Minimizing the Distance Travelled by a Drill Head on a Rectangular Grid with Forbidden Regions

Ünsal Öztürk[1], Faruk Oruç[2], and Doğa Oruç[3]

[1]Department of Computer Science, Bilkent University, ID: 21601565
[2]Department of Computer Science, Bilkent University, ID: 21601844
[3]Department of Computer Science, Bilkent University, ID: 21602022

October 24, 2020

1 Problem Definition

We are given a rectangular grid representing a circuit board. Each vertex of this rectangular grid is labelled with a 2D Cartesian coordinate. We are also given a set of vertices, each denoted by a two dimensional coordinate, to drill a hole through. The drill moves according to the shortest Manhattan distance between these vertices. There are some predefined forbidden regions in the form of subgrids on the rectangular grid, through which the drill head can not move through. We are required to minimize the distance travelled by the drill head given that the drill head must return to the first vertex in the set of vertices through which the holes are drilled.

1.1 Terminology

The following terminology is to be used throughout this report:

Hole A point on the grid through which a hole is to be drilled. Given as part of the problem definition. A hole is uniquely represented with a 2D Cartesian coordinate. The drill head travels between the holes and can only drill a hole at the vertex at which it is located.

Forbidden Region A rectangular subset of the rectangular grid, through the vertices of which the drill head can not move. A forbidden region is uniquely represented with a 2D Cartesian coordinate, its width and its height, all integral values.

Drill Head The instrument that drills holes on the rectangular grid. The starting position of the drill head is assumed to be the first point in the set of holes. The drill head is able to move from one vertex to another in the rectangular grid. Whenever the drill head travels from a vertex to another, we assume that it has travelled 1 unit of distance. The drill head must visit all the holes by moving from one vertex to another, and then it must return back to the location of the original hole. The distance travelled by the drill head is to be minimized.

2 Approach

To minimize the distance travelled by the drill head, we assume that the during the motion of the drill head, the drill head visits the holes only once. By introducing this extra assumption, considering the other assumptions, definitions and our objective, the problem reduces to the symmetric travelling salesman problem.

2.1 Mathematical Model

This subsection describes the precise mathematical model to minimize the total distance travelled by the drill head.

2.1.1 Graph Model

We first abstract the geometry and the quantization due to discrete nature of the grid and the L_1 norm and forbidden regions by converting the problem into a graph problem. We construct a graph $G = (V, E)$ where V is the set of holes, the elements of which are indexed by the order of appearance in the provided set of holes, i.e. $V = \{v_1, v_2, ..., v_N\}$ where N is the number of holes to be drilled. We assume that the graph is completely connected and undirected. Accordingly, we choose

$$E = \{(v_i, v_j) \mid i, j \in [1, N] \ \wedge \ i < j\}$$

The assignment of weights to edges is the tricky part of the graph formulation of the problem. We are looking for a weight mapping W of the form

$$W : E \to \mathbb{Z}^+$$

so that moving from some v_i to v_j in the graph accurately simulates the movement of the drill head on the rectangular grid. For this purpose, we propose two types of edge-weight assignments based on the allowed movements of the drill head. Both modes of movements rely on the following lemma.

Lemma 1. *Given a Hamiltonian cycle spanning over all the vertices of G, the sum of the weights of the edges connecting the vertices in this Hamiltonian cycle is minimized when W maps the smallest possible value to the corresponding edge.*

Proof. We prove this lemma by contradiction using a cut and paste argument. Let $\mathbb{H}$ be a minimum edge weighted Hamiltonian cycle with vertices $v_{a_1}, v_{a_2}, ..., v_{a_n}, v_{a_1}$ such that for all edges (i, j), $W(i, j) = W_{ij}$. Assume that for some (a, b) there exists a more optimal edge assignment W'_{ab} such that $W_{ab} > W'_{ab}$ in W. Note that

$$\sum_{(i,j)\in E(\mathbb{H})} W_{ij} > \sum_{(i,j)\in E(\mathbb{H})-(a,b)} W_{ij} + W'_{ab}$$

since $W_{ab} > W'_{ab}$. This is a contradiction, since there exists an edge assignment W' with $W'((a, b)) = W'_{ab} < W_{ab}$ that has a sum of edge weights less than the Hamiltonian cycle, which means such a cycle can not be a minimum Hamiltonian cycle. ■

Corollary 1. *The optimal drill head movement (i.e. the minimum weighted Hamiltonian cycle) has a mapping W such that W assigns weights to edges according to the length of the shortest allowed paths in the rectangular grid.*

The two models for edge weight assignments we propose enforce constraints on the connectedness of any two nodes rather than the length of the shortest paths between the holes in the rectangular grid.

2.1.2 Weight Assignment Model Based on Shortest Manhattan Distance

This model of weight assignment assumes that the drill head can only move through one of the paths that attains shortest Manhattan distance between holes. That is to say, a hole with coordinates (x_i, y_i) is reachable from another hole (x_j, y_j) if and only if there exists a path through which the drill head can reach hole j from hole i in $|(x_i - x_j)| + |(y_i - y_j)|$ steps. More precisely, we let

$$W((i, j)) = \begin{cases} |(x_i - x_j)| + |(y_i - y_j)| & \text{if there is a shortest Manhattan distance path from hole i to j} \\ \infty & \text{else} \end{cases}$$

That is to say that if a forbidden region completely covers a layer of the rectangular region whose corners are marked with two holes, the holes are regarded to be disconnected, and the drill hole is not allowed to move from the first hole to the other, even if there is a way around the forbidden region. To check whether there exists a path from one hole to another, the following procedure, DFS-TC is employed.

```
1: procedure DFS-TC(M, N, v_i, v_j)
2:     visited ← ASSOCIATIVE LIST OF SIZE MN
3:     s ← STACK WITH ZERO ENTRIES
4:     PUSH(s, v_i)
5:     v_p ← NULL
6:     d ← ASSOCIATIVE LIST OF SIZE MN
7:     while SIZE(s) > 0 do
8:         v ← POP(s)
9:         if M(v_x, v_y) is blocked then
10:            continue
11:        else if v = v_j then
12:            d[v] ← v_p
13:            return d
14:        else if visited[v] = FALSE then
15:            visited[v] ← TRUE
16:            d[v] ← v_p
17:            v_p ← v
18:            for all neighbors v_n of v towards v_j do
19:                PUSH(s, v_n)
20:            end for
21:        end if
22:    end while
23:    return NULL
24: end procedure
```

Note that if DFS-TC finds a path from v_i to v_j, i.e. returns the parent pointer list, the shortest Manhattan distance between the two points can be computed simply by computing $|(x_i - x_j)| + |(y_i - y_j)|$, the value of which is known beforehand. The path from v_i to v_j can be easily reconstructed using d.

2.1.3 Weight Assignment Model Based on the Floyd-Warshall Algorithm

This weight assignment assumes that the drill head can move to any hole from a given hole, if the two holes are connected, regardless of whether this connection is a path with the shortest Manhattan distance or not. I.e., even if there is a forbidden region spanning over a layer of the rectangular subgrid formed by two holes, the drill head is allowed to go around this region. Therefore, we create the mapping W as

$$W((i,j)) = \begin{cases} \text{shortest distance between i and j} & \text{if hole i is reachable from hole j} \\ \infty & \text{else} \end{cases}$$

The previous procedure does not work here, as it assumes that the drill head can not go away from the source node. Therefore, we employ the 4-neighborhood Floyd-Warshall algorithm and find all-pairs shortest paths. The procedure below computes all-pairs shortest paths for the graph, given the rectangular grid edge configuration X, where X assumes 4-neighborhood connectivity over the grid with edge weights 1, or if a vertex on the rectangular grid borders a forbidden region, then the edge between the members of the region and other regions are assumed to be of weight ∞.

```
1: procedure FLOYD-WARSHALL(N, X)
2:     dist ← MATRIX OF SIZE N^2
3:     next ← MATRIX OF SIZE N^2
4:     for i ← 1, N do
5:         dist[i, i] ← 0
6:         next[i, i] ← i
7:         for j ← 1, N do
8:             dist[i, j] ← ∞
9:             next[i, j] ← j
10:        end for
11:    end for
12:    for k ← 1, N do
```

```
        for i ← 1, N do
            for j ← 1, N do
                if dist[i, j] > dist[i, k] + dist[k, j] then
                    dist[i, j] = dist[i, k] + dist[k, j]
                    next[i, j] = next[i, k]
                end if
            end for
        end for
    end for
    return DIST, NEXT
end procedure
```

Paths can be easily constructed from $next$, and $W((i,j))$ can be simply assigned as $dist[i,j]$, i.e. the 4-neighbourhood shortest path from hole i to j. This way, the drill can move around the forbidden regions, and this formulation hence offers a more efficient solution.

2.1.4 Symmetric Travelling Salesman Problem Integer Programming Formulation

The integer programming formulation of the symmetric travelling salesman can be used after all parameters, i.e. the edge weights W_{ij}, are set properly using one of the methods described in the previous sections. For the graph formulation, we have $C(N,2)$ binary decision variables, each of them labelled X_{ij}, denoting whether an edge is in the resulting minimum Hamiltonian cycle. X_{ij} is assigned 1 if it is the part of the min Hamiltonian cycle, and 0 otherwise. We have $C(N,2)$ parameters, namely W_{ij}, which denotes the edge weight between hole i and j. We may now express the problem as an linear integer program, as

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}X_{ij} \\
\text{subject to} \quad & \sum_{j=1}^{n} X_{ij} = 1, \quad i = 1,\ldots,n, \\
& \sum_{i=1}^{n} X_{ij} = 1, \quad j = 1,\ldots,n, \\
& X_{ii} = 0, \quad i = 1,\ldots,n, \\
& \sum_{i\in S}\sum_{j\notin S} X_{ij} \geq 1, \quad \forall S \subseteq N, S \neq \emptyset, \\
& X_{ij} \in \{0,1\}, \quad i,j = 1,\ldots,n
\end{aligned} \tag{1}$$

which is the classic TSP formulation.

3 Implementation Details, Results, and Simulation

We implemented the graph processing stage (check sections 2.1.1, 2.1.2, 2.1.3) in MATLAB. The relevant scripts to preprocess the input and convert the input into a graph is located in the tsp_preprocess.m file. Running the script for computing a weight mapping as described in 2.1.2 takes around 0.25 seconds, whereas computing all-pairs shortest paths using Floyd-Warshall takes around 120 seconds, due to the poor asymptotic complexity of the algorithm and complexity of the problem (shortest-path pairs for 2500 nodes results in $C(2500,2) = 3123750$ shortest path calculations, which is rather expensive to calculate). To obtain the preprocessed results, simply put the data.xlsx file and the MATLAB script in the same directory, and run the script.

The implementation of the mathematical model was done in CPLEX. We used the example Symmetric Travelling Salesman model as distributed by IBM, along with CPLEX. The model is located in two files, I400F19PROJ_fw.mod and I400F19PROJ_tc.mod. The former file reads in shortest paths in Floyd-Warshall format, and the latter reads in the shortest Manhattan distance only format. These are

organized into two different run configurations, called Floyd-Warshall and Minimum_Manhattan. Please note that the TSP model in IBM's version of TSP is slightly different than the formulation given in the previous section due to the way the data is stored. The distances are stored as an upper triangular matrix, due to the symmetricity of the problem, hence rendering the formulation slightly different. The formulation in the implementation is given as:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}X_{ij} \\ \text{subject to} \quad & \sum_{(i,j)\in E_j} X_{ij} + \sum_{(j,k)\in E_j} X_{jk} = 2, \quad j = 1,\ldots,n, \\ & \sum_{i=1}^{n} X_{\min(i,S_i),\max(i,S_i)} \leq |S| - 1, \quad S_i \in S \subseteq N, \\ & X_{ij} \in \{0,1\}, \quad i,j = 1,\ldots,n \end{aligned} \tag{2}$$

The implementation is a relaxation of the original problem in the sense that subtours are enumerated in a post-process manner, and the subtours are removed after the solution to the 'relaxed' TSP. This allows computing solutions faster, in comparison to the model given in previous sections. Note that the model assumes $W_{ij} = W_{ji}$ and $W_{ii} = 0$ a priori. The optimal objective function values for each of these run configurations are as follows:

Floyd-Warshall:
Iteration 9 with 8 subtours.
Found subtour of size : 50
Current solution : 368
Writing solution to tsp_fw_soln.csv.

Minimum_Manhattan
Iteration 35 with 34 subtours.
Found subtour of size : 50
Current solution : 382
Writing solution to tsp_tc_soln.csv.

Decision variable values for the optimal solution are written to tsp_fw_soln.csv and tsp_tc_soln.csv. The visualization and simulation tool finds a Hamiltonian cycle using the values of the decision variables, and displays/prints the indices of the holes that the drill head should visit, in order.

Order of holes for Floyd-Warshall edge weight assignment that results in a minimum Hamiltonian cycle:
1 - 48 - 28 - 5 - 47 - 20 - 50 - 45 - 46 - 32 - 34 - 39 - 33 - 3 - 10 - 44 - 49 - 27 - 16 - 7 - 41 - 37 - 6 - 21 - 31 - 15 - 29 - 40 - 13 - 24 - 36 - 25 - 17 - 2 - 23 - 9 - 8 - 38 - 19 - 4 - 43 - 30 - 42 - 12 - 11 - 22 - 18 - 26 - 35 - 14 - 1

Order of holes for Manhattan edge weight assignment that results in a minimum Hamiltonian cycle:
1 - 18 - 26 - 50 - 45 - 20 - 47 - 5 - 28 - 48 - 42 - 30 - 43 - 7 - 41 - 16 - 37 - 49 - 27 - 46 - 32 - 34 - 39 - 33 - 3 - 44 - 10 - 6 - 29 - 15 - 21 - 31 - 40 - 13 - 24 - 36 - 25 - 17 - 23 - 2 - 9 - 8 - 38 - 4 - 19 - 12 - 11 - 22 - 35 - 14 - 1

We implemented the simulation and visualization in Processing, an open-source visualization tool based on Java. To run the visualization tool, one can simply install Processing, and run the sketch named TSPVisualize. To run properly, all output files produced by the CPLEX model and the MATLAB script must be present in the directory of TSPVisualize. The tool is interactive, and simulates the movement of the drill head. The following figures are the results of the simulation with both edge weight mappings.

Current Mode: Paths in between holes are allowed only through the shortest Manhattan Distance

Current Path Length: 382

Current Number of Edges: 50

Edges: 1 - 18 - 26 - 50 - 45 - 20 - 47 - 5 - 28 - 48 - 42 - 30 - 43 - 7 - 41 - 16 - 37 - 49 - 27 - 46 - 32 - 34 - 39 - 33 - 3 - 44 - 10 - 6 - 29 - 15 - 21 - 31 - 40 - 13 - 24 - 36 - 25 - 17 - 23 - 2 - 9 - 8 - 38 - 4 - 19 - 12 - 11 - 22 - 35 - 14 - 1

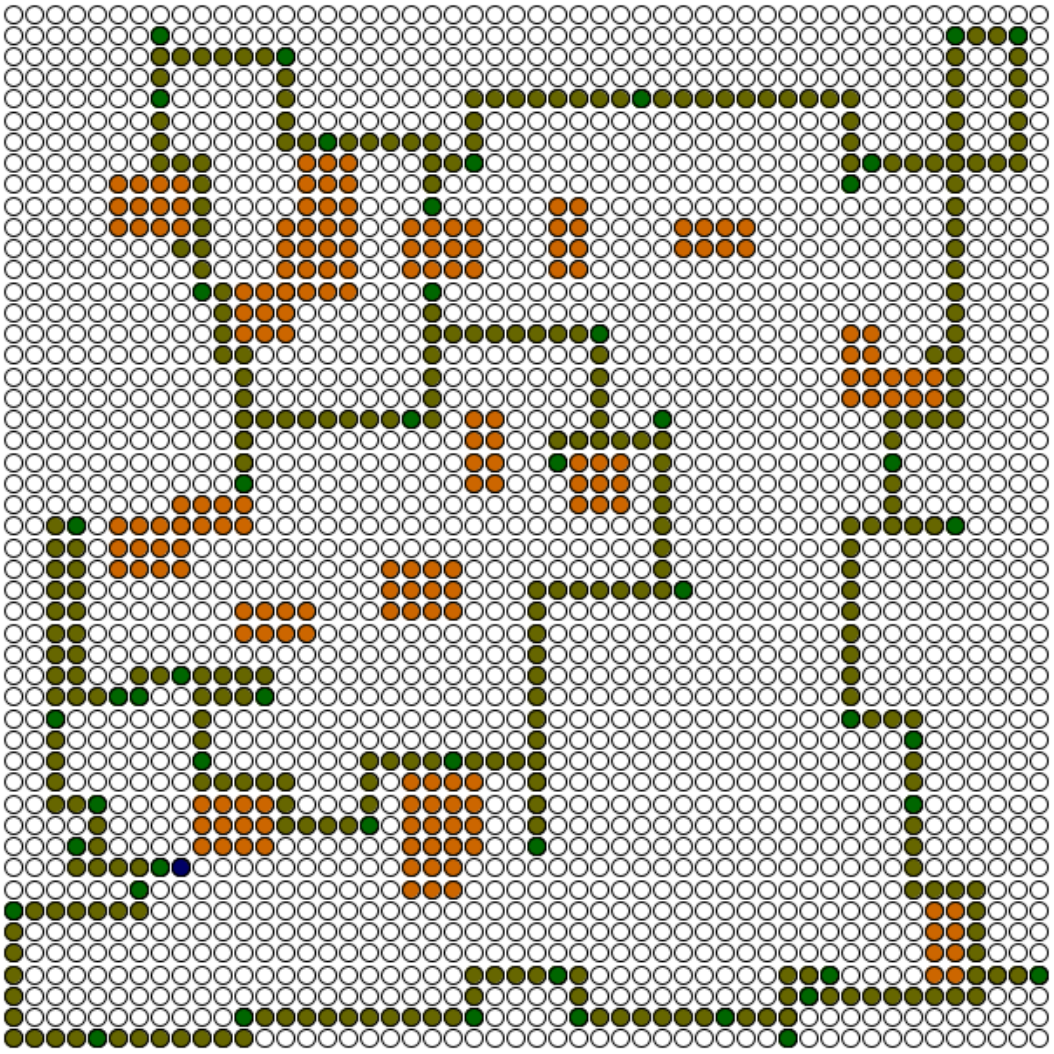

Figure 1: Simulation results for drill head movement for the Minimum Manhattan setting.

Current Mode: Paths in between holes are always allowed, regardless of the shortest Manhattan Distance

Current Path Length: 368

Current Number of Edges: 50

Edges: 1 - 48 - 28 - 5 - 47 - 20 - 50 - 45 - 46 - 32 - 34 - 39 - 33 - 3 - 10 - 44 - 49 - 27 - 16 - 7 - 41 - 37 - 6 - 21 - 31 - 15 - 29 - 40 - 13 - 24 - 36 - 25 - 17 - 2 - 23 - 9 - 8 - 38 - 19 - 4 - 43 - 30 - 42 - 12 - 11 - 22 - 18 - 26 - 35 - 14 - 1

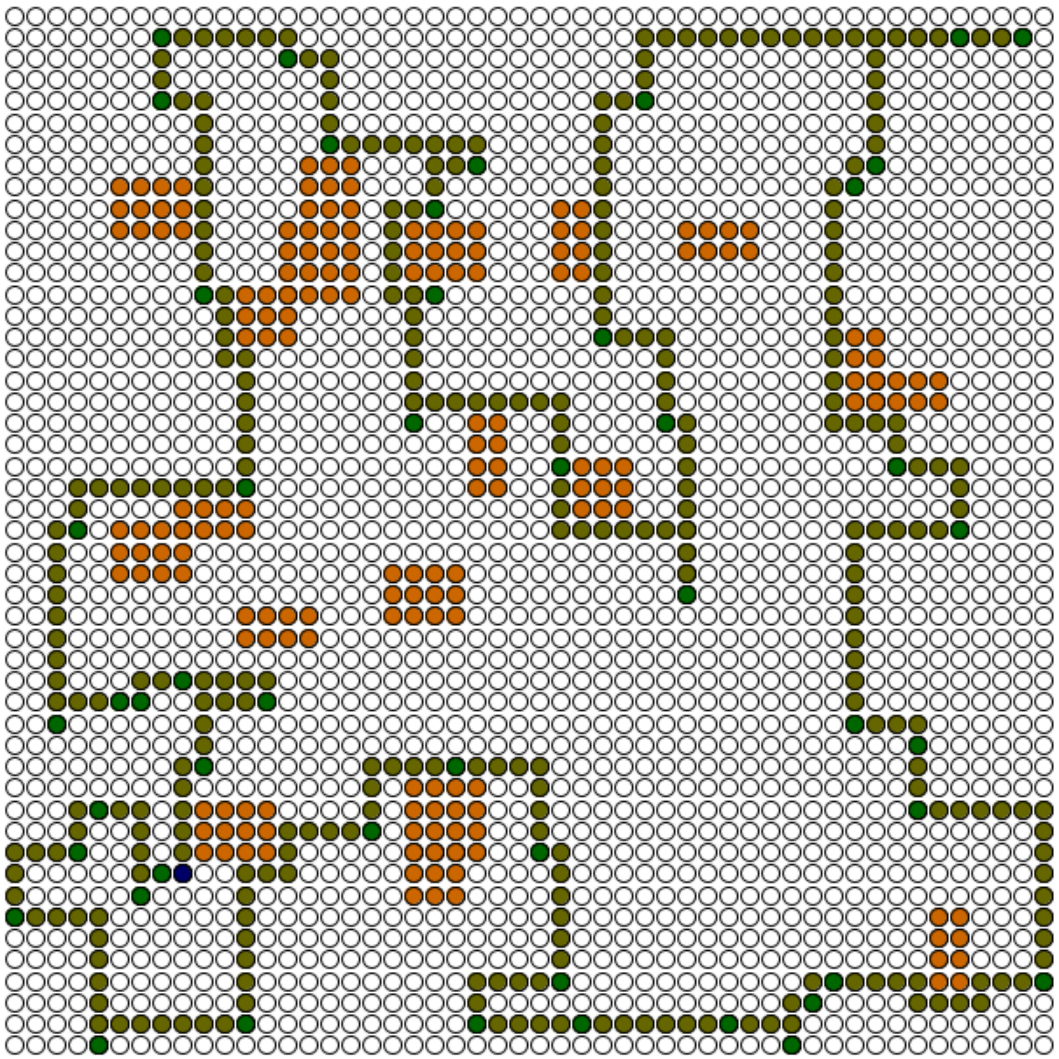

Figure 2: Simulation results for drill head movement for the Floyd-Warshall setting.