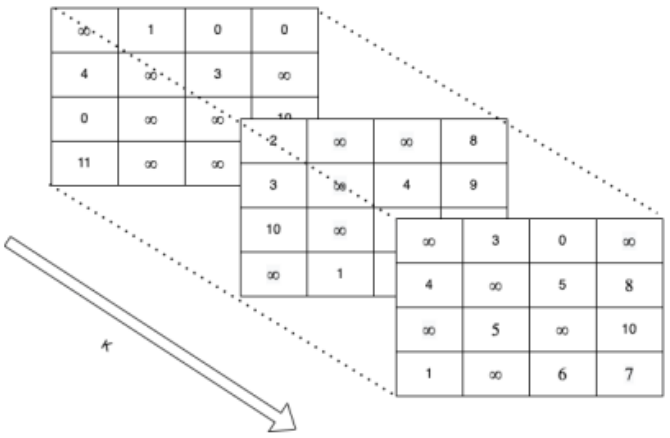
DEPARTMENT OF CIVIL ENGINEERING
INDIAN INSTITUTE OF
TECHNOLOGY
MADRAS
CHENNAI-600 036

Journey Planner Algorithm



A Report

Submitted by

Sayash Raaj Hiraou

November, 2022

Acknowledgments

I would like to acknowledge and give my thanks to my guide, Professor Bhargava Rama Chilukuri, 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

KEYWORDS: Graph, Routing, Linear Programming, Algorithms, Journey Planner

Journey planning involves a significant time investment when done manually, finding the optimal route or plan to follow in order to minimise travel time without compromising on the pre-decided time to be spent at each location.

The problem can be formulated as a graph routing problem, with a certain objective minimising the travel time and trains as links for commuting between locations.
A key feature of the problem statement is the usage of multiple links among each pair of nodes, which stands out from the usual representation of graphs wherein nodes are pairwise linked with at most a singular link.

The algorithm would choose time-based links while optimising, with each link having a start time and a travel time, visualising the graph as a 3-Dimensional network, unlike the 2-Dimensional network representation for most graphs. A linear programming approach has been proposed, including bounds and constraints imposed on the decision variables, all of which has been discussed in detail in the report.

A modified version of the 62 year-old Miller, Tucker, Zemlin constraints (described in the report) for a 3-Dimensional matrix representation of a network has been developed independently. 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.

TABLE OF CONTENTS

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

Notation

- ∞ - Infinity, or a large value to denote infeasibility of a link cost
- **n** - Number of nodes
- **k** - Number of links between
- $\mathbf{C}_{ijk}$ - Cost of travel of each link
- $\mathbf{Start}_{ijk}$ - Start time of each link
- $\mathbf{Buffer}_{i}$ - Time decided to stay/spend at a node/place
- $\mathbf{x}_{ijk}$ - Binary, 0 if link is not selected, 1 otherwise
- $\mathbf{u}_{i}$ - Rank of node, MTZ constraints
- $\mathbf{t}_{i}$ - Time when one is ready to exit a node, prior to waiting for a link. 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

Routing: Issues, methodologies and limitations

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 which 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

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 table).

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 other algorithmic solutions, some of them discussed and referenced in this project.

Chapter 2

The problem defined

2.1 Introducing the problem

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

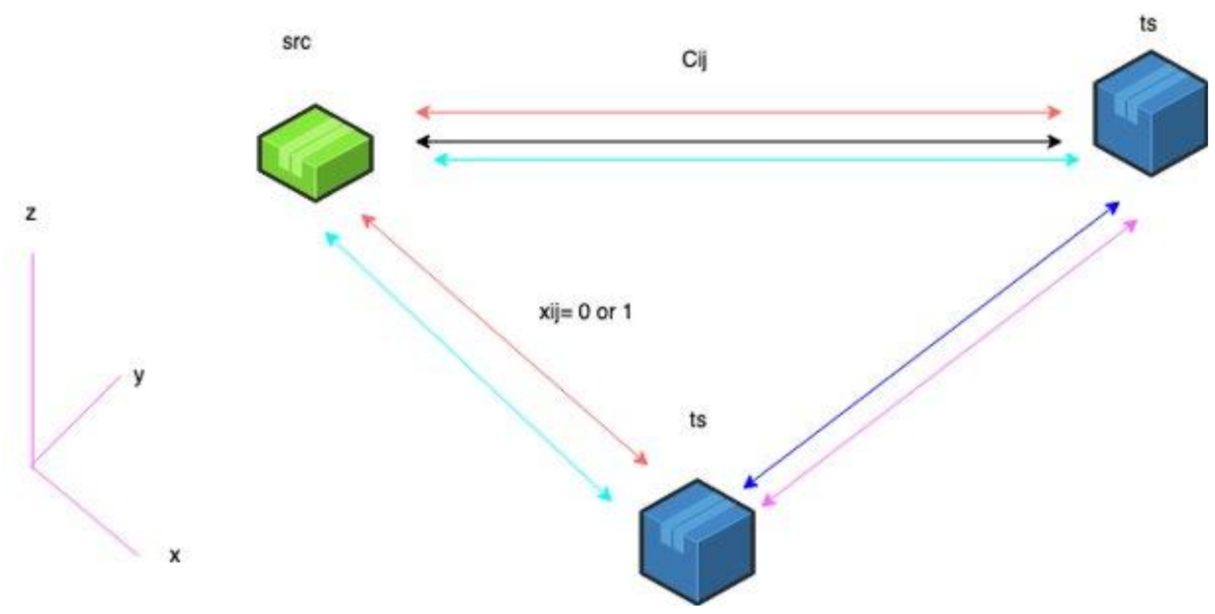
Formally, the problem includes the following:

- 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 2.1 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.

2.2 Potential solutions

To solve this problem, we looked at multiple potential solutions. From 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 with accuracy of the final solution
B. Classic cost minimisation algorithms and their variants, most of which are resource-intensive
C. Linear 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.

2.3 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 for formulating the problem and the constraints to use, including

- Network Flows- Ahuja, Magnanti et al.- Formulating TSP with an extra decision variable
- Introduction to operations research- Hillier, Lieberman.- Towards operations research and a manual method for solving TSP

2.4 The TSP Variant

A linear 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 linear programming approaches for the classic TSP include formulating the constraints in primarily in two ways-

- MTZ constraints
- DFJ constraints

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

$$\min \sum_{i \neq j}^{n} C_{ij} \cdot x_{ij} \tag{2.1}$$

$$\forall\, i \leq n \quad \sum_{j=1}^{n} x_{ij} = 1 \tag{2.2}$$

$$\forall\, j \leq n \quad \sum_{i=1}^{n} x_{ij} = 1 \tag{2.3}$$

defining x_{ij} as a binary variable-

- 1 if the link is chosen

- 0 if the link is not chosen

$$\forall\, i,j \leq n \quad x_{ij} \in \{0,1\} \tag{2.4}$$

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 linear 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 2.1 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 2.2 Classic TSP Problem, 17 nodes Solved

Node	Rank
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

2.4.1 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.

$$\forall\ i \neq j \neq 1, \quad u_i - u_j + nx_{ij} \leq n - 1 \tag{2.5}$$

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,

$$u_i - u_j + n <= n - 1 \qquad (2.6)$$

Simplified,

$$u_i + 1 <= u_j \qquad (2.7)$$

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,

$$u_i - u_j + 0 <= n - 1 \qquad (2.8)$$

Simplified,

$$u_i - u_j <= n - 1 \qquad (2.9)$$

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 linear programming to offset one side of an inequality.

2.4.2 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.

$$\sum_{i,j \in S}^{n} x_{ij} \leq |S| - 1, \quad \forall\, S \subset V, \quad 2 \leq |S| \leq n - 1 \qquad (2.10)$$

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.

Chapter 3

Solving the variant

3.1 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 being:

- 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:

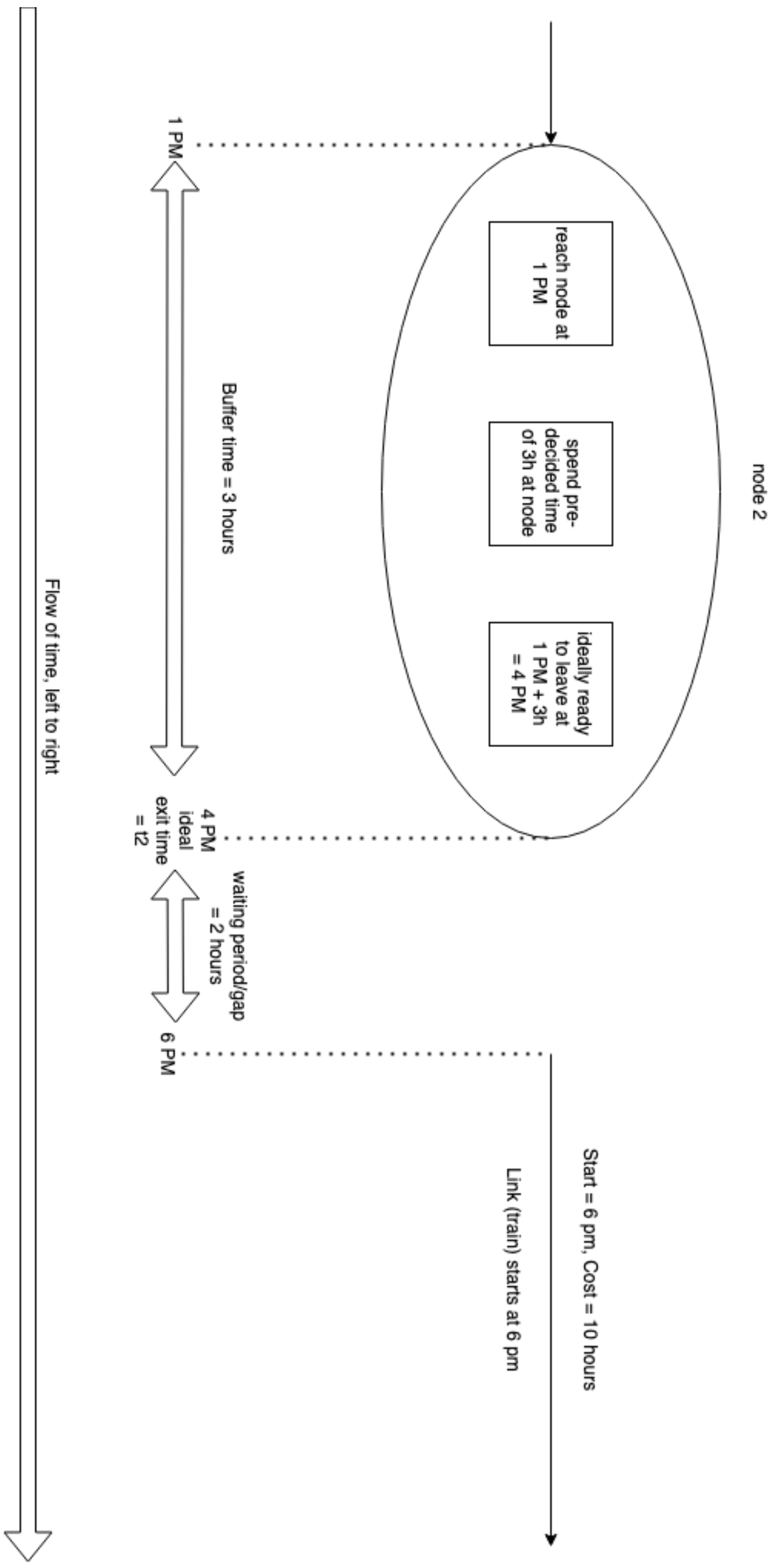
t_i , corresponding to the exit time of node i, denotes the time when the tour is ready to exit node before waiting for the connecting train's start times.

- Eg- tour arrives at node 2 at 1 pm, 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- Next train starts from 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 3.1 Decision variable ***t*** explained


node 2
reach node at 1 PM
spend pre-decided time of 3h at node
ideally ready to leave at 1 PM + 3h = 4 PM
1 PM
Buffer time = 3 hours
4 PM ideal exit time = t2
waiting period/gap = 2 hours
6 PM
Start = 6 pm, Cost = 10 hours
Link (train) starts at 6 pm
Flow of time, left to right

3.2 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 linear 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.

3.3 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 3.1 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	Objective function changed	Consistent results, rank of nodes not continuous, objective function requires refining
5	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 function to compare and find time constraints	Tests successful

3.4 Formulation

The objective function

$$\text{Minimise } \mathbf{t}[\text{destination}]$$

The consolidated list of constraints-

- Outdegree of each node = 1

  For all nodes i,

$$\Sigma\, x_{ijk} = 1 \text{ (summed over j, k)} \qquad (3.1)$$

- Indegree of each node = 1

  For all nodes j,

$$\Sigma\, x_{ijk} = 1 \text{ (summed over i, k)} \qquad (3.2)$$

- 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-

$$u_i + x_{ijk} <= u_j + (n-1)*(1-x_{ijk}) \qquad (3.3)$$

Modified for 3D networks-

$$u_i - u_j - (n-1) <= -n*\Sigma\, x_{ijk} \text{ (summed over k)} \qquad (3.4)$$

$$u_1=0 \qquad (3.5)$$

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

$$t_i <= Start_{ijk}*x_{ijk} + M*(1-x_{ijk}) \qquad (3.6)$$

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

$$t_i <= 0 + M = M \tag{3.7}$$

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

$$t_i <= Start_{ijk} + 0 \tag{3.8}$$

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

$$\max(t_i, Start_{ijk}) + Cost_{ijk}*x_{ijk} <= t_j + M*(1-x_{ijk}) \tag{3.9}$$

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

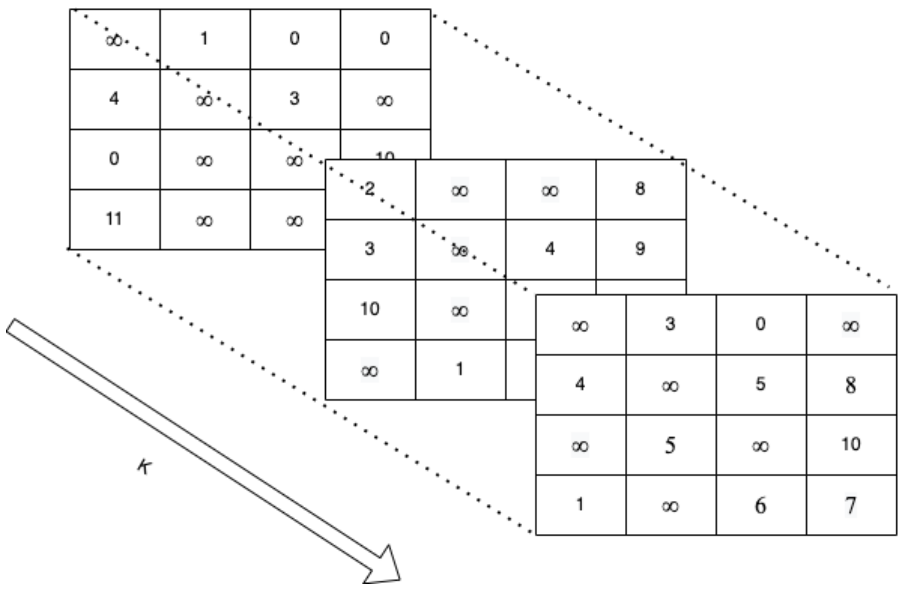
$$\max(t_i, Start_{ijk}) + 0 <= t_j + M \tag{3.10}$$

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

$$\max(t_i, Start_{ijk}) + Cost_{ijk} <= t_j + 0 \tag{3.11}$$

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 3.2 3-Dimension matrix visualised



3.5 Results and Conclusion

Example 1

n = 2, k = 2

Table 3.2   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 3.3 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 3.4 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 3.5 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

Conclusion

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 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 obstacle 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.

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