1. Problem Overview

In this problem, we aim to simulate a simplified version of VLSI floorplanning within a chip circuit, a design challenge often found in the electrical chip manufacturing industry. Your objective is to determine the most optimal placement of 6 common chip components on a 2D chip grid, **minimizing** both **wire length** (that interconnects certain components) and the overall required **chip area**, while **avoiding overlapping** chip components. Your task is to design a suitable chip layout optimizer using what you have been taught in your Genetic Algorithm class.

2. Grid and Components

Suppose the chip grid is of dimension **25 x 25** unit square (you are encouraged to take it as an input parameter to experiment later on how your algorithm behaves upon changing the grid size).

Say, the chip contains the following 6 functional blocks, each with a fixed size listed below:

Table 1

Block Name	**Width (unit)**	**Height (unit)**
ALU	5	5
Cache	7	4
Control Unit	4	4
Register File	6	6
Decoder	5	3
Floating Unit	5	5

3. Required Interconnections

One of the three goals of your layout design is to **minimize** the wiring distance between the following 6 connected component pairs:

Register File → ALU

Control Unit → ALU
ALU → Cache
Register File → Floating Unit
Cache → Decoder
Decoder → Floating Unit

You can assume all wiring connections are bidirectional, incur no additional space, and wires run **center-to-center** between components. Generally, the more compact the chip component placements are, the shorter the total wiring length becomes, and vice versa.

4. Fitness Objectives

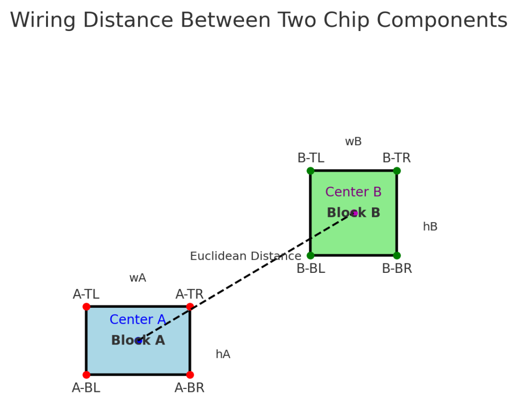

Figure 1

1. **Minimizing the total wiring distance:** Figure 1 above explains how the center-to-center wiring length can be calculated when both the blocks' height, width (wA, hA, wB, hB), and any single pair of corner points are given [suppose, bottom left of A (A-BL) and bottom left of B (B-BL) are given]. This is essentially the Euclidean Distance between two geometric points (two centers) in a 2D space. The **less total** wiring distance needed, the **more desirable** the layout is.

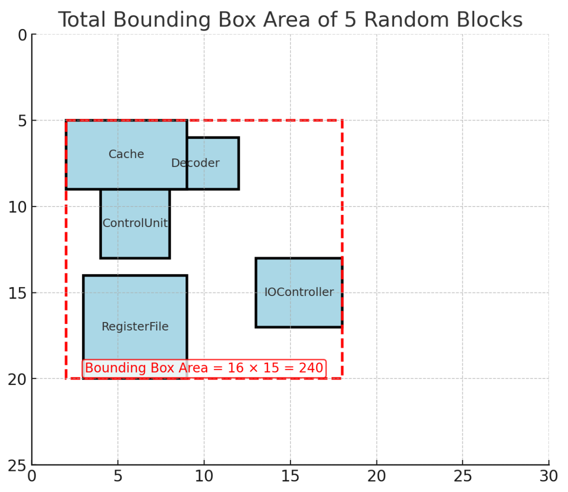

Figure 2

2. **Minimizing the total bounding area:** Figure 2 above hints at how the total bounding area can be calculated for a random placement of 5 component blocks. The red dashed rectangle is the bounding box- aka the smallest rectangle that contains all of the blocks inside. Given the bottom left coordinates (x,y) for all block components, you can easily find out the minimum and maximum of all x and y values, which will help you to calculate the area of such a bounding box [(x_max - x_min) * (y_max - y_min)]. The **smaller** the bounding box area is, the **better** the layout is.

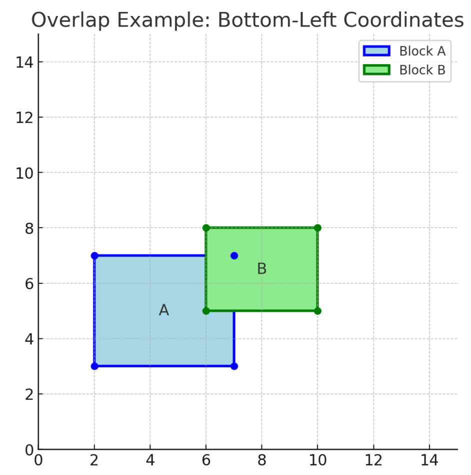

Figure 3

3. **Minimizing total component overlaps:** In Figure 2, there exists an overlap between the two components (cache and decoder). A single block can, in fact, be overlapped with multiple other blocks in a random placement. **Overlaps are not desirable.** Hence, a heavy penalty will be applied for each overlapping block pair. For any two block pairs A and B (Figure 3), one possible way to figure out the overlap is by checking the following condition:

```
overlap = not (
A_right <= B_left or
A_left >= B_right or
A_bottom >= B_top or
A_top <= B_bottom
)
```

where right, left, and top, bottom can be derived from the bottom-left x,y coordinate values of both blocks and the width and height of the corresponding blocks, respectively. The idea is that no overlap exists if one rectangle is completely to the left, right, above, **or** below the other.

Task 1 Instructions

Start with a random initial population of 6 members, where each chromosome represents a candidate layout. Suppose the layout is specified by all the **bottom-left coordinates** of the 6 components (outlined in section 2 in that exact order). Check section 7 for a sample input analysis.

1. **Chromosome Representation / Encoding:**
    a. Devise a suitable encoding scheme to represent all the block components specified in the question. Note that the placement coordinate values will be in the range of [0, 25]. Be as concise as possible in your encoding; that is, try not to keep redundant information in your encoded version.
    b. Then generate an initial population of 6 chromosomes to start the GA loop, where each chromosome represents a placement strategy.
2. **Fitness Function Implementation:**
    a. Based on the 3 fitness objectives outlined above, design a suitable fitness function that focuses on penalizing bad placement combinations. Implementation hints are outlined in detail in section 4. Follow a prioritized order of these objectives as mentioned below, and reflect that in your fitness calculation.
        i. Overlap counts should be considered the least desirable, hence penalized way more than the other two
        ii. The total wiring distance and the total bounding area should be considered next
    b. Calculate the fitness of the chromosome pool using the hints specified above. A weighted sum can be a good fitness function to start with. Check the sample input output section for some ideas to start with.
3. **Parent Selection:**
    a. Choose two parents based on random selection at each generation.

4. **Cross-over:**
    a. Perform single-point crossover to create 2 new offspring from each pair of selected parents.
    b. To achieve this, pick a random split point from the chromosome representation of the parents and swap the complementary pieces to generate offspring.
5. **Mutation:**
    a. Mutation ensures genetic diversity and prevents the algorithm from getting stuck in local optima.
    b. Implement the following mutation strategy to introduce random changes: pick any arbitrary component block from the chromosome representation and introduce random coordinate (i.e., the value of x and y) changes to that single particular block with a low probability (5-10% mutation rate).
6. **New Generation Creation:**
    a. Remember, the population size should be the same for all generations. Apply elitism to select new individuals (allow 1 or 2 elites to be carried forward to the next generation). Select the remaining candidates from the created offspring for the next generation. Random Uniform selection, Tournament selection, and Roulette Wheel selection are some of the other popular selection techniques.
7. **GA Loop:**
    a. Run genetic algorithms on such a population until the best fitness (a plateau) is achieved or the maximum number of 15 iterations is reached. Keep this generation count as a variable input for further experiments.
8. **Required Output:**
    a. Output the best possible placement strategy found once the algorithm halts. Deliverables include the best total fitness value, the corresponding total wiring length, the total bounding box area, and the total overlap counts. Also, decode the best chromosome representation to reflect the optimal placement of bottom-left coordinates.

Task 2

Take a look at **Step 4 of Task 1**. Can you incorporate the **Two-point crossover mechanism** into it?

- To implement this task, randomly select two parents from your initial population. Then perform a two-point crossover to generate two children. The two points have to be chosen randomly, but it has to be ensured that the second point always comes after the first point.

Sample Input and Output

A sample initial population set of 6 chromosomes, where each line below represents the bottom-left position coordinate of 6 chip component blocks:

P1 → (9,3), (12, 15), (13, 16), (1,13), (4,15), (9, 6)
P2 → (8, 0), (7,12), (4,11), (1,13), (14,10), (9,11)
P3 → (6, 5), (12, 9), (9, 7), (8, 6), (2, 7), (3, 1)
P4 → (3,11), (11, 12), (14, 11), (6, 10), (3,11), (3,0)
P5 → (10, 12) (8, 16), (10, 4), (13, 6), (6, 0), (3, 7)
P6 → (0, 2), (0, 0), (14, 12), (4, 5), (12, 4), (3, 10)

For **P1,**

Pairwise block overlap count = 3

Total wiring distance (center-to-center) of the specified connected pairs = 12.91 + 12.98 + 12.18 + 10.61 + 9.01 + 9.43 = 67.1

Total bounding box area = (x_max - x_min) * (y_max - y_min) = (19-1) * (20-3) = 306

Total fitness value = - (alpha * overlap penalty + beta * wiring length penalty + gamma * bounding area penalty)
Considering, alpha = 1000, beta = 2, gamma = 1:

Fitness for generation 1 chromosome 1 = - (1000 * 3) - (67.1 * 2) - 306 = -3440.23