Optimization of Water Cooler Placement

Hammond Liu and Shizhao Yang

Abstract. This project investigates the optimal placement strategy of water coolers. Specifically, we convert continuous spatial regions to a discrete network with popularity weights and build an optimization model to find the best spots. We apply the proposed methodology on the 3rd floor of the New Bund campus and obtain some promising placement plans. The code is available at: https://github.com/hmdliu/Optim-SP23.

1 Introduction

1.1 Background

Staying hydrated is a daily necessity. The National Academy of Medicine suggests an adequate intake of daily fluids of 13 cups [1] and 9 cups for healthy men and women, respectively [1]. Correspondingly, exploring effective strategies for ensuring that the locations of water coolers within expansive public edifices can adequately cater to the daily hydration requirements of the individuals who study and work within these premises has emerged as a highly pertinent subject of discussion.

As students, we undoubtedly need enough water intake during lecture intermissions or after a long study session. However, some students may reflect on the inconvenience of the relatively long distance from some open study areas to the nearest water cooler. Therefore, it is both essential and practical to optimize the water cooler placement on the New Bund campus.

1.2 Problem Description

For the sake of time and feasibility, this project will focus on the 3rd floor of the New Bund campus and solve for the optimal placement strategy given a fixed number of water coolers. In particular, we want to minimize the total distance to the nearest water cooler for all on-campus population. In Section 2, we manifest our methodology and present the formal formulation of the problem. In Section 3, we elaborate on our data sources and the processing procedures. Lastly, the findings of the study are consolidated and presented in Section 4.

[1] 1 cup = 8 fl oz or 237 ml.

2 Methodology

2.1 From a Map to a Network

To model continuous spatial region effectively and efficiently, a natural approach is to convert it to a discrete network $G(V, E)$, where the vertices (V) denote different spatial locations and the edges (E) correspond to the distance between the vertices. Accordingly, our objective is to find the best way of placing a fixed number of water coolers on some of the vertices, such that the total distance to the nearest water cooler is minimized for all on-campus population. More details about how we convert the 3rd floor of the New Bund campus to a discrete network are covered in Section 3.1.

In the context of this project, we mainly focus on the shortest path distance to the nearest water cooler vertex as well as the popularity (i.e., the population distribution) of each vertex. The former can be efficiently computed using the Dijkstra's algorithm, which is introduced in Section 2.2; the latter can be obtained through an online survey, which is discussed in Section 3.2.

2.2 Shortest Path Distance

For simplicity, we assume that everyone will follow the shortest path to travel to the nearest water cooler. On a network $G(V, E)$, the shortest path distance refers to the minimum edge weights (i.e., geographical distance) needed to move from a starting vertex to a target vertex. In our case, the starting vertex can be any vertices, and the target vertices are the ones with water coolers. Figure 1 shows a sample network $G(V, E)$. For example, the shortest path from vertex V_1 to vertex V_5 should be $V_1 \to V_2 \to V_5$, with a shortest path distance of 27.

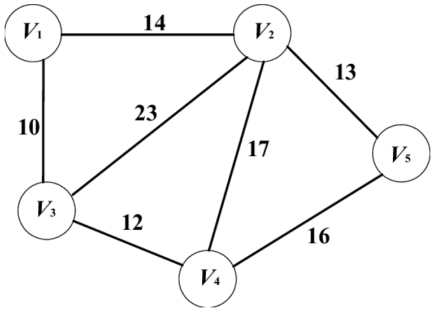


Figure 1: An example of the network $G(V, E)$.

There are several algorithms that can be used to calculate the shortest path distance, such as Dijkstra's algorithm, Bellman-Ford algorithm, and Floyd-Warshall algorithm. We use the Dijkstra's algorithm for this project. Dijkstra's algorithm works by maintaining a priority queue of vertices with the smallest distance. The algorithm starts from the source vertex and explores all the adjacent vertices, updating the distance and adding them to the priority queue if they are not already visited. It then selects the vertex with the smallest distance from the queue and repeats the process until it reaches the

target vertex or all vertices are visited. An implementation of Dijkstra's algorithm (in Python-like pseudo code) is attached below:

```
def dijkstra(graph, start):
    # init distance and the starting vertex
    unvisited = set(graph.keys())
    dist = {v: float('inf') for v in unvisited}
    dist[start] = 0
    # visit all vertices
    while unvisited:
        current = min(unvisited, key=lambda v: dist[v])
        unvisited.remove(current)
        # check all adjacent vertices
        for neighbor, cost in graph[current].items():
            tentative_dist = dist[current] + cost
            if tentative_dist < dist[neighbor]:
                dist[neighbor] = tentative_dist
    return dist
```

We apply the Dijkstra's algorithm to compute the shortest path distance between any pair of vertices (since the water cooler can be placed on any vertices). Note that we are modeling a continuous spatial region, so the converted network should only have one connected component, which means we are able to visit any other vertices from each vertex. By doing so, we can obtain a shortest path distance matrix. Remarkably, this matrix can also be viewed as a strongly connected version of the original graph.

2.3 Model Formulation

In this section, we introduce the parameters and decision variables in the proposed model. We still consider a network $G(V, E)$, where V is the set vertices and E is the set of edges. Let $\boldsymbol{p} \in \mathbb{R}^{|V|}$ denote the popularity vector that describes the popularity distribution over all vertices, where $\boldsymbol{p}_v$ denotes the normalized popularity score at vertex v. Note that vector $\boldsymbol{p}$ is a pre-computed parameter. Let $\boldsymbol{D} \in \mathbb{R}^{|V|\times|V|}$ be the pre-computed shortest path distance matrix (discussed in Section 2.2), then for a vertex pair (u, v), we can denote the shortest path distance between them as $\boldsymbol{D}_{u,v}$. Let K denotes the total capacity of the water coolers.

As for the decision variables, we define a binary assignment matrix $\boldsymbol{a} \in \mathbb{R}^{|V|\times|V|}$, where the first dimension denotes the source vertex u (i.e., any vertices in the network) and the second dimension denotes the target vertex v (i.e., water cooler vertices), and an entry is equal to 1 if the need at vertex u is going to be served by vertex v. Besides, we introduce $\boldsymbol{x} \in \mathbb{R}^{|V|}$, a vector of binary variables that entails whether we place a water cooler at each vertex.

Formally, we can formulate the optimization problem as follows:

$$
\begin{aligned}
\min_{\boldsymbol{a}} \quad & z = \sum_u \sum_v \boldsymbol{p}_u \boldsymbol{a}_{u,v} \boldsymbol{D}_{u,v}, \\
\text{s.t.} \quad & \boldsymbol{a}_{u,v} \in \{0,1\}, \forall\, u, v \in V, \\
& \boldsymbol{x}_v \in \{0,1\}, \forall\, v \in V, \\
& \sum_v \boldsymbol{a}_{u,v} \geq 1, \forall\, u, v \in V, \\
& \boldsymbol{x}_v \geq \boldsymbol{a}_{u,v}, \forall\, u, v \in V, \\
& \sum_v \boldsymbol{x}_v \leq K,
\end{aligned}
$$

where the objective z is to minimize the popularity-weighted shortest path distance; the first two constraints enforce all the entries of $\boldsymbol{a}$ and $\boldsymbol{x}$ to binary; the third constraint ensures each vertex is assigned to at least one water cooler vertex (i.e., all the demands are served); the forth constraint ensures $\boldsymbol{x}_v = 1$ if v will be served as a water cooler vertex ($\boldsymbol{x}_v = 0$ otherwise); the last constraint ensures the total number of allocated water coolers is within the given capacity.

3 Data Description

3.1 Network Building

In the process of fashioning a complete graph, we initially harness the comprehensive map of the 3rd floor as a reference for constructing a preliminary network $G(V, E)$, adhering to the following rules. In general, we establish a vertex in instances where it constitutes an open space (for example, study zones or corridors) or is situated in proximity to the exits of partitioned rooms, thereby accounting for the likelihood of encountering a significant population segment. Moreover, we allocate several vertices uniformly throughout the map, ensuring optimal locations for water cooler installations are considered. In this way, we fashion 54 vertices in total (Figure 2). On top of that, we build 75 edges between adjacent vertices, considering their actual inter-connectivity, while the spatial distance ascribes the corresponding lengths measured with Adobe (an example can be found in Figure 5 of Appendix A). Hence, we successfully assemble a connected network $G(V, E)$.

Then we apply the Dijkstra's algorithm to compute the shortest path distance between all vertex pairs. Figure 3 presents a visualization of the shortest path distance $\boldsymbol{D}$. The ordering of the vertices follows: North $\rightarrow$ East $\rightarrow$ South $\rightarrow$ West. A brighter color indicates a larger shortest path distance, whereas a darker color entails that the two vertices are closer to each other. From the figure, we can clearly observe that the diagonal region is overall darker because the entries along the diagonal corresponds to adjacent vertex pairs, which is geographically closer to each other.

Figure 2: Vertex Distribution of the 3rd Floor

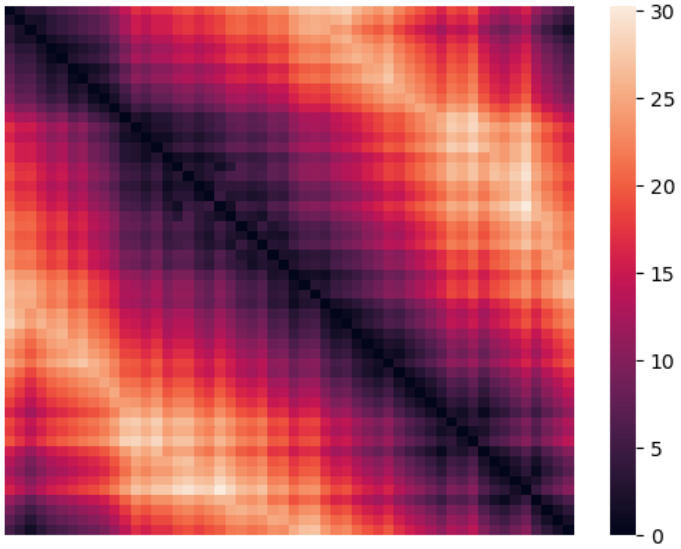


Figure 3: Visualization of the shortest path distance matrix $\boldsymbol{D}$.

3.2 Popularity Modeling

Meanwhile, the population distribution $\boldsymbol{p}$ is estimated by an online survey. The survey consists of three parts: the first part asks the participants to rate their likelihood (from 1 to 5) of staying in a certain open area (e.g., study zones or corridors); the second part requires participants to indicate the number of classrooms they attend and the weekly frequency of their visits to each classroom; the final part inquires about the ratio of time spent in open areas versus classrooms, allowing conversion of classroom visit frequency into likelihood scores. As a result, we've collected a total amount of 68 effective surveys from all classes of NYU Shanghai students. We compute the mean of likelihood scores for both open areas and classrooms after data collection, then normalize these scores to a 0 to 1 scale utilizing the following formula:

$$\tilde{S}_v = \frac{S_v - S_{min}}{S_{max} - S_{min}}$$

where S_v denotes the original likelihood score for vertex v.

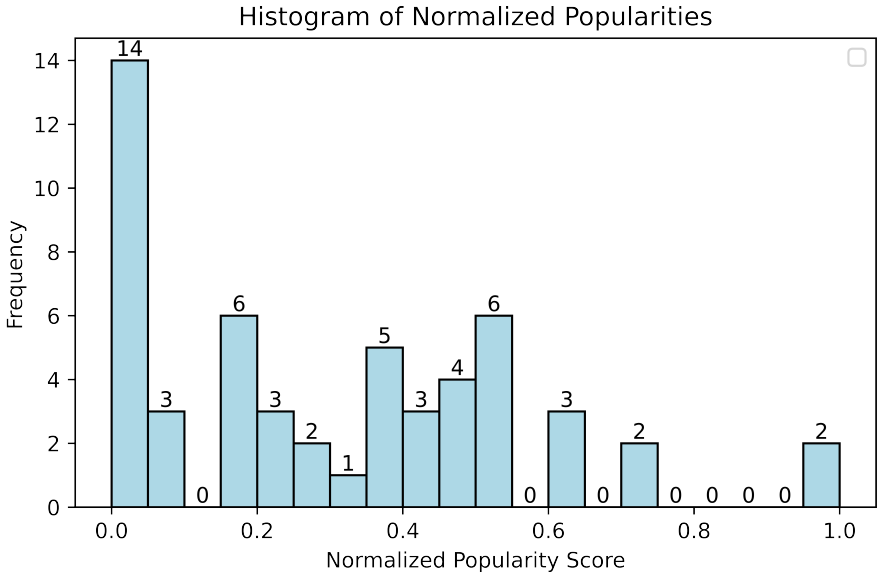


Figure 4: Histogram of the normalized popularity scores.

As depicted in the histogram of normalized popularity scores in Figure 4, the scores exhibit a long-tail distribution. There are 14 out of 54 vertices possessing low popularity scores ranging between 0 and 0.05, suggesting minimal student presence in these areas, which often include corridors or secluded corners. Conversely, two vertices ($S03$ and $E14$) boast significantly high scores nearing 1. These vertices are situated near classrooms with a high weekly lecture frequency (e.g., $E304$) and in proximity to clustered open study spaces.

4 Results & Discussion

We implement the proposed optimization model using GAMS and solve for the optimal placement plans when $K = 1, 2, 3, 4, 5$. The optimal water cooler vertices are shown in Table 1. We also highlight their corresponding positions on the map, which can be found in Appendix B. We also attach the GAMS code in Appendix C.

Capacity (K)	Optimal Water Cooler Vertices
K = 1	E12
K = 2	E12, W07
K = 3	N04, E12, **S15**
K = 4	**N09**, E12, **S15**, W09
K = 5	**N09**, E08, S01, **S15**, W09

Table 1: Optimal water cooler vertices. The vertices shown in **bold** are the ground-truth water cooler positions (i.e., water coolers physically present at these spots).

Broadly speaking, the Magnolia Hall in the east building is the most popular spot on the 3rd floor for classes and self-studying, so it is reasonable to place the water cooler at vertex $E12$ when $K = 1$. When $K = 2$, the second water cooler is placed at vertex $W07$, which is a popular study area that is opposite to vertex $E12$. When $K = 3$, the three water coolers roughly form a equilateral triangle on the map. When K is set to 4 or larger, there will be at least one water cooler being placed in each building. Overall, the placement plans obtained via the proposed model are quite promising.

Furthermore, it is worth mentioning that the proposed model is able to provide insightful suggestions for all sorts of placement problem, such as vending machines and restrooms. Taking the case of $K = 4$ as an example, the model suggests to place the water coolers at vertex $N09, E12, S15$, and $W09$. Interestingly, three of them ($N09, E12, S15$) are precisely the true positions of the vending machines.

The future work of this project mainly lies in the data perspective. On the one hand, the network building stage can be further polished to better abstract the spatial structure; on the other hand, collecting more high-quality data can further improve the accuracy of the placement suggestions, which may potentially involve designing more sophisticated survey questions and inviting more participants.

Acknowledgement

We thank Professor Zhibin Chen for his suggestions on the project.

References

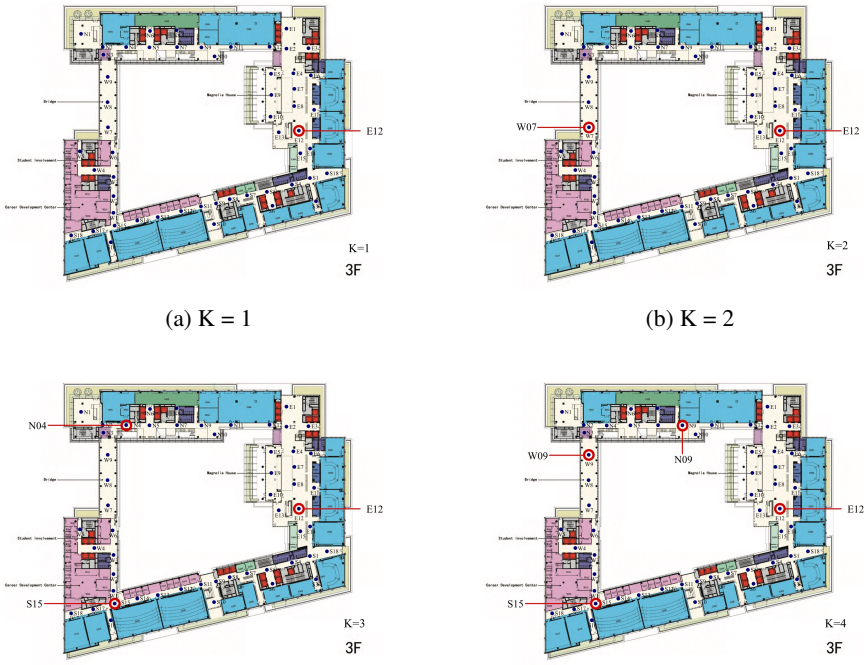
[1] *Dietary Reference Intakes for Water, Potassium, Sodium, Chloride, and Sulfate*. National Academies Press, May 2005.

Appendix A: Adjacency List for the Network

name	index	neighbors
N01	0	N02(1.9)
N02	1	N01(1.9)|N03(2.0)|N04(1.8)
N03	2	N02(2.0)|N04(1.9)|W09(1.5)
N04	3	N02(1.8)|N03(1.9)|N05(1.6)
N05	4	N04(1.6)|N06(0.9)|N07(1.8)
N06	5	N05(0.9)
N07	6	N05(1.8)|N08(0.9)|N09(1.6)
N08	7	N07(0.9)
N09	8	N07(1.6)|N10(1.1)
N10	9	N09(1.1)|N11(2.0)
N11	10	N10(2.0)|N12(2.6)
N12	11	E01(2.6)|E02(1.6)|N11(2.6)
E01	12	E02(1.3)|N12(2.6)
E02	13	E01(1.3)|E03(1.4)|E04(1.4)|N12(1.6)
E03	14	E02(1.4)
E04	15	E02(1.4)|E05(1.4)|E06(2.0)|E07(1.0)
E05	16	E04(1.4)|E06(3.1)|E07(2.2)|E09(1.5)
E06	17	E04(2.0)|E05(3.1)|E07(2.2)
E07	18	E04(1.0)|E05(2.2)|E06(2.2)|E08(1.0)
E08	19	E07(1.0)|E10(2.8)|E11(3.0)|E12(2.2)|E13(2.7)
E09	20	E05(1.5)|E10(1.7)

Figure 5: Example of the adjacency list that represents $G(V, E)$.

Appendix B: Map View of the Optimal Solutions



(a) K = 1 (b) K = 2

(c) K = 3 (d) K = 4

Figure 6: Map view of the optimal solutions.

Appendix C: GAMS Implementation

```
set
    v_n     north       /N01*N12/
    v_e     east        /E01*E15/
    v_s     south       /S01*S18/
    v_w     west        /W01*W09/
    v       vertices    /#v_n, #v_e, #v_s, #v_w/;
alias (v, u);
parameter
    K           num of water coolers
    p(v)        normalized popularity vector
    D(v, u)     shortest path distance matrix;
variable
    z           objective;
binary variable
    a(v, u)     assignment matrix
    sol(u)      water cooler placement;

* load distance matrix
$CALL GDXXRW.EXE D.xlsx par=D rng=A1:BC55
$GDXIN D.gdx
$LOAD D
$GDXIN
display D;
* load popularity vector
$CALL GDXXRW.EXE p.xlsx par=p rdim=1 rng=A1:B54
$GDXIN p.gdx
$LOAD p
$GDXIN
display p;
* init num of water coolers
K = 3;

* define equations
equations
    obj             total popularity-weighted distance
    cap_cons        capacity constraint
    ser_cons(v)     serving constraint
    sol_cons(v, u)  sol constraint;
obj.. z =e= sum((v, u), p(v) * a(v, u) * D(v, u));
sol_cons(v, u).. sol(u) =g= a(v, u);
ser_cons(v).. sum(u, a(v, u)) =g= 1;
cap_cons.. sum(u, sol(u)) =l= K;

* solve the problem
model placement /all/;
solve placement using mip minimizing z;
display z.l, sol.l;
```