

Simulation Study on Outdoor Wind Environment of Residential Complexes in Hot-Summer and Cold-Winter Climate Zones Based on Entropy-Based TOPSIS Method

Xiang Liu, Wanjiang Wang *, Zixuan Wang, Junkang Song and Ke Li

School of Architecture and Engineering, Xinjiang University, Urumqi 830047, China;
107552201576@stu.xju.edu.cn (X.L.); 107552201582@stu.xju.edu.cn (Z.W.);
junkangsong@stu.xju.edu.cn (J.S.); 107552101504@stu.xju.edu.cn (K.L.)

* Correspondence: wangwanjiang@xju.edu.cn

Figure S1: Entropy-Based TOPSIS Method related codes

```
1. %% Step 1: Copy the data to the workspace and name this matrix X
2. % (1) Right click in the workspace, click New (Ctrl+N), enter the
   variable name as X
3. % (2) Copy the data in Excel, then go back to Excel and right click,
   click Paste Excel Data (Ctrl+Shift+V)
4. % (3) Close this window, click on the X variable, right-click Save As,
   and save as a mat file (next time you won't have to copy and paste,
   just use the load command to load the data)
5. % (4) Note that the code and data should be in the same directory,
   and the current folder of Matlab should be the same directory
6.
7. clear;clc
8. load data_water_quality.mat
9.
10.%% Step 2: Determine if forwarding is required
11.[n,m] = size(X);
12.disp(['there are ' num2str(n) 'evaluation objects, ' num2str(m)
   'evaluation metrics'])
13.Judge = input(['Does this ' num2str(m) 'indicator need to be normalized,
   enter 1 if it does and 0 if it doesn't: ']);
14.
15.if Judge == 1
16.    Position = input('Please enter the columns of the indicators that
   need to be normalized, for example, columns 2, 3, and 6 need to be
   processed, then you need to enter [2,3,6]: '); %[2,3,4]
17.    disp('Please enter the indicator types for these columns that
   need to be processed (1: very small, 2: intermediate, 3: interval) ')
18.    Type = input('For example, if column 2 is Extremely Small, column
   3 is Interval, and column 6 is Intermediate, enter [1,3,2]: ');
   %[2,1,3]
19.    % Note that Position and Type are two row vectors of the same
   dimension
```

```

20.     for i = 1 : size(Position,2) % Here these columns need to be
        processed separately, so we need to know the total number of ti
        mes we have to process them, i.e., the number of loops
21.         X(:,Position(i)) = Positivization(X(:,Position(i)),Type(
            i),Position(i));
22.     % Positivization is a function that we define to do the norm
        alization, and it takes three arguments.
23.     % The first argument is the vector of columns to be positiv
        ized, X(:,Position(i)) Recall from the previous lecture that X(:,
        n) takes all the elements of column n. % The second argument is
        the vector of the corresponding columns, X(:,Type(i),Position(i)
        ); % Positivization is a function we defined ourselves.
24.     % The second argument is the type of indicator that correspo
        nds to the column (1: very small, 2: intermediate, 3: interval)
25.     % The third argument tells the function which column of the
        original matrix we are dealing with.
26.     % The function has a return value, which returns the normali
        zed indicator, which we can assign directly to the original vect
        or of columns we are working with.
27.     end
28.     disp('Normalized matrix X = ')
29.     disp(X)
30.end
31.%% Let the user decide if the weight needs to be increased or no
    t
32.disp('Please enter whether you need to increase the weight vecto
    r, 1 if you do, 0 if you don't')
33.Judge = input('Please enter whether you need to increase the wei
    ghts: ');
34.if Judge == 1
35.    disp(['If you have 3 metrics, you need to enter 3 weights, f
        or example if they are 0.25,0.25,0.5, then you need to enter [0.
        25,0.25,0.5]']);
36.    weigh = input(['You need to enter' num2str(m) 'weights.' 'Pl
        ease enter this' num2str(m) 'number of weights as a row vector:'
        ]);
37.    OK = 0; % Use to determine if the user input is in the corre
        ct format.
38.    while OK == 0
39.        if abs(sum(weigh) - 1)<0.000001 && size(weigh,1) == 1 &&
            size(weigh,2) == m % Note that floating point numbers are impre
            cise.
40.            OK =1; else
41.            OK =1; else
42.                weigh = input('There was an error in your input, ple
                    ase re-enter the weight row vector: ');
43.            end
44.        end
45.end
46.    weigh = ones(1,m) . / m ; %If no weights are needed, the def
        ault weights are all the same, i.e. all 1/m.
47.end

```

```

48.
49.
50.%% Step 3: Normalize the normalized matrix
51.Z = X . / repmat(sum(X.*X) . ^ 0.5, n, 1);
52.disp('Normalized matrix Z = ')
53.disp(Z)
54.
55.%% Step 4: Calculate the distance to the maximum value and the d
    istance to the minimum value and calculate the score
56.D_P = sum([(Z - repmat(max(Z),n,1)) . ^ 2 ] . * repmat(weigh,n,1
    ),2) . ^ 0.5; % D+ distance vector from max
57.D_N = sum([(Z - repmat(min(Z),n,1)) . ^ 2 ] . * repmat(weigh,n,1
    ),2) . ^ 0.5; % D- Distance vector to the minimum value
58.S = D_N . / (D_P+D_N); % Unnormalized score
59.disp('Final score:')
60.stand_S = S / sum(S)
61.[sorted_S,index] = sort(stand_S , 'descend')

```

Positivization code

```

1. function [posit_x] = Positivization(x,type,i)
2. % There are three input variables:
3. % x: the original column vector corresponding to the indicator t
    o be positivized
4. % type: type of the indicator (1: very small, 2: intermediate, 3
    : interval)
5. % i: which column of the original matrix is being processed
6. % The output variable posit_x represents: the normalized column
    vector.
7.     if type == 1 % extremely small
8.         disp(['First' num2str(i) 'Column is extremely small and
    is being normalized'] )
9.         posit_x = Min2Max(x); %call Min2Max function to normaliz
    e
10.        disp(['First' num2str(i) 'Columns are extremely small an
    d are being normalized'] )
11.        disp('~~~~~ divider ~~~~~')
    )
12.    elseif type == 2 % intermediate type
13.        disp(['First' num2str(i) 'Column is intermediate'] )
14.        best = input('Please enter the value of the best one: ')
    ;
15.        posit_x = Mid2Max(x,best);
16.        disp(['First' num2str(i) 'Column intermediate type norma
    lization processing complete'] )
17.        disp('~~~~~ dividing line ~~~~~')
    )
18.    elseif type == 3 % interval type
19.        disp(['First' num2str(i) 'Column is interval type'] )
20.        a = input('Please enter the lower bound of the interval:
    ');

```

```

21.         b = input('Please enter the upper bound of the interval:
                ');
22.         posit_x = Inter2Max(x,a,b);
23.         disp(['First' num2str(i) 'Column interval type normaliza
                tion processing complete'] )
24.         disp('~~~~~ divider ~~~~~'
                )
25.     else
26.         disp('There are no indicators of this type, please check
                if there are values other than 1, 2, 3 in the Type vector')
27.     end
28. end

```

Min2Max code

```

1. function [posit_x] = Min2Max(x)
2.     posit_x = max(x) - x; %posit_x = 1 .
3.     %posit_x = 1 . / x; %If all of x is greater than 0, it can
        be positively normalized this way as well.
4. end

```

Mid2Max code

```

1. function [posit_x] = Mid2Max(x,best)
2.     M = max(abs(x-best));
3.     posit_x = 1 - abs(x-best) / M;
4. end

```

Inter2Max code

```

1. function [posit_x] = Inter2Max(x,a,b)
2.     r_x = size(x,1); % row of x
3.     m = max([a-min(x),max(x)-
        b]); posit_x = zeros(r_x,1); % row of x
4.     posit_x = zeros(r_x,1); % zeros function usage: zeros(3) zer
        os(3,1) ones(3)
5.     % Initialize posit_x to all zeros Initialization is done to
        save processing time.
6.     for i = 1: r_x
7.         if x(i) < a
8.             posit_x(i) = 1-(a-x(i))/M;
9.         elseif x(i) > b
10.            posit_x(i) = 1-(x(i)-b)/M; elseif x(i) > b
11.            posit_x(i) = 1-(x(i)-b)/M; else
12.            posit_x(i) = 1; elseif x(i) > b
13.        end
14.    end
15. end

```

Table S1. Butterfly Mesh Setting

Butterfly Table Settings			
Mesh segmentation of feature edges and feature surfaces	castellatedMesh	Do you want to cut the grid?	TRUE
	locationInMesh	The reserved domain (position vector) of the grid cannot coincide with the faces or edges of the grid element	Set within the field
	maxGlobalCells	The maximum number of meshes to refine before removing the mesh	4000000
	nCellsBetweenLevels	Number of buffer layers during refinement at different levels	10
	resolveFeatureAngle	Apply the maximum level of refinement when the bending angle of a face or edge exceeds this angle	90
Mesh refinement	refineLevel	Before surface fitting, output indicators such as refinement level	(2,4)
	globRefineLevel	Global cell division level	(0,1)
Surface alignment	snap	Is grid fitting (alignment) performed?	TRUE
Grid Boundary layer	addLayers	Add mesh Boundary layer?	FALSE
	expensionRatio	Boundary layer mesh expansion factor	1.5
	finalLayerThickness	The nearest Boundary layer thickness of the wall surface is set as a relative or absolute value through relativeSizes	0.1
	minThickness	Minimum thickness of Boundary layer grid (relative or absolute value, as above)	0.01
	nLayerIter	Maximum Boundary layer addition iteration number	50
Grid quality control	meshSet	Grid Quality Control Dictionary	smoothMeshSettings