRESS	311580	CHIDAMBARAM
2751	15120	MANDUADIH - RAMESWARAM WEEKLY EXPRESS	311700	CHIDAMBARAM
2752	15120	MANDUADIH - RAMESWARAM WEEKLY EXPRESS	312840	SIRKAZHI
2753	15120	MANDUADIH - RAMESWARAM WEEKLY EXPRESS	312900	SIRKAZHI
2754	15120	MANDUADIH - RAMESWARAM WEEKLY EXPRESS	314280	MAYILADUTURAI JUNCTION

2755	15120	MANDUADIH - RAMESWARAM WEEKLY EXPRESS	314400	MAYILADUTURAI JUNCTION
2756	15120	MANDUADIH - RAMESWARAM WEEKLY EXPRESS	316980	KUMBAKONAM
2757	15120	MANDUADIH - RAMESWARAM WEEKLY EXPRESS	317100	KUMBAKONAM
2758	15120	MANDUADIH - RAMESWARAM WEEKLY EXPRESS	319080	THANJAVUR JUNCTION
2759	15120	MANDUADIH - RAMESWARAM WEEKLY EXPRESS	319200	THANJAVUR JUNCTION
2760	15120	MANDUADIH - RAMESWARAM WEEKLY EXPRESS	323400	TIRUCHCHIRAPPA LLI JUNCTION

Table A1.4 MDA 3 Solution D

Node ID	Station ID	Train Name	Time	Station Name
608	12521	RAPTI SAGAR SF EXPRESS	220920	KANPUR CENTRAL
609	12521	RAPTI SAGAR SF EXPRESS	224640	POKHRAYAN
610	12521	RAPTI SAGAR SF EXPRESS	224760	POKHRAYAN
611	12521	RAPTI SAGAR SF EXPRESS	226800	ORAI
612	12521	RAPTI SAGAR SF EXPRESS	227100	ORAI
613	12521	RAPTI SAGAR SF EXPRESS	233100	JHANSI JUNCTION
3825	16318-Slip	HIMSAGAR EXPRESS SLIP	251100	JHANSI JUNCTION
3826	16318-Slip	HIMSAGAR EXPRESS SLIP	265200	BHOPAL JUNCTION
3679	16318	HIMSAGAR EXPRESS	265800	BHOPAL JUNCTION
3680	16318	HIMSAGAR EXPRESS	271800	ITARSI JUNCTION
3829	16318-Slip	HIMSAGAR EXPRESS SLIP	272100	ITARSI JUNCTION
3830	16318-Slip	HIMSAGAR EXPRESS SLIP	289800	NAGPUR JUNCTION
3831	16318-Slip	HIMSAGAR EXPRESS SLIP	290400	NAGPUR JUNCTION
3832	16318-Slip	HIMSAGAR EXPRESS SLIP	293940	SEWAGRAM JUNCTION
3833	16318-Slip	HIMSAGAR EXPRESS SLIP	294060	SEWAGRAM JUNCTION
3834	16318-Slip	HIMSAGAR EXPRESS SLIP	299880	CHANDRAPUR
3835	16318-Slip	HIMSAGAR EXPRESS SLIP	300000	CHANDRAPUR
3836	16318-Slip	HIMSAGAR EXPRESS SLIP	302400	BALHARSHAH JUNCTION
3837	16318-Slip	HIMSAGAR EXPRESS SLIP	303000	BALHARSHAH JUNCTION
3838	16318-Slip	HIMSAGAR EXPRESS SLIP	309000	RAMAGUNDAM
3839	16318-Slip	HIMSAGAR EXPRESS SLIP	309120	RAMAGUNDAM
3840	16318-Slip	HIMSAGAR EXPRESS SLIP	315180	WARANGAL
3841	16318-Slip	HIMSAGAR EXPRESS SLIP	315480	WARANGAL
3842	16318-Slip	HIMSAGAR EXPRESS SLIP	320760	KHAMMAM

3843	16318-Slip	HIMSAGAR EXPRESS SLIP	320880	KHAMMAM
3844	16318-Slip	HIMSAGAR EXPRESS SLIP	327900	VIJAYAWADA JUNCTION
3845	16318-Slip	HIMSAGAR EXPRESS SLIP	328800	VIJAYAWADA JUNCTION
3846	16318-Slip	HIMSAGAR EXPRESS SLIP	330360	TENALI JUNCTION
3847	16318-Slip	HIMSAGAR EXPRESS SLIP	330480	TENALI JUNCTION
3848	16318-Slip	HIMSAGAR EXPRESS SLIP	335460	ONGOLE
3849	16318-Slip	HIMSAGAR EXPRESS SLIP	335580	ONGOLE
3850	16318-Slip	HIMSAGAR EXPRESS SLIP	340020	NELLORE
3851	16318-Slip	HIMSAGAR EXPRESS SLIP	340140	NELLORE
3852	16318-Slip	HIMSAGAR EXPRESS SLIP	344460	GUDUR JUNCTION
3705	16318	HIMSAGAR EXPRESS	344580	GUDUR JUNCTION
3706	16318	HIMSAGAR EXPRESS	348900	RENIGUNTA JUNCTION
3855	16318-Slip	HIMSAGAR EXPRESS SLIP	349500	RENIGUNTA JUNCTION
3856	16318-Slip	HIMSAGAR EXPRESS SLIP	350400	TIRUPATI
3857	16318-Slip	HIMSAGAR EXPRESS SLIP	350520	TIRUPATI
3858	16318-Slip	HIMSAGAR EXPRESS SLIP	354780	CHITTOOR
3859	16318-Slip	HIMSAGAR EXPRESS SLIP	354900	CHITTOOR
3860	16318-Slip	HIMSAGAR EXPRESS SLIP	358380	KATPADI JUNCTION
3713	16318	HIMSAGAR EXPRESS	358500	KATPADI JUNCTION
3714	16318	HIMSAGAR EXPRESS	363780	JOLARPETTAI JUNCTION
3715	16318	HIMSAGAR EXPRESS	363900	JOLARPETTAI JUNCTION
3716	16318	HIMSAGAR EXPRESS	369300	SALEM JUNCTION
3717	16318	HIMSAGAR EXPRESS	369600	SALEM JUNCTION
3718	16318	HIMSAGAR EXPRESS	374400	ERODE JUNCTION
3867	16318-Slip	HIMSAGAR EXPRESS SLIP	378300	ERODE JUNCTION
3868	16318-Slip	HIMSAGAR EXPRESS SLIP	381180	KARUR JUNCTION

3869	16318-Slip	HIMSAGAR EXPRESS SLIP	381300	KARUR JUNCTION
3870	16318-Slip	HIMSAGAR EXPRESS SLIP	387600	TIRUCHCHIRAPPALLI JUNCTION