ESE 504/OIDD FALL 2019
INTRODUCTION TO OPTIMIZATION THEORY

PROJECT

PATH OPTIMIZATION FOR AUTONOMOUS MOBILE ROBOTIC SYSTEMS

Submitted by
GROUP 14_Tufted Titmouse

Degree: Master's in Electrical Engineering (EE)

Table of Contents

CHAPTER 1

INTRODUCTION

In autonomous mobile robotics (AMR) optimal path finding is a critical aspect of motion planning. It aims to minimize the cost in terms of time or energy (of the robot) to achieve the target. Various algorithms like Dijkstra's and Traveling Salesman Problem (TSP) will solve a straightforward shortest path problem, which gives an optimal path to reach a destination node from a source node. The TSP solves a typical n-cities tour optimization where all the n cities (nodes) must be visited only once. However, complex path planning in AMR do not require each node to be visited and may necessitate some nodes be visited while traveling from source node to destination node. Therefore, such intricate problems can be solved by integer programming. The choice of the optimization technique to be an integer programming model is so because the path traversed between any two nodes is a matter of decision (i.e. whether the robot takes the particular path between two nodes (True/1) or it does not (False/0)) , which cannot be fractional.

Path planning in multi-obstacle avoidance environment i.e. given a starting node 'A' and a target node: planning a trajectory to take the vehicle to the target under some specific conditions (constraints) while avoiding the obstacle, is named as Target-Pursuit and Obstacle-Avoidance (TPOA). Traditional methods such as probability road mapping and graphical solutions are not effective for such complex path planning problems. Many AI techniques like Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) are used to solve path planning problems but do not estimate the convergence time.

In this project we will identify major path planning problems and model them as an optimization problem involving integer programming. The models are built for various path planning scenarios in a 16-node floor map on which the robot traverses. This 16-node floor map is built for a Warehouse where each of these nodes can be potential source node, destination node or transitory node for the Autonomous Robot. The cost of travel is between any two given nodes is dependent on the layout of a particular warehouse. In this project, the costs of the edges are assigned arbitrarily. These costs can always be replaced with the actual costs of a warehouse layout. The cost of traversal is basically a representative of the energy expended by the robot and/or the time taken by the robot to traverse the node.

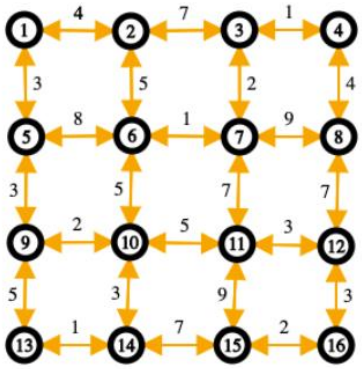
In path planning scenarios, the problem statements are usually like: Plan an optimal path for the robot to start at Order Counter (Node 1), pick up Item X at Counter 15 (Node 3), Pick up Item Y at Counter 22 (Node 8) and deliver it to Shipping Counter (Node 10), all while avoiding the path between Counter 20 (Node 6) and Counter 21 (Node 7) because there is a maintenance scheduled in those two counters. This project addresses Optimization techniques for solving such path planning problems.

Various integer programming models are built to implement path planning algorithms. These models are solved using LINDO. The LINDO files along with the corresponding outputs for each of these models are attached in the Appendix.

CHAPTER 2

PROBLEM FORMULATION

The figure below represents the floor map of the environment in which the robot is slated to traverse. The nodes represent the all possible origin and destination locations of the AMR. The edges joining these nodes represent all possible paths that the robot can take to traverse. Each edge in the map is associated with a cost of travel (alternatively, could be the time required to travel between the two nodes). All the models are built on the basis of this map/network where the costs associated with the edges are the coefficients of the decision variables in the objective function.



The decision variables in this scenario are defined as ***xi-j*** where ***i*** and ***j*** denote any two nodes. ***xi-j*** can take only two values: 0 or 1 based on whether the robot traverses the edge joining the nodes i and j. From the map, the costs associated with all the edges are ***ci_j***. The objective of any path planning problem is to reduce the overall cost of traversal across all the nodes. The objective function of the problem will give information about the total cost, in the form of time or energy, expended by the robot to complete its journey.

The constraints ensure that the robot travels from the source node to the required destination node. It is the responsibility of the constraints to clearly define a path optimization scenario which ensures the source node, the destination node, obstacle in the paths, compulsory nodes to be visited are all covered for the defined scenario. Three problems with increasing complexity in path planning are defined and the system is modelled for each of these and LINDO is used for solving these problem formulations.

The path planning problems to be addressed are:
1. Trivial shortest path problem
2. Obstacle avoiding shortest path problem
3. Constrained node shortest path problem

The techniques used in formulating the constraints for each of these types is discussed in detail and implemented using LINDO. A detailed analysis of the LINDO output will be done for each of these formulations.

CHAPTER 3

TRIVIAL SHORTEST PATH PROBLEM

Consider a path planning scenario in which a robot must start at node 1 and reach node 15 while minimizing the cost of traversal. For the given graph, *xi-j* denotes the edges/paths and *ci-j* denotes the cost associated with the traversal along the edge joining node *i* and *j*.

The objective function is thus given as:

$$\textbf{Minimize } z = \sum_{i,j}^{n} \textbf{Ci\_j} * \textbf{xi\_j}$$

The constraints are written for every node in the graph. The sum of all the xi-j terms for a given node must be 0 and only two constraint will have non-zero RHS values: RHS =1 corresponds to Source Node constraint and RHS = -1 which corresponds to Destination Node constraint.

The Trivial Shortest Path has been computed in LINDO. The objective function and path constraints for a trivial shortest path problem with source node 1 and destination node 15 is as follows:

Minimize z = 4x1_2 + 4x2_1 + 3x1_5 + 3x5_1 + 7x2_3 + 7x3_2 + 5x2_6 + 5x6_2 + 1x3_4 + 1x4_3 + 2x3_7 + 2x7_3 + 4x4_8 + 4x8_4 + 8x5_6 + 8x6_5 + 3x5_9 + + 1x9_5 + 1x6_7 + 1x7_6 + 5x6_10 + 5x10_6 + 9x7_8 + 9x8_7 + 7x7_11 + 7x11_7 + 7x8_12+ 7x12_8 + 2x9_10 + 2x10_9 + 5x9_13 + 5x13_9 + 5x10_11 + 5x11_10 + 3x10_14 + 3x14_10 + 3x11_12 + 3x12_11 + 9x11_15 + 9x15_11 + 3x12_16 + 3x16_12 + 1x13_14 + 1x14_13 + 7x14_15 + 7x15_14 + 2x15_16 + 2x16_15

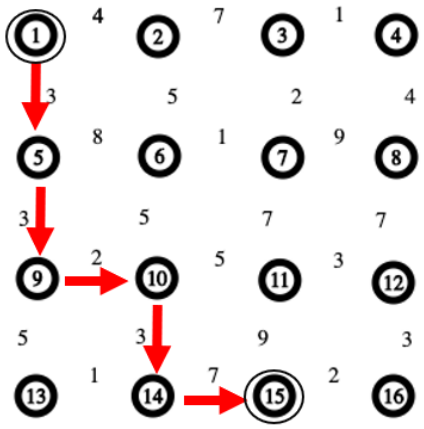
Subject to the following constraints:

x1_2 - x2_1 + x1_5 - x5_1 = 1 (for the source node)
-x1_2 + x2_1 + x2_3 - x3_2 + x2_6 - x6_2 = 0
-x2_3 + x3_2 + x3_4 - x4_3 + x3_7 - x7_3 = 0
-x3_4 + x4_3 + x4_8 - x8_4 = 0
-x1_5 + x5_1 + x5_6 - x6_5 + x5_9 - x9_5 = 0
-x2_6 + x6_2 - x5_6 + x6_5 + x6_7 - x7_6 + x6_10 - x10_6 = 0
-x3_7 + x7_3 - x6_7 + x7_6 + x7_8 - x8_7 + x7_11 - x11_7 = 0
-x4_8 + x8_4 - x7_8 + x8_7 + x8_12 - x12_8 = 0
-x5_9 + x9_5 + x9_10 - x10_9 + x9_13 - x13_9 = 0
-x6_10 + x10_6 - x9_10 + x10_9 + x10_11 - x11_10 + x10_14 - x14_10 = 0
-x7_11 + x11_7 - x10_11 + x11_10 + x11_12 - x12_11 + x11_15 - x15_11 = 0
-x8_12 + x8_12 - x11_12 + x12_11 + x12_16 - x16_12 = 0
-x9_13 + x13_9 + x13_14 - x14_13 = 0
-x10_14 + x14_10 - x13_14 + x14_13 + x14_15 - x15_14 = 0
-x11_15 + x15_11 - x14_15 + x15_14 + x15_16 - x16_15 = -1 (for the destination node)

-x12_16 + x16_12 - x15_16 + x16_15 = 0

∀ xi_j ≥ 0 (no paths are negative)

Results: On solving the above constraints using LINDO, we get the following path.
$x_{1\_5} = x_{5\_9} = x_{9\_10} = x_{10\_14} = x_{14\_15} = 1$
Thus, the path followed by the robot is: **1 → 5 → 9 → 10 → 14 → 15**
The minimum overall cost associated with this journey is 18.



CHAPTER 4

OBSTACLE AVOIDING- SHORTEST PATH PROBLEM

Consider a path planning scenario in which a robot must start at node 1 and reach node 10 while it avoids some paths which are not fit for the robot to traverse. This condition may arise in a real-world scenario when some parts of an environment are under construction or repair and hence the robot cannot travel there. In such cases, Dynamic programming techniques are used to tackle dynamic changes in the system and incorporate these conditions into the problem.

The objective function and constraints of the previous problem can still hold in this scenario, but in addition a few modifications must be made in them to ensure that the robot does not take the path which has obstacles.

There are two ways of tackling such Obstacle Avoiding Scenarios:

1. Modify the cost associated with the path in which an obstacle is present to a very large value which forces the optimization algorithm to not take that edge into the optimal solution. The very large cost could be as simple as the sum of all costs in the map.

2. Create additional binding constraints which ensure that those decision variables are zero. For example, if the path between 3 and 5 must be avoided, we must set $x_{3-5} = 0$ and $x_{5-3} = 0$.

Method 1- Cost Modification:

The following constraints have been solved using LINDO to avoid the path between nodes 9 and 10 and reach destination node 15 from source node 1.

Minimize z = 4x1_2 + 4x2_1 + 3x1_5 + 3x5_1 + 7x2_3 + 7x3_2 + 5x2_6 + 5x6_2 + 1x3_4 + 1x4_3 + 2x3_7 + 2x7_3 + 4x4_8 + 4x8_4 + 8x5_6 + 8x6_5 + 3x5_9 + 3x9_5 + 1x6_7 + 1x7_6 + 5x6_10 + 5x10_6 + 9x7_8 + 9x8_7 + 7x7_11 + 7x11_7 + 7x8_12+ 7x12_8 + 100x9_10 + 100x10_9 + 5x9_13 + 5x13_9 + 5x10_11 + 5x11_10 + 3x10_14 + 3x14_10 + 3x11_12 + 3x12_11 + 9x11_15 + 9x15_11 + 3x12_16 + 3x16_12 + 1x13_14 + 1x14_13 + 7x14_15 + 7x15_14 + 2x15_16 + 2x16_15

Subject to the following path constraints:

x1_2 - x2_1 + x1_5 - x5_1 = 1 (source node)
-x1_2 + x2_1 + x2_3 - x3_2 + x2_6 - x6_2 = 0
-x2_3 + x3_2 + x3_4 - x4_3 + x3_7 - x7_3 = 0
-x3_4 + x4_3 + x4_8 - x8_4 = 0
-x1_5 + x5_1 + x5_6 - x6_5 + x5_9 - x9_5 = 0

-x2_6 + x6_2 - x5_6 + x6_5 + x6_7 - x7_6 + x6_10 - x10_6 = 0
-x3_7 + x7_3 - x6_7 + x7_6 + x7_8 - x8_7 + x7_11 - x11_7 = 0
-x4_8 + x8_4 - x7_8 + x8_7 + x8_12 - x12_8 = 0
-x5_9 + x9_5 + x9_10 - x10_9 + x9_13 - x13_9 = 0
-x6_10 + x10_6 - x9_10 + x10_9 + x10_11 - x11_10 + x10_14 - x14_10 = 0
-x7_11 + x11_7 - x10_11 + x11_10 + x11_12 - x12_11 + x11_15 - x15_11 = 0
-x8_12 + x8_12 - x11_12 + x12_11 + x12_16 - x16_12 = 0
-x9_13 + x13_9 + x13_14 - x14_13 = 0
-x10_14 + x14_10 - x13_14 + x14_13 + x14_15 - x15_14 = 0
-x11_15 + x15_11 - x14_15 + x15_14 + x15_16 - x16_15 = -1 (destination node)
-x12_16 + x16_12 - x15_16 + x16_15 = 0

∀ xi_j ≥ 0 (no paths are negative)

Results: On solving the above problem, we get the following path.
$x_{1\_5} = x_{5\_9} = x_{9\_13} = x_{13\_14} = x_{14\_15} = 1$
Thus, the path followed by the robot is: **1 → 5 → 9 → 13 → 14 → 15**
The minimum overall cost associated with this journey is 19.

Method 2: Creating additional binding constraints

The following constraints have been solved using LINDO to avoid the path between nodes 9 and 10 and reach destination node 15 from source node 1.

Minimize z = 4x1_2 + 4x2_1 + 3x1_5 + 3x5_1 + 7x2_3 + 7x3_2 + 5x2_6 + 5x6_2 + 1x3_4 + 1x4_3 + 2x3_7 + 2x7_3 + 4x4_8 + 4x8_4 + 8x5_6 + 8x6_5 + 3x5_9 + 3x9_5 + 1x6_7 + 1x7_6 + 5x6_10 + 5x10_6 + 9x7_8 + 9x8_7 + 7x7_11 + 7x11_7 + 7x8_12+ 7x12_8 + 2x9_10 + 2x10_9 + 5x9_13 + 5x13_9 + 5x10_11 + 5x11_10 + 3x10_14 + 3x14_10 + 3x11_12 + 3x12_11 + 9x11_15 + 9x15_11 + 3x12_16 + 3x16_12 + 1x13_14 + 1x14_13 + 7x14_15 + 7x15_14 + 2x15_16 + 2x16_15

Subject to the following path constraints:

x1_2 - x2_1 + x1_5 - x5_1 = 1 (source node)
-x1_2 + x2_1 + x2_3 - x3_2 + x2_6 - x6_2 = 0
-x2_3 + x3_2 + x3_4 - x4_3 + x3_7 - x7_3 = 0
-x3_4 + x4_3 + x4_8 - x8_4 = 0
-x1_5 + x5_1 + x5_6 - x6_5 + x5_9 - x9_5 = 0
-x2_6 + x6_2 - x5_6 + x6_5 + x6_7 - x7_6 + x6_10 - x10_6 = 0
-x3_7 + x7_3 - x6_7 + x7_6 + x7_8 - x8_7 + x7_11 - x11_7 = 0
-x4_8 + x8_4 - x7_8 + x8_7 + x8_12 - x12_8 = 0
-x5_9 + x9_5 + x9_10 - x10_9 + x9_13 - x13_9 = 0
-x6_10 + x10_6 - x9_10 + x10_9 + x10_11 - x11_10 + x10_14 - x14_10 = 0
-x7_11 + x11_7 - x10_11 + x11_10 + x11_12 - x12_11 + x11_15 - x15_11 = 0
-x8_12 + x8_12 - x11_12 + x12_11 + x12_16 - x16_12 = 0
-x9_13 + x13_9 + x13_14 - x14_13 = 0

-x10_14 + x14_10 - x13_14 + x14_13 + x14_15 - x15_14 = 0
-x11_15 + x15_11 - x14_15 + x15_14 + x15_16 - x16_15 = -1 (destination node)
-x12_16 + x16_12 - x15_16 + x16_15 = 0

(forcing path between unwanted nodes to be zero)
x9_10 = 0
x10_9 = 0

∀ xi_j ≥ 0 (no paths are negative)

Results: On solving the above problem, we get the following path.
$x_{1\_5} = x_{5\_9} = x_{9\_13} = x_{13\_14} = x_{14\_15} = 1$
Thus, the path followed by the robot is: **1 → 5 → 9 → 13 → 14 → 15**
The minimum overall cost associated with this journey is 19.

CHAPTER 5

NODE CONSTRAINED - SHORTEST PATH PROBLEM

Consider a path planning scenario in which a robot must start at node 1 and reach node 15 but must visit nodes 3, 9 and 8. Such scenarios occur when security rounds are to be conducted using Autonomous Mobile Robot Systems and the robots must visit some nodes/locations and perform actions as described by a command. Therefore, the robot must visit the specified nodes at least once, in any order, before it reaches the target node. In such a situation, the problem becomes complex as the objective is to find an optimal path that reduces the overall cost of the robot reaching the target node by ensuring it passes through all the required intermediate nodes defined by the command.

A naïve method to solve this problem would be to add additional constraints to ensure that some nodes are visited, in the sense, we force those edges connecting these nodes to be '1'. For example, in the given problem, if the node 2 must compulsorily be traversed, the additional constraints would be as follows:
(The two constraints below are included in addition to the constraint set in chapter 3.)

x1_2 + x6_2 + x3_2 = 1
x2_1 + x2_6 + x2_3 = 1

Results: On running the corresponding program, LINDO gives the following path for the robot:
x1_2 = x2_6 = x6_10 = x10_14 = x14_15=1
Thus, the nodes traversed by the robot is: 1→ 2 → 6 → 10 → 14 → 15
The overall minimum cost associated with this journey is 24.

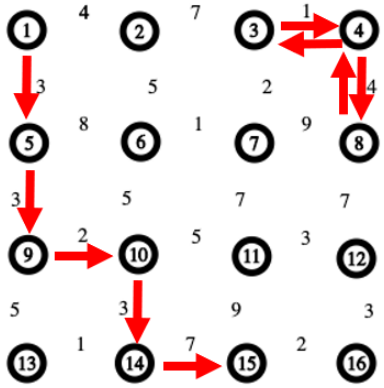
However, when we try to enforce multiple node constraints, this naïve method does not always ensure a fully connected optimal path. This can be seen when we try to force nodes 3, 8 and 9 before the robot reaches its destination node 15. Therefore, the additional constraints for forcing nodes 3, 8 and 9 are as follows. *(These constraints below are included in addition to the constraint set in chapter 3.)*

x2_3 + x7_3 + x4_3 = 1
x3_2 + x3_7 + x3_4 = 1

x4_8 + x7_8 + x12_8 = 1
x8_4 + x8_7 + x8_12 = 1

x5_9 + x10_9 + x13_9 = 1
x9_5 + x9_10 + x9_13 = 1

Results: On running the corresponding program, LINDO gives the following path for the robot:
x1_5 = x5_9 = x9_10 = x10_14 = x14_15 = x3_4 = x4_8 = x8_4 = x4_3 = 1
Thus, the nodes traversed by the robot is: **1→ 5 → 9 → 10 → 14 → 15 and 3 → 4 → 8 → 4 → 3**
The overall minimum cost associated with this journey is 19.



It is evident from the above graph that the robot has a subtour for traversing nodes 3 and 8. The two tours are not connected and thus this solution is not feasible in the real-world solution. Therefore, a better approach must be developed for problems involving multiple nodes to be enforced. Chapter 6 describes an algorithm that proposes a solution which combines traditional computational algorithms along with a trivial shortest path planning problem.

CHAPTER 6

HYBRID ALGORITHM FOR NODE CONSTRAINED PATH OPTIMIZATIONS

As discussed in the previous chapter, just adding additional constraints for forcing each node will not always a fully connected optimal path. Therefore, a new approach which combines the Traveling Salesman Problem (TSP) and the Trivial Shortest Path algorithm is proposed. The solution to this problem is in two phases.

Phase 1: Formulation of trivial shortest path for every pair of nodes which must be traversed by the robot.

The robot has to start at node 1, pass through the nodes 3, 8 and 9 and finally reach node 15. Therefore, the trivial shortest path algorithm is run for every pair of nodes: (1,3), (1,8), (1,9), (3,8), (8,9), (3,9), (3,15), (8,15), (9,15). The table below contains the optimal path for the pairs of nodes.

Table 1: Trivial Shortest Path between Nodes of the TSP Graph

Tableau	Node 3	Node 8	Node 9	Node 15
Node 1	X1-2 → X2-3	X1-2 → X2-3 → X3-4 → X4-8	X1-5 → X5-9	-
Node 3	-	X3-4 → X4-8	X3-7 → X7-6 → X6-10 → X10-9	X3-7 → X11-12 → X12-16 → X16-15
Node 8	X8-4 → X4-3	-	X8-4 → X4-3 → X3-7 → X7-6 → X6-10 → X10-9	X8-4 → X4-3 → X3-7 → X7-11 → X11-12 → X12-16 → X16-15
Node 9	X9-10 → X10-6 → X6-7 → X7-13	X9-10 → X10-6 → X6-7 → X7-3 → X3-4 → X4-8	-	X9-10 → X10-14 → X14-15

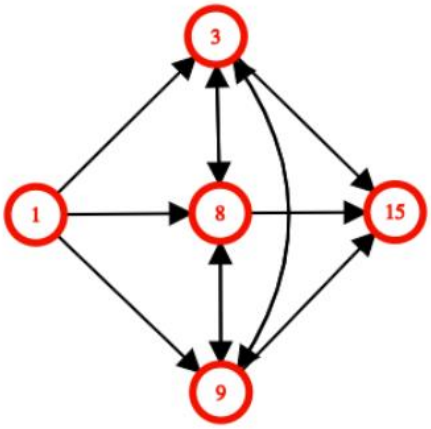
The table below summarizes the optimal costs of the above paths obtained from running the trivial shortest path algorithm.

Table 2: Cost Table

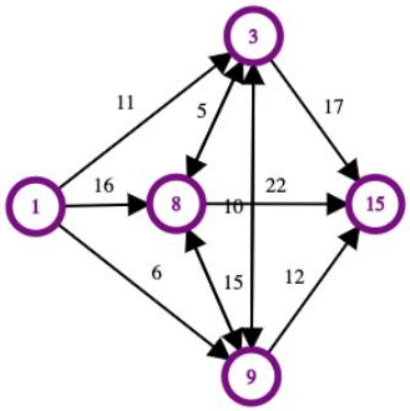
Tableau	Node 3	Node 8	Node 9	Node 15
Node 1	11	16	6	-
Node 3	-	5	10	17
Node 8	5	-	15	22
Node 9	10	15	-	12

Phase 2: In this phase, we run the TSP algorithm on nodes 1, 3, 8, 9, 15.
Source node: Node 1
Destination node: Node 15
Other nodes to be traversed: Nodes 3, 8, 9.

A graph containing only these nodes has to be solved using the TSP algorithm. In TSP, each node must be visited exactly once, starting from the source node and ending at the destination node. The graph for the TSP is as shown below.



From Table 2, the costs for the edges of the above graph are filled to get the graph to be solved using TSP.



The objective function and constraints to solve a TSP for the above graph are formulated as below.

Minimize z = 11x1_3 + 16x1_8 + 6x1_9 + 10x3_9 + 5x3_8 + 17x3_15 + 5x8_3 + 22x8_15 + 15x8_9 + 15x9_8 + 10x9_3 + 12x9_15

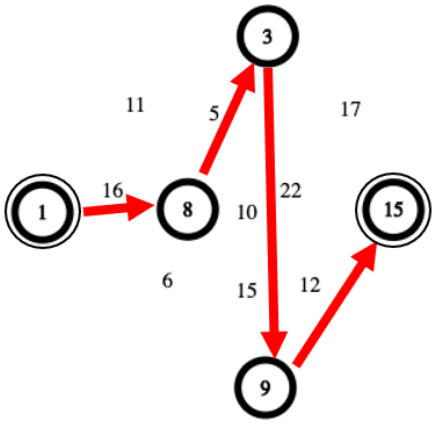
Subject to the following constraints:

x1_3 + x1_8 + x1_9 = 1 (start at node 1)

$x8\_3 - x3\_8 + x8\_9 - x9\_8 + x8\_15 - x1\_8 = 0$
$x3\_8 - x8\_3 + x3\_15 - x1\_3 + x3\_9 - x9\_3 = 0$
$x9\_8 - x8\_9 + x9\_15 - x1\_9 + x9\_3 - x3\_9 = 0$
$x9\_15 + x8\_15 + x3\_15 = 1$
$x8\_3 + x8\_9 + x8\_15 = 1$
$x1\_8 + x3\_8 + x9\_8 = 1$
$x3\_8 + x3\_15 + x3\_9 = 1$
$x1\_3 + x8\_3 + x9\_3 = 1$
$x9\_8 + x9\_15 + x9\_3 = 1$
$x1\_9 + x8\_9 + x3\_9 = 1$

$x_{1\_3} + x1\_9 + x1\_8 + x3\_8 + x3\_9 + x3\_15 + x8\_3 + x8\_9 + x8\_15 + x9\_3 + x9\_8 + x9\_15 = 4$

By implementing the above constraints and the objective function in LINDO, we get the solution for the TSP, which ensures that the robot begins at node 1 and terminates its journey at node 15 while passing through the nodes 3, 8 and 9 exactly once. The graph below shows the optimal path returned by the TSP algorithm.

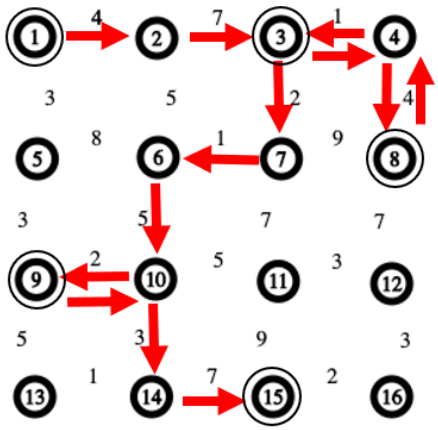


Now, the data stored in phase 1 tableau is used to obtain the final path of the robot. Table below summarizes the final solution to the node constrained problem.

Table 3: Trivial Shortest Path between Nodes of the TSP Graph

Edges of TSP Solution	Optimal Path from Original Graph
X1-8	X1-2 → X2-3 → X3-4 → X4-8
X8-3	X8-4 → X4-3
X3-9	X3-7 → X7-6 → X6-10 → X10-9
X9-15	X9-10 → X10-14 → X14-15

The final optimal path which ensures a fully connected graph for a node constrained path optimization problem is depicted in the figure below.



The optimal cost of traversal for the above solution as computed by LINDO is 43.

This solution ensures that there are no subtours and that all the nodes, which are forced, are fully connected. However, the path is such that it revisits some other nodes, all because the algorithm tries to keep the cost minimal by avoiding other costlier paths in the map.

Therefore, the hybrid algorithm combines both- the TSP and the Trivial Shortest Path finding to ensure that a fair and practically optimal solution is obtained.

REFERENCES

[1] L.Yang, J. Han, C. Wu, and Y. Nie. “A Solution of Mixed Integer Linear Programming for Obstacle-Avoided Pursuit Problem”, 2008 WCCI, Hongkong, pp.110-115, 2008.

[2] L. Yang, C. Wu, J. Han, X. Zhao and Y. Nie. “A New LP-based Obstacle-avoided Model in Path Planning Problem”, 2009 CCDC, Guilin, pp.110-115.

[3] Schouwenaars, T., De Moor, B., Feron, E., & How, J. (2001, September). Mixed integer programming for multi-vehicle path planning. In 2001 European control conference (ECC) (pp. 2603-2608). IEEE.

[4] Ding, H., Zhou, M., & Stursberg, O. (2009, October). Optimal path planning in the workspace for articulated robots using mixed integer programming. In 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 5770-5775). IEEE.