COL 215 Software Assignment -2

Himanshi Bhandari(2023CS10888)

Mannat(2023CS10138)

2 October 2024

Problem statement

The primary goal is to arrange the gates in a way that minimizes the wire length for the given connections. Gates must be placed without overlapping each other and also rotation of gates is not allowed.

Design decisions

This report outlines the design and implementation of a greedy placement algorithm for positioning gates in a circuit layout. The algorithm aims to minimize total wire length while avoiding gate overlaps. The solution prioritizes gates with the maximum connections and attempts to find positions that minimize wire length with previously placed gates.

Input Processing

- Read the input file, creating Gate objects with their dimensions and pins
- Build the connection network between gates

Gate Sorting

- Sort gates by the number of connections in descending order
- This prioritizes placing more connected gates first, potentially leading to better overall wire length

Initial Placement

- Place the most connected gate at the origin (0, 0)
- This serves as an anchor point for the rest of the placement

Iterative Placement

For each remaining gate:

a) Select up to 4 candidate gates (most connected among the unplaced) b) For each candidate:

- Try placing it in 4 directions around each already placed gate
- Check for overlaps with existing gates
- If no overlap, calculate the resulting wire length
- Keep track of the placement with the minimum wire length .Place the gate at the position that resulted in the minimum wire length

Wire length calculation optimization:

- For n * connections > 250 * 3000: Minimize the wire length of the newly placed gate with previously placed gates
- Otherwise: Minimize the total wire length
- This optimization reduces execution time for larger circuits

Wire Length Calculation

The wire length calculation process involves:

1. Pin-based Iteration:
    - Iterate through all placed gates and their pins
    - Calculate wire length only for unprocessed pins to avoid redundancy
2. Connected Coordinates Collection:
    - For each pin, collect the absolute coordinates of itself and all connected pins
    - Skip already processed pins to avoid redundant calculations
3. Bounding Box Calculation:
    - Determine the bounding box encompassing all connected pins
    - Find minimum and maximum x and y coordinates among all connected points
4. Semiperimeter Addition:
    - Estimate wire length for each pin's connections as the semiperimeter of its bounding box
    - Calculate as (max_x - min_x) + (max_y - min_y)
    - Add this semiperimeter to the total wire length

This approach approximates the minimal wiring needed to connect all pins in the group.

Output Generation

- Calculate the final bounding box of the entire layout
- Shift all gate positions to ensure the layout starts at (0, 0)
- Write the results to the output file

Design Rationale

1. Greedy Approach: The algorithm uses a greedy strategy to balance solution quality and computational efficiency.
2. Centralized Placement: The gate with maximum connections is placed in the center (0,0) initially. This strategy aims to reduce overall wire length, as the most connected gate will have minimal average distance to other gates.
3. Connection-based Sorting: Gates are sorted based on their number of connections. This prioritizes the placement of highly connected gates, which have a more significant impact on the overall wire length.
4. Candidate Selection: Up to 4 candidate gates are considered for each placement, balancing between exploring multiple options and maintaining algorithm efficiency.
5. Wire Length Optimization: The algorithm adapts its wire length calculation based on the problem size, optimizing for execution time in larger circuits while maintaining accuracy for smaller ones.

This greedy placement algorithm provides an efficient solution to the gate placement problem. By prioritizing highly connected gates and using a strategic placement approach, it aims to minimize total wire length while avoiding gate overlaps. The algorithm's adaptability to different problem sizes and its balance between solution quality and computational efficiency make it suitable for various circuit layout scenarios.

Time Complexity Analysis

1. read_input function

- Time Complexity: O(n + m), where n is the number of lines in the input file and m is the total number of pins across all gates.
- Explanation:
  - Reading each line: O(n)
  - For each 'pins' line, we iterate over pin coordinates: O(m)
  - The 'wire' lines are processed in constant time each.

2. calculate_wire_length function

- Time Complexity: O(k), where k is the total number of connections, p is the total number of pins across all gates and c is the average number of connections per pin.
- Explanation:
  - Outer loop iterates over all gates and their pins: O(p)
  - For each pin, we process its connections: O(c)
  - Finding min and max coordinates: O(c)
  - The set operations (adding to processed_pins) are O(1) on average.
  - So time complexity is O(p*c) , p*c=k ,so O(k)

3. calculate_wire_length_gates function

- Time Complexity: O(f), where f is the total number of connections of the gate, p is the total number of pins across that gate and c is the average number of connections per pin.
- Explanation:
  - Outer loop iterates over all pins: O(p)
  - For each pin, we process its connections: O(c)
  - Finding min and max coordinates: O(c)
  - The set operations (adding to processed_pins) are O(1) on average.
  - So time complexity is O(p*c) , p*c=f ,so O(f)

4. calculate_bounding_box function

- Time Complexity: O(n), where n is the number of gates.
- Explanation:
  - Single pass over all gates to find min and max coordinates.

5. is_overlapping function

- Time Complexity: O(g), where g is the number of placed gates.
- Explanation:
  - Checks overlap with each placed gate.

6. greedy_placement function

- Time Complexity: O(g*k), where g is the number of gates, k is the total number of connections, p is the average number of pins per gate, and c is the average number of connections per pin.
- Explanation:
  - Sorting gates: O(g log g)
  - Main loop runs n times (for each gate):
    - For each gate, we try 4 positions around up to placed gates: O()
    - For each position, we call is_overlapping: O(g)
    - If not overlapping, we call calculate_wire_length: O(p * c)
  - Total: O(g * (g * (g + p * c))) = O(g^3 + g^2 * p * c)
  - The g^3 term is typically dominated by g^2 * p * c for realistic circuits, so we can simplify to O(g^2 * p * c)
  - g*p*c=total number of connections i.e. k. Therefore, time complexity is O(g*k).

7. main function

- Time Complexity: O(g^2 * p * c + n + m), where g is the number of gates, p is the average number of pins per gate, c is the average number of connections per pin, n is the number of lines in the input file, and m is the total number of pins.
- Explanation:
  - read_input: O(n + m)
  - greedy_placement: O(g^2 * p * c) i.e O(g*k)
  - final calculate_wire_length: O(p * c)
  - calculate_bounding_box: O(g)
  - Writing output: O(g)

Overall Time Complexity

The overall time complexity of the program is dominated by the greedy_placement function: O(g^2 * p * c) i.e O(g*k) where g is total number of gates and k is total number of connections.

Public Test Cases:

1)test case 1

input	output
g1 3 3 pins g1 0 1 0 2 3 1 3 2 g2 4 1 pins g2 0 1 4 0 g3 4 1 pins g3 4 1 4 0 0 1 0 0 g4 4 3 pins g4 0 3 4 1 4 2 0 1 g5 4 4 pins g5 0 1 0 2 0 3 4 1 4 2 4 3 g6 3 3 pins g6 0 1 0 2 3 1 3 2 g7 3 4	Total Wire Length: 32 bounding_box 11 10 g3 4 4 g4 0 4 g6 4 5 g2 4 8 g7 8 4 g1 0 7 g8 0 0 g5 4 0

pins g7 0 0 0 2 0 3 3 1 g8 4 4 pins g8 0 1 0 2 0 3 4 1 wire g3.p2 g7.p1 wire g7.p2 g3.p1 wire g5.p5 g3.p1 wire g5.p1 g8.p4 wire g2.p2 g6.p4 wire g2.p1 g6.p1 wire g6.p2 g4.p1 wire g1.p4 g6.p1 wire g4.p3 g8.p1 wire g1.p1 g8.p3 wire g3.p3 g4.p4	

2)test case 2

input	output
g1 3 4 pins g1 0 3 0 1 3 2 3 1 g2 4 4 pins g2 0 2 4 3 4 1 g3 2 4 pins g3 0 1 0 2 0 3 2 2 g4 2 3 pins g4 0 1 2 2 2 1 wire g1.p1 g3.p2 wire g1.p3 g4.p2 wire g1.p4 g2.p1 wire g4.p1 g3.p3 wire g4.p3 g3.p1 wire g2.p2 g3.p4 wire g2.p3 g1.p2	Total Wire Length: 30 bounding_box 11 4 g1 4 0 g2 0 0 g3 7 0 g4 9 0

3)test Cases

input	output
g1 3 3 pins g1 0 1 3 2 g2 4 4 pins g2 0 0 4 3 g3 2 4 pins g3 0 1 2 3 g4 3 3 pins g4 0 1 3 2 g5 4 3 pins g5 0 1 4 2 0 2 g6 2 3 pins g6 0 0 2 2 g7 3 5 pins g7 0 1 3 4 g8 3 3 pins g8 0 1 3 2 g9 4 4 pins g9 0 0 4 3 4 1 g10 2 3	Total Wire Length: 134 bounding_box 28 19 g9 16 8 g3 16 4 g6 14 8 g7 20 8 g2 16 12 g10 14 5 g8 13 12 g12 23 8 g13 12 8 g14 10 5 g5 20 5 g4 9 8 g16 20 13 g17 22 13 g15 6 8 g19 16 0 g20 26 8

```
pins g10 0 0 2 2
g11 3 4
pins g11 0 1 3 3
g12 3 5
pins g12 0 2 3 4 3 1
g13 2 4
pins g13 0 1 2 3 0 2
g14 4 3
pins g14 0 0 4 2
g15 3 3
pins g15 0 1 3 2 3 3
g16 2 5
pins g16 0 1 2 4
g17 3 4
pins g17 0 2 3 3 3 4
g18 3 3
pins g18 0 0 3 1 3 2
g19 4 4
pins g19 0 0 4 3
g20 2 3
pins g20 0 0 2 2
g21 3 4
pins g21 0 2 3 3 3 0
g22 3 5
pins g22 0 1 3 4
g23 4 3
pins g23 0 0 4 2
g24 2 5
pins g24 0 1 2 4 2 3
g25 3 3
pins g25 0 1 3 2
wire g1.p1 g4.p2
wire g2.p1 g8.p1
wire g3.p2 g10.p1
wire g4.p1 g15.p2
wire g5.p2 g20.p1
wire g6.p2 g9.p1
wire g7.p1 g12.p3
wire g8.p2 g13.p2
wire g9.p3 g16.p1
wire g10.p2 g23.p1
wire g11.p1 g18.p2
wire g12.p1 g19.p2
wire g13.p3 g14.p1
wire g14.p2 g24.p1
wire g15.p3 g21.p2
wire g16.p2 g17.p3
wire g17.p1 g25.p2
wire g18.p3 g22.p1
wire g19.p1 g3.p1
wire g20.p2 g7.p2
wire g21.p3 g6.p2
wire g22.p2 g11.p2
wire g23.p2 g5.p3
wire g24.p3 g9.p2
wire g25.p2 g2.p2
```

```
g21 9 11
g22 6 11
g23 20 2
g11 3 11
g18 0 11
g25 16 16
g1 13 15
g24 14 0
```

4)test case 4

input	output
g1 3 3 pins g1 3 2 3 0 g2 2 4 pins g2 0 1 0 3 2 4 2 2 g3 3 4 pins g3 3 4 3 3 0 2 g4 4 2 pins g4 0 1 4 1 4 0 g5 10 4 pins g5 0 4 0 1 10 1 10 4 wire g1.p1 g2.p1 wire g1.p1 g2.p2 wire g1.p1 g3.p3 wire g2.p3 g4.p1 wire g2.p3 g3.p2 wire g3.p1 g4.p2 wire g3.p1 g5.p1 wire g2.p3 g5.p3 wire g4.p3 g5.p4	Total Wire Length: 40 bounding_box 15 8 g2 10 0 g1 7 0 g4 10 4 g5 0 4 g3 12 0

Test cases

1) This involves case **only one gate and no connection** present

input	output
g1 3 2 pins g1 0 0	Total Wire Length: 0 bounding_box 3 2 g1 0 0

2) This involves **two gates with one connection** between them

input	output
g1 3 2 pins g1 0 0 2 1 g2 4 3 pins g2 0 0 3 2 wire g1.p1 g2.p1	Total Wire Length: 2 bounding_box 4 5 g1 0 0 g2 0 2

3)Taking **gates with maximum dimensions i.e where all gates have side of 100 by 100**.

input	output
g1 100 100 pins g1 0 0 100 0 g2 100 100 pins g2 0 0 100 0	Total Wire Length: 0 bounding_box 300 100 g2 100 0 g1 0 0

g3 100 100	g3 200 0
pins g3 0 0 100 0	
wire g1.p2 g2.p1	
wire g2.p2 g3.p1	

4) It is the case involving dense pin connections where **number of connections is much more than the number of gates**.

input	output
g1 5 5	Total Wire Length: 95
pins g1 0 0 5 0 5 5	bounding_box 15 10
g2 5 5	g1 10 0
pins g2 0 0 5 0 5 5	g3 5 0
g3 5 5	g4 5 5
pins g3 0 0 5 0 5 5	g5 0 0
g4 5 5	g2 0 5
pins g4 0 0 5 0 5 5	
g5 5 5	
pins g5 0 0 5 0 5 5	
wire g1.p1 g2.p2	
wire g1.p2 g3.p1	
wire g1.p3 g4.p2	
wire g1.p1 g5.p1	
wire g2.p1 g3.p2	
wire g2.p2 g4.p3	
wire g2.p3 g5.p2	
wire g3.p1 g4.p1	
wire g3.p2 g5.p3	
wire g3.p3 g1.p2	
wire g4.p1 g5.p3	
wire g4.p2 g1.p1	
wire g4.p3 g2.p2	
wire g5.p1 g3.p1	
wire g5.p2 g4.p2	
wire g5.p3 g1.p3	

5)It is the case where **one central gate is connected to multiple perpheral gates.**

input	output
g1 5 5	Total Wire Length: 46
pins g1 0 0 0 2 0 4 2 0 2 5 4 0 4 2 4 4 5 0 5 2 5 4	bounding_box 14 12
g2 3 3	g1 3 3
pins g2 1 0 1 3	g2 3 0
g3 3 3	g3 8 3
pins g3 1 0 1 3	g6 3 8
g4 3 3	g4 8 6
pins g4 1 0 1 3	g9 8 9
g5 3 3	g5 0 3
pins g5 1 0 1 3	g7 6 0
g6 3 3	g8 11 6
pins g6 1 0 1 3	
g7 3 3	
pins g7 1 0 1 3	
g8 3 3	

pins g8 1 0 1 3	
g9 3 3	
pins g9 1 0 1 3	
wire g1.p1 g2.p1	
wire g1.p2 g3.p1	
wire g1.p3 g4.p1	
wire g1.p4 g5.p1	
wire g1.p5 g6.p1	
wire g1.p6 g7.p1	
wire g1.p7 g8.p1	
wire g1.p8 g9.p1	
wire g1.p9 g2.p2	
wire g1.p10 g3.p2	
wire g1.p11 g4.p2	

6)It involves cases when we have **large number of gates but less connections i.e there might be some isolated gates.**

input	output
g1 2 2	Total Wire Length: 2
pins g1 0 0 2 2	bounding_box 6 14
g2 2 2	g1 2 4
pins g2 0 0 2 2	g2 2 2
g3 2 2	g3 2 6
pins g3 0 0 2 2	g4 0 4
g4 2 2	g5 2 8
pins g4 0 0 2 2	g6 0 6
g5 2 2	g7 2 10
pins g5 0 0 2 2	g8 0 8
g6 2 2	g9 2 12
pins g6 0 0 2 2	g10 0 10
g7 2 2	g11 4 4
pins g7 0 0 2 2	g12 4 2
g8 2 2	g13 2 0
pins g8 0 0 2 2	g14 0 2
g9 2 2	g15 4 6
pins g9 0 0 2 2	
g10 2 2	
pins g10 0 0 2 2	
g11 2 2	
pins g11 0 0 2 2	
g12 2 2	
pins g12 0 0 2 2	
g13 2 2	
pins g13 0 0 2 2	
g14 2 2	
pins g14 0 0 2 2	
g15 2 2	
pins g15 0 0 2 2	
wire g1.p1 g2.p2	
wire g3.p1 g4.p2	
wire g5.p1 g6.p2	
wire g7.p1 g8.p2	
wire g9.p1 g10.p2	

7) **Large Test Cases**:

1)this consists of **1000 gates along with 3300 connnections,** the output we got was

Total Wire Length: 1168979

bounding_box 2104 2208

2)this consisits of **500 gates along with 2200 connections**, the output we got was

Total Wire Length: 954880

bounding_box 1500 1453.

3) this consists of **750 gates along with 4800 connections,** the output we got was

Total Wire Length: 2454092

bounding_box 1790 1808

4)this consists of **1000 gates with 11000 connections** ,the output we got was

Total Wire Length: 7356205

bounding_box 2126 2138

Conclusion

In summary, the program implements a straightforward yet effective greedy algorithm for gate placement on a grid, focusing on minimizing wire length and avoiding gate overlap. The algorithm calculates and outputs the total wire length, bounding box size, and final gate positions. For moderately sized circuits, the program provides highly optimized wire lengths within minimal execution time, making it ideal for real-world applications where wire length and layout efficiency are crucial.

For larger circuits, we use a little different algorithm so that time for wire length calculations are further optimized depending on the complexity of the circuit, particularly the number of gates and connections. While the wire lengths are still optimized for larger circuits, the algorithm designed for moderately sized circuits tends to yield better outcomes overall.

The program efficiently handles typical input sizes, and the final layout is outputted in a structured format that includes the total wire length and bounding box dimensions. This makes the solution suitable for practical circuit layout problems where minimizing wire length directly impacts performance and cost-effectiveness.