IE400 - Project Report

Sait Aktürk 21501734 | Kerem Ayöz 21501569 | Abdullah Talayhan 21401137

31 December 2019

1 Introduction

Our aim is to find the shortest path that traverses all the holes in order to drill them in an efficient manner. A drilling head should traverse all of the holes and return to the starting point. The total distance that the head has to be moved should be minimized in a given Cartesian coordinates of 50 points that must be drilled and 20 rectangular blocks in which drilling machine is not allowed to pass through them. We developed a mathematical model similar to Travelling Salesman Problem. Our mathematical model is as follows.

2 Mathematical Model

For learning purposes, we didn't want to use the already implemented model given in the CPLEX software. Hence, we have used another formulation for Traveling Salesman Problem called MTZ [1] and implemented it ourselves in CPLEX.

Decision Variables
x_{ij} : There exists a path between node i and j in the optimal solution
u_i : Dummy decision variable for eliminating subtours using the MTZ formulation.

Parameters
c_{ij} : Cost of traveling from node i to node j
B : Set of (i, j) pairs where there exists a block between node i and node j
n : Number of nodes

$$\text{minimize} \quad \sum_{i,j \in V, i \neq j} c_{ij} x_{ij}$$

$$\text{subject to} \quad \sum_{i=1, i \neq j} x_{ij} = 1 \quad j = 1, ..., n$$

$$\sum_{j=1, i \neq j} x_{ij} = 1 \quad i = 1, ..., n$$

$$x_{ij} = 0 \quad \forall (i, j) \in B$$

$$x_{ij} \in \{0, 1\} \quad 1 \leq i, j \leq n, i \neq j$$

$$u_1 = 1, \quad 2 \leq u_i \leq n \quad \forall i = 2, ..., n$$

$$u_j - u_i + 1 \leq (n-1) \cdot (1 - x_{ij}) \quad 2 \leq i, j \leq n, \quad i \neq j$$

MTZ formulation mainly differs from the original by its subtour elimination constraint. It uses dummy decision variables that eliminates subtours instead of generating the powerset of the given nodes. Hence, resulting in a more concise formulation.

Block Detection

During preprocessing (**file:** preproc.py), we take the given input of nodes and obstacles, then we run Breadth First Search Algorithm between pairwise nodes for finding the shortest path between them. If the shortest path is longer than the minimum manhattan distance between the to nodes, then we can say that there exists an obstacle between them.

CPLEX Implementation

See Appendix for the implementation source code.

3 Results & Simulation

Figure 1: Optimal drill movement.

- We have simulated the path using python during postprocessing (**file:** postproc.py) and created an animation for the optimal path (Figure 1). The gray blocks represent obstacles, yellow blocks represents nodes to be drilled. The animation can be watched via the following link:
- https://www.youtube.com/watch?v=5C31YzJpA-4
- We have obtained an optimal path with size 382 (Figure 2).

Statistic	Value
▼Cplex	solution (optimal) with objective 382
Constraints	2744
▼Variables	2500
Binary	2450
Other	50
Non-zero coefficients	12248
▼MIP	
Objective	382
Incumbent	382
Nodes	15555
Remaining nodes	0
Iterations	299999
▼Solution pool	
Count	16
Mean objective	527.25

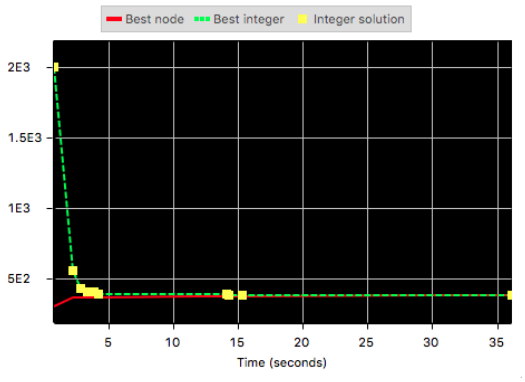

Figure 2: Optimal Solution Statistics.

4 Appendix - CPLEX Implementation

```
// problem size
int n = ...;
range nodes = 1..n;

// edge tuple
tuple edge {
  int i;
  int j;
}

setof(edge) edges = {<i,j> | i,j in nodes : i!= j};
float dist[edges] = ...;
setof(edge) blocked = ...;

// decision variable
dvar boolean x[edges];
dvar float+ u[1..n];

// expressions
dexpr float TotalDistance = sum(e in edges) dist[e]*x[e];

// objective
minimize TotalDistance;

// constraints
subject to {

  forall (j in nodes)
    flow_in:
    sum( i in nodes: i != j) x[<i,j>] == 1;

  forall (i in nodes)
    flow_out:
    sum( j in nodes: j != i) x[<i,j>] == 1;

  forall (e in blocked)
      no_path:
      x[e] == 0;

   u[1] == 1;
   forall(i in 2..n) 2<=u[i]<=n;
   forall(e in edges: e.i!=1 && e.j!=1) (u[e.j]-u[e.i])+1<=(n-1)*(1-x[e]);
}
```

5 References

[1] C. Miller, A. Tucker and R. Zemlin, "Integer Programming Formulation of Traveling Salesman Problems", Journal of the ACM, vol. 7, no. 4, pp. 326-329, 1960. Available: 10.1145/321043.321046.