Wells

14th November 2017

Hüseyin Utku ASLAN, Tou AMMAR

PROJECT DESCRIPTION

Our purpose in this project is to find a solution and implementing a that solution.

PROBLEM DESCRIPTION

Design and implement an algorithm to find the cheapest (total sum cost) way to provide houses with water. There are given **n** wells and **kn** houses (points on plane), each well provides **k** houses with water, where n, k and kn are positive integers.

The distance with any given two points will be calculated by Euclidean distance formula. x and y are two given points in two dimensional space. Distance between them calculated as follows:

$$d(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

Total cost is the summation of all the distances between connections.

$$cost = \sum_{w \, \varepsilon \, wells} \; \sum_{h \, \varepsilon \, wells.connected} d(w,h)$$

GOAL

1. Connect every well with k number of houses
2. Find minimum total cost

SOLUTION DESCRIPTION

For bipartite matching, Hungarian algorithm gives good results. Hungarian algorithm uses a weighted(distance) matrix that shows distances between each well and house. Hungarian

method is good for square matrix and we need to modify our problem according to that. Shape of the matrix will be **[kn X kn].** For that we copied lines over and over again till we get a square matrix.

Steps:

1. Create square matrix.
2. Row and column reduction.
3. Draw a minimum number of lines to cover all zeros of the matrix. (Row and column scanning)
4. Identify the minimum value of the non deleted cell values.
5. Select these rows (houses) and add their value to total sum. If there are only k number of rows, finish the task

Pseudo Code:

WeightMatrix(wells,houses):

1. $for\ i \leftarrow 0 : wells.size$
2. $\quad for\ j \leftarrow 0 : houses.size$
3. $\quad\quad for\ x \leftarrow i;\ i < houses.size;\ x \leftarrow x + houses.size\ /\ wells.size$
4. $\quad\quad\quad w_{xj} \leftarrow distance(i, j)$

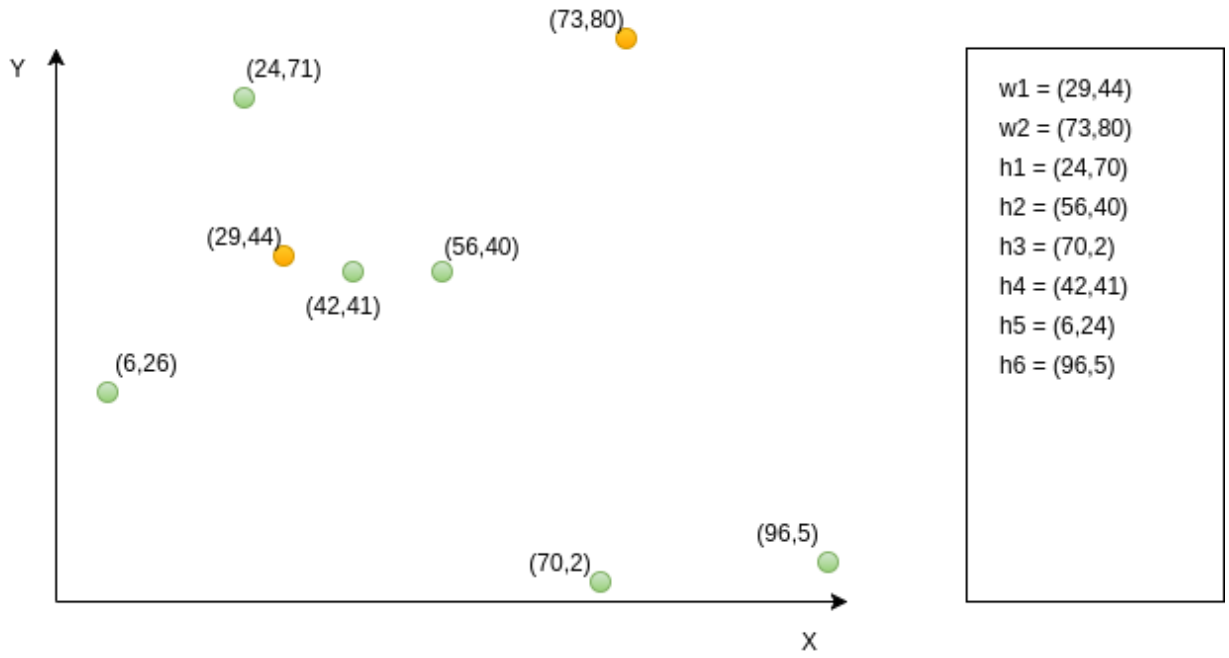
Time complexity : $O((kn)^2)$

1. $r \leftarrow \oslash$
2. $w \leftarrow WeightMatrix(wells, houses)$
3. $w \leftarrow rowReduction(w)$
4. $w \leftarrow columnReduction(w)$
5. $found \leftarrow False$
6. $while\ not\ found$
7. $\quad if\ numberOfZeroes\ is\ numberOfRows$
8. $\quad\quad found \leftarrow True$
9. $\quad\quad r \leftarrow r + zeroes$
10. $\quad else$
11. $\quad\quad for\ i \leftarrow 0 : wells.size$
12. $\quad\quad\quad r \leftarrow r + min(corossing(w))$
13. $\quad\quad\quad w \leftarrow w.delete(min(corossing(w)))$
14. $return\ r$

Overall time complexity $O((kn)^3)$

EXAMPLE

Let $w_1, w_2 \, \varepsilon \, wells$ and $h_1, h_2, h_3, h_4, h_4, h_5, h_6 \, \varepsilon \, houses$. Thus every well should have 3 connections.



Distance matrix :

	w_1	w_2	w_1	w_2	w_1	w_2
h_1	25.46	49.82	25.46	49.82	25.46	49.82
h_2	27.29	43.46	27.29	43.46	27.29	43.46
h_3	58.69	78.06	58.69	78.06	58.69	78.06
h_4	13.34	49.82	13.34	49.82	13.34	49.82
h_5	29.21	86.05	29.21	86.05	29.21	86.05
h_6	77.52	78.45	77.52	78.45	77.52	78.45

After row reduction :

	w_1	w_2	w_1	w_2	w_1	w_2
h_1	0	22,36	0	21.44	0	21.44
h_2	0	16,17	0	15.24	0	15.24
h_3	0	19,36	0	18.44	0	18.44
h_4	0	36,48	0	35.55	0	35.55
h_5	0	56,84	0	55.92	0	55.92
h_6	0	0,9232	0	0	0	0

After column reduction :

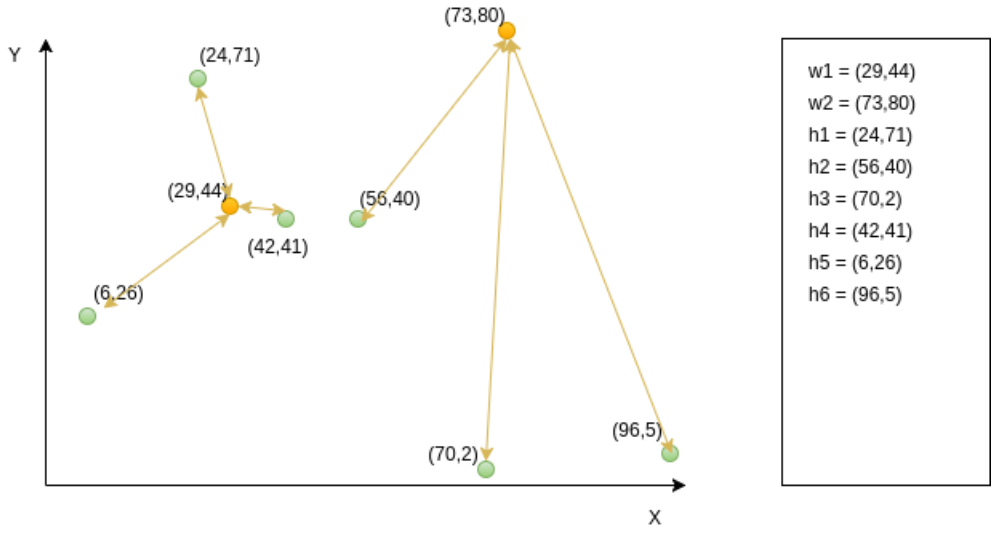
	w_1	w_2	w_1	w_2	w_1	w_2
h_1	0	21.44	0	21.44	0	21.44
h_2	0	15.24	0	15.24	0	15.24
h_3	0	18.44	0	18.44	0	18.44
h_4	0	35.55	0	35.55	0	35.55
h_5	0	55.92	0	55.92	0	55.92
h_6	0	0	0	0	0	0

After line cover :

	w_1	w_2	w_1	w_2	w_1	w_2
h_1	0	21.44	0	21.44	0	21.44
h_2	0	15.24	0	15.24	0	15.24
h_3	0	18.44	0	18.44	0	18.44
h_4	0	35.55	0	35.55	0	35.55
h_5	0	55.92	0	55.92	0	55.92
h_6	0	0	0	0	0	0

Result:

h_2, h_3 and h_6 will be connected to w_2 and h_1, h_4 and h_5 will be connected to w_1. Total distance is sum(X) + sum(Y) = 68,01 + 199,97 = 267,98 which is minimum distance.



CORRECTNESS[1]

1. The dual solution is always feasible. Note that u, v are clearly dual feasible in the beginning. Now, assume that u, v are dual feasible in the beginning of some iteration. They will only be changed during that iteration if there is no perfect matching in G0. Feasibility simply means nonnegativity of reduced costs for all edges. Note that the only reduced costs that decrease are for edges (i, j) where $i \in A \cap L$ and $j \in B \backslash L$. However, note that the reduced costs of those edges decrease by exactly δ, and by definition of δ equals the minimum of them; thus, none of them drop below 0 and dual feasibility is maintained.

2. Complementary slackness is maintained. Trivial: note that the matching is found only along edges with zero reduced cost. (Note that this is true at the point in iteration immediately after M is computed, which is what we need since this is the point where we will eventually exit the loop).

3. The algorithm reaches an ending. First, note that δ always stores a strictly positive number. If δ gets the value 0 at some point, then there is an edge (i, j) for which $i \in A \cap L$, $j \in B \backslash L$, and $w_{ij} = 0$; however, since $w_{ij} = 0$, then $(i, j) \in E0$, and j is reachable from $A \cap L$ or i was reached via (i, j) (if $(i, j) \in M$), so $j \in L$, which is not possible. Next, recall that after we run the bipartite (unweighted) matching algorithm, $C = (A \backslash L) \cup (B \cap L)$ is a vertex cover of the same cardinality as the matching M we find. Note that in every iteration, the dual increase is $\delta|A \cap L| - \delta|B \cap L| = \delta(|A \cap L| + |A \backslash L| - |A \backslash L| - |B \cap L|) = \delta(|A| - |C|) = \delta(n2 - |M|) \geq \delta$ where we get $n2 - |M| \geq 1$ from the fact that M is not perfect. Thus, each iteration increases the dual by at least δ, which is at least 1 since the data is integral; since the primal problem is feasible (there exists a perfect matching in the original graph), the dual

problem is bounded, and so the dual solution value cannot increase forever and the algorithm must terminate.

INPUT AND OUTPUT DESCRIPTION

Input and output in the final product will be a text file. Text files have the number of wells and houses, and all of their connections. Semicolon will separate data.

There are 2 inputs in the header; number of houses and number of wells. Right format is %;h;numberOfHouses;w;numberOfWells;

Remaining of the file should contain points, types and id. Correct format is Type;XAxis;YAxis;Id;

An example of input file:
%;h;6;w;2;
W;29;44;w1;
W;73;80;w2;
H;24;71;h1;
H;56;40;h2;
H;70;2;h3;
H;42;41;h4;
H;6;26;h5;
H;96;5;h6;

Output files header is the same as input files header but with addition of total cost. Remaining data shows connections and distance between them. Correct format is Well_Id;House_Id;distance;

An example of output file:
%;h;2;w;2;10;
w1;h1;5;
w2;h2;5

SOURCE

1.IEOR 8100: Matchings Lecture 6: The Hungarian Algorithm (Columbia University) (http://www.columbia.edu/~cs2035/courses/ieor8100.F12/lec6.pdf)