Objective:

To minimize the total Manhattan distance between matched pairs of points. The objective function is expressed as:

$$\text{Minimize} \quad Z = \sum_{i<j} d_{ij} \times x_{ij}$$

Where d_{ij} is the Manhattan distance between points i and j, and x_{ij} is a binary decision variable that is 1 if points i and j are matched, and 0 otherwise.

Decision Variables:

$x_{ij} \in \{0, 1\}$

For each pair of points (i, j) where $i < j$, indicating whether the points are paired (1) or not (0).

Constraints:

Each point must be matched exactly once, resulting in constraints for each point:

$$\sum_{\substack{j \neq i \\ j > i}} x_{ij} + \sum_{\substack{j \neq i \\ j < i}} x_{ji} = 1 \quad \forall i \in \text{points}$$

This ensures every point is part of one and only one pair.

Solution Obtained

Total Minimized Distance: 61519463.0
Total matched points: 60

Python Script

[13]:
```
#importing the required libraries
import pulp
import itertools
import matplotlib.pyplot as plt
```

[14]:
```
# Reading files and extracting the points.
points = {}
with open('sink.txt', 'r') as file:
    next(file)  # Skip the first line (bounding box boundry)
    for line in file:
        index, x, y = map(int, line.strip().split())
        points[index] = (x, y)
```

[15]:
```
# Calculate Manhattan distances between all pairs of points
distances = {
    (i, j): abs(points[i][0] - points[j][0]) + abs(points[i][1] - points[j][1])
    for i, j in itertools.combinations(points.keys(), 2)
}
```

[16]:
```
# Create the problem variable to contain the problem data
prob = pulp.LpProblem("MinCostMaxMatching", pulp.LpMinimize)

# Decision variables: x_ij = 1 if points i and j are matched, 0 otherwise
x = pulp.LpVariable.dicts("Pair", [(i, j) for i in points for j in points if i <␣
 ↪j], cat=pulp.LpBinary)

# Objective Function: Minimize the sum of distances for the matched points
prob += pulp.lpSum([distances[pair] * x[pair] for pair in distances])

# Constraints: Ensure each point is matched exactly once
for point in points:
```

```
    prob += pulp.lpSum([x[(min(point, other), max(point, other))] for other in␣
 ↪points if point != other]) == 1

# Solve the problem
prob.solve()

# Output results
if pulp.LpStatus[prob.status] == 'Optimal':
    print("Optimal solution found")
    total_minimized_distance = pulp.value(prob.objective)
    print(f"Total Minimized Distance: {total_minimized_distance}")
    matched_pairs = [(i, j) for i, j in distances if x[(i, j)].varValue == 1]
    print("Total matched points: ", len(matched_pairs))
    for pair in matched_pairs:
        print(f"Matched pair: {pair}")
else:
    print("No optimal solution found. Status:", pulp.LpStatus[prob.status])
```

```
Optimal solution found
Total Minimized Distance: 61519463.0
Total matched points:  60
Matched pair: (1, 2)
Matched pair: (3, 14)
Matched pair: (4, 15)
Matched pair: (5, 6)
Matched pair: (7, 18)
Matched pair: (8, 19)
Matched pair: (9, 10)
Matched pair: (11, 22)
Matched pair: (12, 13)
Matched pair: (16, 17)
Matched pair: (20, 21)
Matched pair: (23, 34)
Matched pair: (24, 35)
Matched pair: (25, 36)
Matched pair: (26, 37)
Matched pair: (27, 38)
Matched pair: (28, 29)
Matched pair: (30, 41)
Matched pair: (31, 42)
Matched pair: (32, 43)
Matched pair: (33, 44)
Matched pair: (39, 40)
Matched pair: (45, 46)
Matched pair: (47, 58)
Matched pair: (48, 59)
Matched pair: (49, 60)
Matched pair: (50, 51)
```

```
Matched pair: (52, 63)
Matched pair: (53, 54)
Matched pair: (55, 66)
Matched pair: (56, 57)
Matched pair: (61, 62)
Matched pair: (64, 65)
Matched pair: (67, 68)
Matched pair: (69, 70)
Matched pair: (71, 72)
Matched pair: (73, 74)
Matched pair: (75, 76)
Matched pair: (77, 88)
Matched pair: (78, 89)
Matched pair: (79, 80)
Matched pair: (81, 82)
Matched pair: (83, 84)
Matched pair: (85, 96)
Matched pair: (86, 97)
Matched pair: (87, 98)
Matched pair: (90, 101)
Matched pair: (91, 102)
Matched pair: (92, 93)
Matched pair: (94, 95)
Matched pair: (99, 110)
Matched pair: (100, 111)
Matched pair: (103, 114)
Matched pair: (104, 105)
Matched pair: (106, 107)
Matched pair: (108, 119)
Matched pair: (109, 120)
Matched pair: (112, 113)
Matched pair: (115, 116)
Matched pair: (117, 118)
```

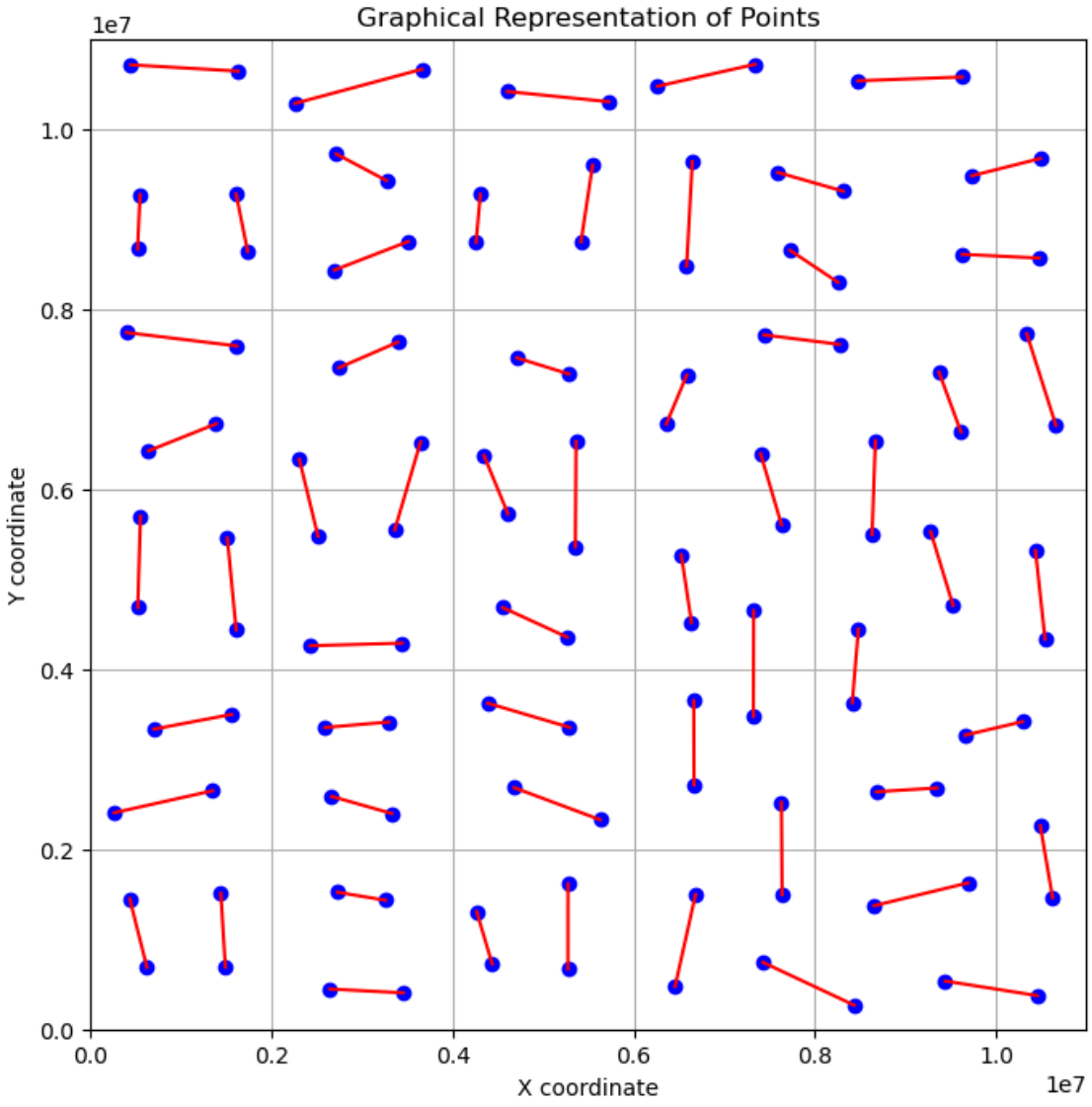
[17]:
```python
# Coordinates extracted earlier from the file
x_vals = [points[i][0] for i in sorted(points)]
y_vals = [points[i][1] for i in sorted(points)]

# Matched pairs extracted from the LP solution
matched_pairs_zero_indexed = [(i-1, j-1) for (i, j) in matched_pairs]


plt.figure(figsize=(8, 8))
plt.scatter(x_vals, y_vals, c='blue', marker='o')
for i, j in matched_pairs_zero_indexed:
    plt.plot([x_vals[i], x_vals[j]], [y_vals[i], y_vals[j]], 'r-')
plt.xlim(0, 11000000)
```

```
plt.ylim(0, 11000000)
plt.xlabel('X coordinate')
plt.ylabel('Y coordinate')
plt.title('Graphical Representation of Points')
plt.grid(True)

# Show the plot
plt.show()
```



[ ]: