Separating Lines using Local Optimization method

OPERATION: Separate n-points in a graph with axis-parallel lines using Local Optimization approach.

INPUT: An integer "max_files" which has the number of input instances to be read. The program will read instances files each containing the co-ordinates of the points. The points are sorted along x-axis and they won't share each other's axes.

OUPUT: A Feasible solution containing the set 'S' of axis-parallel lines that separate the given instance points such that |S| is minimum. The solutions are printed in a file.

/***/

Explaination

A random feasible solution is assumed at the start which has a axis parallel verical and axis parallel horizontal lines between every two points.

A connection is formed between all the points. After that a combination of two lines is compared with the third line from this feasible solution. Then the solution is checked for feasibility after removing two

line and adding the third one. The solution is checked for feasibility by removing all the connections that the lines(other than first two) are crossing. If the

graph is left with no connections then the solution is said to be feasible and the two lines are replaced with the third line.

/***/

Pseudocode

```
1.  SEPARATE_POINTS_LOCAL_OPTIMIZATION () {
2.  // checking combination of two verical lines with a third line(horizontal or vertical)
3.   for (first = 0 to (vertical lines count - 1)) {   //select first vertical line for comparison
4.    for (second = i+1 to (vertical lines count)) {  //select second vertical line for comparison
5.     for (third = second+1 to (vertical lines count)) {   //select third vertical line for comparison
6.      // check feasibilty of the solution
7.      remove all the edges crossing a line other than first and second one by one.
8.      remove all the edges for the crossing third line.
9.      if (no connections found) {
10.       remove first and second lines from the feasible solution.
11.       add third line in feasible solution.
12.       break;
13.     }
14.    }
```

```
15.       if (replacement not found) {
16.         for (third = 0 to (horizontal lines count)) {  // select third horizontal line for comparison
17.           // check feasibilty of the solution
18.           remove all the edges crossing a line other than first and second one by one.
19.           remove all the edges for the crossing third line.
20.           if (no connections found) {
21.             remove first and second lines from the feasible solution.
22.             add third line in feasible solution.
23.             break;
24.           }
25.         }
26.       }
27.       if (replacement not found) {
28.         add first and second lines to the feasible solution
29.       }
30.     }
31.   }
32.    // checking combination of two horizontal lines with a third line(horizontal or vertical)
33.   for (first = 0 to (horizontal lines count - 1)) {  //select first horizontal line for comparison
34.     for (second = i+1 to (horizontal lines count)) {  //select second horizontal line for comparison
35.       for (third = 0 to (vertical lines count)) {  //select third vertical line for comparison
36.         // check feasibilty of the solution
37.         remove all the edges crossing a line other than first and second one by one.
38.         remove all the edges for the crossing third line.
39.         if (no connections found) {
40.           remove first and second lines from the feasible solution.
41.           add third line in feasible solution.
42.           break;
43.         }
44.       }
45.       if (replacement not found) {
```

```
46.         for (third = second+1 to (horizontal lines count)) {  //select third horizontal line for comparison
47.           // check feasibilty of the solution
48.           remove all the edges crossing a line other than first and second one by one.
49.           remove all the edges for the crossing third line.
50.           if (no connections found) {
51.             remove first and second lines from the feasible solution.
52.             add third line in feasible solution.
53.             break;
54.           }
55.         }
56.       }
57.       if (replacement not found) {
58.         add first and second lines to the feasible solution
59.       }
60.     }
61.   }
62.   // checking combination of a horizontal line and a vertical line with a third line(horizontal or vertical)
63.   for (first = 0 to (vertical lines count - 1)) {  //select first vertical line for comparison
64.     for (second = i+1 to (horizontal lines count)) {    //select second horizontal line for comparison
65.       for (third = 0 to (vertical lines count)) {  //select third vertical line for comparison
66.         // check feasibilty of the solution
67.         remove all the edges crossing a line other than first and second one by one.
68.         remove all the edges for the crossing third line.
69.         if (no connections found) {
70.           remove first and second lines from the feasible solution.
71.           add third line in feasible solution.
72.           break;
73.         }
74.       }
75.       if (replacement not found) {
76.         for (third = 0 (horizontal lines count)) { // select third horizontal line for comparison
```

```
77.          // check feasibilty of the solution
78.          remove all the edges crossing a line other than first and second one by one.
79.          remove all the edges for the crossing third line.
80.          if (no connections found) {
81.            remove first and second lines from the feasible solution.
82.            add third line in feasible solution.
83.            break;
84.           }
85.         }
86.       }
87.       if (replacement not found) {
88.         add first and second lines to the feasible solution
89.       }
90.     }
91.   }
92.    Print the solution to the output file
93.  }
```

/***/

Running time analysis

Three for for checking combination of two vertical lines -> O(n^3)

Three for for checking combination of two horizontal lines -> O(n^3)

Three for for checking combination of a vertical line and a horizontal line -> O(n^3)

Checking feasibilty method -> O(n^2)

Hence the running time of the above palgorithm is O(n^6)

/***/

Example where this algorithm doesn't work

This algorithm doesnt work when the x coordinates and y coordinates are not in the range of total number of points.

So for example if the total number of points are 5 (1,2),(4,7),(3,3),(2,1),(5,5).

As you can see the y coordinates have point 7 which is greater than 5.

In this case, the very first random feasible solution will have x lines intersecting at X axis at 1.5, 2.5, 3.5, and 4.5.

And the y lines intersecting Y axis at 1.5, 2.5 and 3.5. Since 7 is out of range, the points 5,5 and 4,7 wouldnt have any

y lines dividing them. Hence there will be no line dividing these two points.

/**/

References:

http://katrinaeg.com/simulated-annealing.html

http://ieeexplore.ieee.org.ezproxy.gl.iit.edu/stamp/stamp.jsp?tp=&arnumber=6921823&tag=1

http://www.theprojectspot.com/tutorial-post/simulated-annealing-algorithm-for-beginners/6