Steiner trees

Due: June 5, 2024

Problem Description

The Steiner Tree Problem is known to be NP-hard, which means that it is unlikely that there is an efficient algorithm that can solve the problem for large instances. However, there are several approximation algorithms that can provide a suboptimal solution with a guarantee on the quality of the solution. Given a graph and a subset of vertices in the graph, a Steiner tree spans through the given subset. The Steiner Tree may contain some vertices which are not in the given subset but are used to connect the vertices of the subset. The given set of vertices is called Terminal Vertices and other vertices that are used to construct the Steiner tree are called Steiner vertices. The Steiner Tree Problem is to find the minimum cost of Steiner Tree.

Please refer to the following RISC-V assembly programs given in
https://hackmd.io/@linyu413/riscv_2024hw3_1 and
https://hackmd.io/@linyu413/riscv_2024hw3_2 .
Please make the first line of the printed message be your student ID number.
You should use RISC-V instruction set simulator **RARS** to develop and execute the assembly code.
The RARS simulator can be downloaded from https://www.rose-hulman.edu/class/csse/csse232/Lab1/rars_27a7c1f.jar .

What Should Be Handed In:

- **The first line of assembly code should consist of your student ID number**. The file name of the assembly code should be **sID.asm** where ID is your student ID number. A valid file name will look like s1111111.asm .
- A clip like the one shown in the example of input and output below. Save the clip as a file called **sID.png**, where ID is your student ID number. A valid file name for an output clip will look like s1111111.png .
- The homework will not be graded if you do not follow the above rules.

Grade point rules:

Basic points: (100%)

Problem Description:

1. The number of graph nodes will not exceed 9

2.The X and Y coordinate range will only be integers between 0 and 9

Evaluation rules:

W_MR: The total wire length of minimum rectilinear spanning tree.

W_Best: The total wire length of best results among all students.

W_Yours: The total wire length of your result.

Score = (100*W_MR-80*W_Best-20*W_Yours)/(W_MR-W_Best)

W_MR	W_Best	W_Yours	Score
1000	600	600	100
1000	600	1000	80
1000	600	1100	75
1000	900	2000	-120
10	5	100	-280
900	1	600	86.67408
900	400	400	100

Extra points: (10%)

Can handle situations where the coordinate values are greater than or equal to 10.

Example of input and output:

Your program must be able to read an input file in a given format and write the results to an output file.

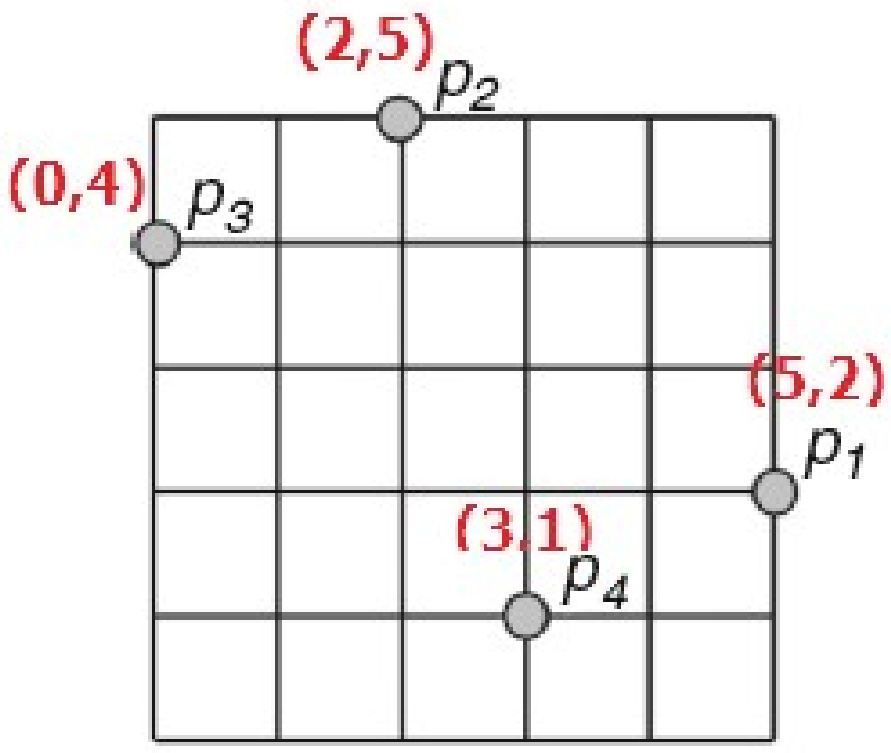
```
s111111
Please enter the input file name = 1.txt
The output file name of result = s111111_output.txt
```

Input file format:

PointsNumber,Point1_X, Point1_Y, Point2_X, Point2_Y…..

Input file sample:

4,5,2,2,5,0,4,3,1



Output file format:

PointStart1_X, PointStart1_Y, PointEnd1_X, PointEnd1_Y

PointStart2_X, PointStart2_Y, PointEnd2_X, PointEnd2_Y

PointStart3_X, PointStart3_Y, PointEnd3_X, PointEnd3_Y

….

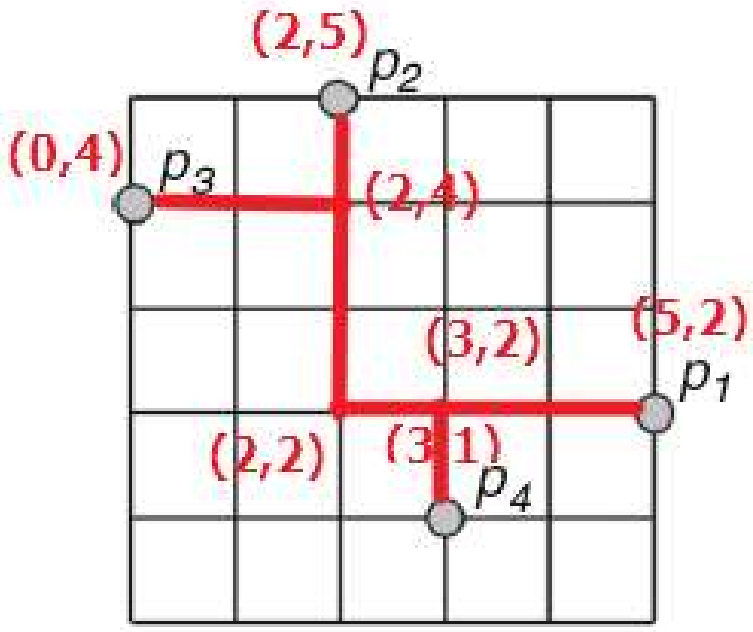
Output file sample:

0,4,2,4

2,5,2,2

2,2,5,2

3,1,3,2



One popular concurrent approach is to formulate global routing as a ***0-1 integer linear programming*** (0-1 ILP) problem. The layout is first modeled as a routing graph $G(V, E)$, where each node represents a tile and each edge denotes the boundary between two adjacent tiles. Each edge $e \in E$ is assigned a capacity, denoted by c_e, which represents the number of tracks crossing that boundary. Given a net, all of its possible routing patterns can be enumerated. Let the variable $x_{i,j} \in \{0,1\}$ indicate whether the routing pattern $r_{i,j}$ is selected from the set R_i of routing patterns of net n_i. Consequently, for a routing graph $G(V, E)$ with the netlist N, the congestion-driven global routing can be formulated as a 0-1 ILP problem as follows:

$$\begin{array}{lll} \text{Minimize} & \lambda & \\ \text{Subject to} & \sum_{r_{i,j} \in R_i} x_{i,j} = 1, & \forall n_i \in N \\ & x_{i,j} \in \{0,1\}, & \forall n_i \in N, \forall r_{i,j} \in R_i \\ & \sum_{i,j:e \in r_{i,j}} x_{i,j} \leq \lambda c_e, & \forall e \in E \end{array} \tag{12.1}$$

The first and the second constraints require that only one routing pattern can be chosen for each net, and the third constraint with the objective together ensure to minimize the maximum congestion. If a solution of $\lambda \leq 1$ exists, a global-routing solution with the maximum congestion being minimized can be achieved.

Because the 0-1 ILP is NP-complete, the high time complexity greatly limits the feasible problem size. An alternate approach to this problem is to first solve the continuous ***linear programming*** (LP) relaxation, obtained by replacing the second constraint with the real variable $x_{i,j} \in [0,1]$, because LP problems can be solved in polynomial time. Then, the resulting fractional solution can be transformed to integer solutions through rounding such as randomized rounding [Raghavan 1987]. However, this approximation could inevitably lose the optimality.

In practice, the 0-1 ILP concurrent routing technique is often embedded into a larger overall global routing framework with a hierarchical, divide-and-conquer manner, such as solving a subproblem, in which the complexity of computing the optimal solution is manageable. Another approach to divide a routing region into subregions such that the routing problem can be handled subregion by subregion to reduce the problem size is BoxRouter [Cho 2006], which is based on box expansion to push the congestion outward progressively.

12.4.3 Steiner trees

The algorithms we have described so far are mainly for two-pin nets. If all nets are two-pin ones, we can apply a general-purpose routing algorithm to handle the problem, such as maze, line-search, and A*-search routing described in Section 12.3.

For three or more multi-pin nets, one naive approach is to *decompose* each net into a set of two-pin connections, and then route the connections one-by-one. One popular decomposition method is to find a ***minimum spanning tree***

(MST) for pins of each net, which is a minimum-length tree of edges connecting all the pins. The MST can efficiently be computed in polynomial time by the **Kruskal** [Kruskal 1956] or **Prim-Dijkstra** [Prim 1957] algorithms. However, the routing result of this approach would depend on the decomposition and often leads to only suboptimal solutions. Figure 12.12 depicts an example 4-pin net decomposed by a rectilinear MST, where each segment runs horizontally or vertically.

A better and more natural method to route multi-pin nets is to adopt the **Steiner-tree–**based approach. Specifically, a ***minimum rectilinear Steiner tree*** (MRST) is used for routing a multi-pin net with the minimum wirelength. Given m points in the plane, an MRST connects all points by rectilinear lines, possibly via some extra points (called **Steiner points**), to achieve a minimum-wirelength tree of rectilinear edges. Let P and S denote the sets of original points and Steiner points, respectively. Then, we have the following relationship between MRST and MST.

$$\text{MRST}(P) = \text{MST}(P \cup S) \tag{12.2}$$

Figure 12.13b shows an example of the MRST with two Steiner points s_1 and s_2 for the four pins p_1, p_2, p_3, and p_4 in Figure 12.13a.

There could be an infinite number of Steiner points that need to be considered for the MRST construction. Fortunately, Hanan proved that for a set P of pins, there exists an MRST of P with *all* Steiner points chosen from the grid points of the **Hanan grid**, which is obtained by constructing vertical and horizontal lines through every pin in P. This is known as **Hanan's theorem** [Hanan 1966]. Figure 12.13c shows the Hanan grid for the four pins in Figure 12.13a. Both the Steiner points s_1 and s_2 of MRST in Figure 12.13b are on the grid points of the Hanan grid.

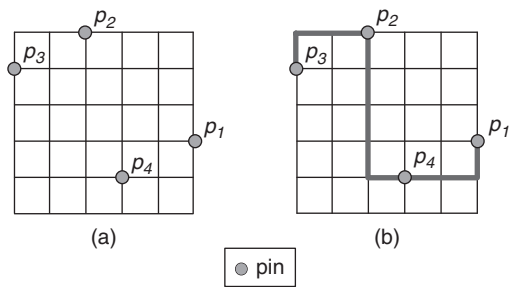


FIGURE 12.12

A 4-pin net decomposed by a minimum rectilinear spanning tree: (a) A net consisting of four pins: p_1, p_2, p_3, and p_4. (b) An MST of (a), which decomposes the net into three two-pin connections.

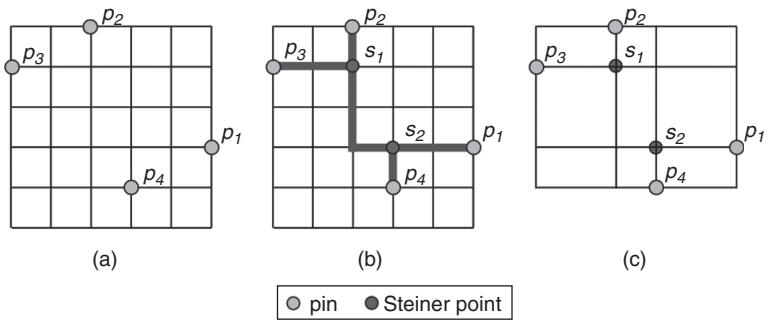


FIGURE 12.13

A minimum rectilinear Steiner tree (MRST) and its Hanan grid: (a) A net consisting of a set P of four pins: p_1, p_2, p_3, and p_4. (b) An MRST of (a) with the two Steiner points s_1 and s_2. (c) The Hanan grid of P. Note that all Steiner points s_1 and s_2 of MRST in (b) are chosen from the grid points on the Hanan grid.

The Hanan theorem greatly reduces the search space for the MRST construction from an infinite number of choices to only m^2-m candidates for the Steiner points, where $m=|P|$. However, the MRST construction is still an NP-hard problem [Garey 1977]. Therefore, many heuristics have been developed.

The relationship between MST and MRST can be stated by **Hwang's theorem** [Hwang 1976] as follows:

$$\frac{Wirelength(MST(P))}{Wirelength(MRST(P))} \leq \frac{3}{2} \tag{12.3}$$

Equation (12.3) gives a strong motivation for constructing an MRST by an MST-based approximation algorithm. Ho *et al.* constructed an MRST from an MST by maximizing monotonic (nondetour) edge (*e.g.*, L-shaped, Z-shaped) overlaps by dynamic programming [Ho 1990]. Kahng and Robins developed the **iterated 1-Steiner heuristic** [Kahng 1990] (see Algorithm 12.1). Starting with an MST, they iteratively select one Steiner point that can reduce the wirelength most and then add the Steiner point to the tree. The iterations continue until the wirelength cannot be further improved. Figures 12.14b–d illustrates the first, second, and third iterations after inserting Steiner points s_1, s_2, and s_3 into the initial MST in Figure 12.14a, respectively. Note that the iterated 1-Steiner heuristic may generate a "degenerate" Steiner point with the number of branches (degrees) $\leqq 2$, such as s_1 in Figure 12.14d. Therefore, we have to remove a degenerate Steiner point whenever it is created (see Figure 12.14e). Figure 12.14e shows the final MRST of Figure 12.14a.

On the basis of the **spanning graph** that contains an MRST in a sparse graph, [Zhou 2004] developed an efficient MRST algorithm with the worst-case time complexity of only $O(m \lg m)$ and solution quality close to that of the

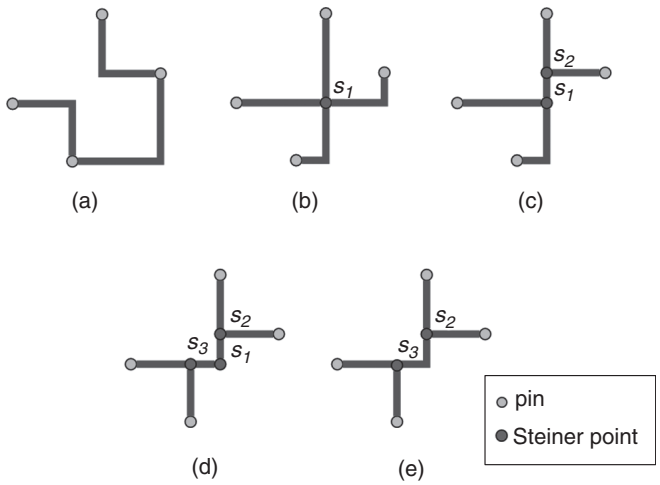


FIGURE 12.14

A step-by-step example of the iterated 1-Steiner heuristic for a 4-pin net: (a) The initial MST. (b) The MRST after the first iteration by inserting the Steiner point s_1. (c) The MRST after the second iteration by inserting the Steiner point s_2. (d) The MRST after the third iteration by inserting the Steiner point s_3. (e) The final MRST after removing the degenerate Steiner point s_1.

iterated 1-Steiner heuristic. [Chu 2004] developed the FLUTE package by use of precomputed lookup tables to efficiently and accurately estimate the wirelength for multi-pin nets. Lin *et al.* constructed a single-layer and a multi-layer obstacle-avoiding MRST to consider routing obstacles incurred from power networks, prerouted nets, IP blocks, and/or feature patterns for manufacturability/reliability improvements [Lin 2007, 2008]. Shi *et al*. constructed an obstacle-avoiding MRST based on a current-driven circuit model [Shi 2006].

Algorithm 12.1 Iterated 1-Steiner Algorithm

Input: P – a set of m pins.
Output: a Steiner tree on P.
1. $S \leftarrow \phi$;
   /*$H(P \cup S)$: set of Hanan points */
   /* $\Delta MST(A, B) = Wirelength(MST(A)) - Wirelength(MST(A \cup B))$ */
2. **while** ($Cand \leftarrow \{x \in H(P \cup S) \mid \Delta MST(P \cup S, \{x\}) > 0\} \neq \phi$) **do**
3. Find $x \in C$ *and* which maximizes $\Delta MST(P \cup S, \{x\})$;
4. $S \leftarrow S \cup \{x\}$;
5. Remove points in S which have degree ≤ 2 in $MST(P \cup S)$;
6. **end while**
7. **Output** $MST(P \cup S)$;