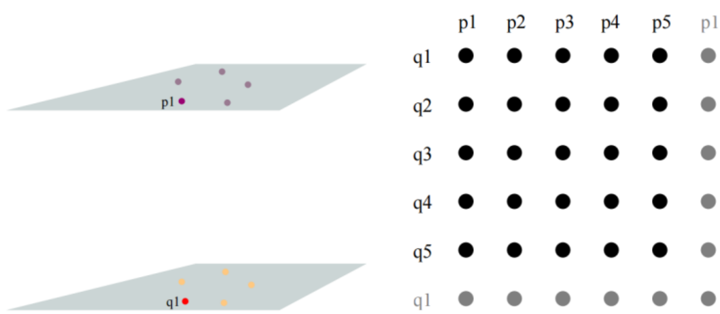
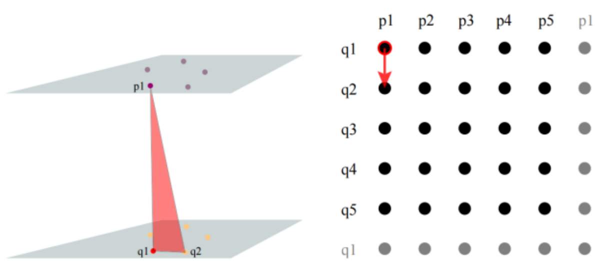
1. To solve the problem, you can create a toroidal graph. Each point represents a line segment. The columns represent the points of the upper contour and the rows represent the points of the

Toroidal Graph Method

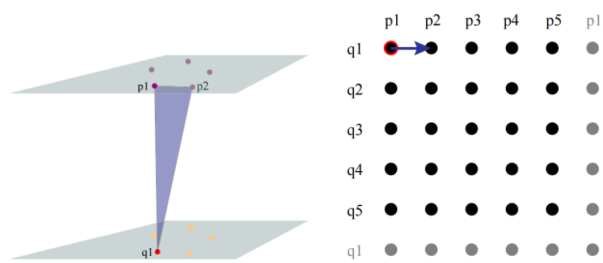



lower contour. Drawing an arrow on the graph from one point to another represents a triangle. The two graph points of the arrow represent two sides and the arrow represents the third side . A rightwards arrow has two points on the upper contour and a downwards arrow has two points on the lower contour.

Right or Down

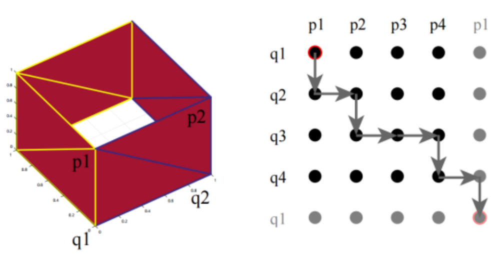



Directed Edges on the Graph

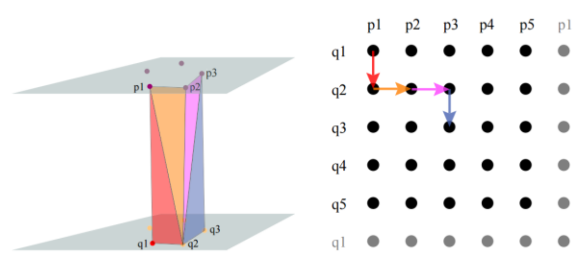



To create a complete triangulation, you need a full path down the graph back to the point you started with.

A Full Path Represents a Solution




Find the Path With Least Cost




To find the optimal path, you can construct a cost matrix.

There is only one way to optimally travel across the top row and that is to go right till you reach the n+1th point where n is the number of points on the upper contour. To go down the 1st column, the optimal solution is to go downwards until you reach the m+1th row where m is the number of points on the lower contour. Then you can fill out the rest of the cost matrix row by row. Each value of the cost matrix is the optimal sum of all triangles leading to the point. Each cost matrix index is equivalent to a point on the toroidal graph.

The principal of optimality comes into play in the fact that when constructing the cost matrix, each matrix entry represents the total cost of the optimal path up to that point on the toroidal graph. Every cost matrix entry uses prior entries to decide the optimal decision. When you have reached the lower right corner of the cost matrix of size m+1 x n+1, you will have the total cost of the optimal path of the toroidal graph. This represents the total area of the optimal triangulation.

From this cost matrix, you can then use a greedy algorith to construct the optimal path and output that path.

| | | |
|---|---|---|
| 1 | 4 | 10 |
| 3 | 5 | 8 |
| 4 | 13 | 12 |

2. The recurrence relation used when creating the cost matrix, which solves the problem, is:

$$M(x,y) = min((M(x-1,y)+area(P(x-1),P(x),Q(y))),(M(x,y-1)+area(P(x),Q(y-1),Q(y)))$$

   Where M(x,y) is a given matrix entry at position x,y and area() calculates the area of the triangle formed by 3 points. P(i) gives the ith point of the upper contour and Q(j) gives the jth point of the lower contour.

3. A table of partial solutions is built in the manner described in 1. You start by calculating the upper row of the cost matrix, which has only one way of being constructed, continuously moving right along the toroidal graph. Then you solve for the first column which is done by continuously moving down the toroidal graph. Then you fill in the cost matrix row by row using the recurrence relation above. Here is a partial cost matrix for 001_boxPoints.txt:

| | | | | |
|---|---|---|---|---|
| 0 | .5 | 1.20711 | 1.91421 | 2.41421 |
| .5 | 1 | 1.5 | 2.20711 | 2.91421 |
| 1.20711 | | | | |
| 1.91421 | | | | |
| 2.41421 | | | | |

   To fill out the next entry ( in red ) you look to the right and add area() from the right to that point which is .5, resulting in 1.70711. You then look above and add area() from above to that point, which is .5, resulting in 1.5. Then you take min(1.70711, 1.5) = 1.5. So 1.5 is the next entry. You continue filling out the table until you arrive at the lower right hand corner which represents the optimal cost of the minimum path of the toroidal graph, which in this case is 4.