These are the results of algorithms to solve the one-dimensional circle packing problem. The program is implemented in C++ on VS2019.
(2nd part of the homework is late by ~17 Hours.)
Unimplemented & Wrong Algorithms -> Ant Colony, Particle Swarm and Branch and Bound.

Although Brute Force method always gives the best answer the time it takes grows exponentially as the time complexity is O(n!) and after 9 circles it times out.

Instance	Best	Average	Worst	Average Time
Instance1	198.99	198.99	198.99	6ms
Instance2	N/A	N/A	N/A	N/A
Instance3	N/A	N/A	N/A	N/A
Instance4	N/A	N/A	N/A	N/A
Instance5	N/A	N/A	N/A	N/A

Brute Force

Branch and Bound was implemented incorrectly and it basically works like a brute-force method.

Instance	Best	Average	Worst	Average Time
Instance1	198.99	198.99	198.99	6ms
Instance2	N/A	N/A	N/A	N/A
Instance3	N/A	N/A	N/A	N/A
Instance4	N/A	N/A	N/A	N/A
Instance5	N/A	N/A	N/A	N/A

Branch and Bound

Greedy method always picks the circle that will make the line shortest. Therefore, it picks the smallest circles first which is basically sorting the circles. Which is always the wrong solution but it is the fastest algorithm to find a solution even if it's a incorrect one at that.

Instance	Best	Average	Worst	Average Time
Instance1	243.049	243.049	243.049	0ms
Instance2	1153.3	1153.3	1153.3	0ms
Instance3	5199.61	5199.61	5199.61	6ms
Instance4	11038.5	11038.5	11038.5	48.8ms
Instance5	21244.9	21244.9	21244.9	357.6ms

Greedy

Iterated Local Search didn't always find the optimal solution but it did found better solutions than greedy and simulated annealing but the average run time is worse than the other algorithms. The stopping condition was 25 iterations without a new best solution.

Instance	Best	Average	Worst	Average Time
Instance1	205.618	205.618	205.618	1.2ms
Instance2	1115.47	1115.47	1115.47	3.2ms
Instance3	4596.53	4596.53	4596.53	96.6ms
Instance4	9575.39	9575.39	9575.39	595ms
Instance5	18303.5	18303.5	18303.5	7701.4ms

Iterated Local Search

Simulated Annealing algorithm is much faster than VNS and Iterated Local Search but with some exceptions it always found a worse solution the other metaheuristics. As there wasn't much of diversification in my algorithm for example heat up again after cool down. The neighborhood was generated with 2 swaps.

Instance	Best	Average	Worst	Average Time
Instance1	204.244	204.244	204.244	0ms
Instance2	1090.32	1090.32	1090.32	1ms
Instance3	4860.93	4860.93	4860.93	17ms
Instance4	10631	10746.4	10775.2	81.2ms
Instance5	20712.5	20737.7	20817.9	476.8ms

SIMULATED ANNEALING

Parameters: TEMP_MIN: 0.000001f float TEMP_DEC : 0.9f NEIGHBORHOOD_SWAP_COUNT : 2

VNDS is the most successful algorithm(Part 1 of the homework) in terms of finding an optimal solution as the circle pool grows except for Instance1 but the Average Time is the worst among them as it timed out for Instance5, even the Instance4 took almost 10 times more than what Simulated Annealing did for Instance5.

Instance	Best	Average	Worst	Average Time
Instance1	225.041	225.041	225.041	8.2ms
Instance2	1089.41	1096.59	1101.38	27.8ms
Instance3	4580.03	4611.58	4650.52	607.6ms
Instance4	9512.83	9539.73	9578.4	4142.8ms
Instance5	N/A	N/A	N/A	N/A

VNDS

(Variable Neighborhood Decomposition Search)

Parameters: MAX_NEIGHBORHOOD: 5 MIN_NEIGHBORHOOD : 1 DELTA_NEIGHBORHOOD: 1;

Genetic algorithm proves to be the fastest algorithm to find a solution but not the best one always. It relies on good initial solutions, crossovers and mutations. The initial population is generated randomly and the size is 12. The mating pool is generated from 8 best chromosomes out of 12 of the current pool. The worst 4 solutions are replaced by offsprings generated from best 8. Stopping condition is 10 iterations without a better global solution.

Instance	Best	Average	Worst	Average Time
Instance1	199.757	199.757	199.757	6ms
Instance2	1099.92	1099.92	1099.02	8.8ms
Instance3	4695.03	4695.03	4695.03	27.4ms
Instance4	10037.7	10037.7	10037.7	50.4ms
Instance5	19455.4	19455.4	19455.4	108.6ms

Genetic Algorithm

While Tabu Search most of the time gives the best solutions out of all the metaheuristics, time complexity is exponential. The long-term memory size is 20 and the neighborhood is generated by 2 swaps in the line. Stopping condition is 15 tries without finding a globally better solution.

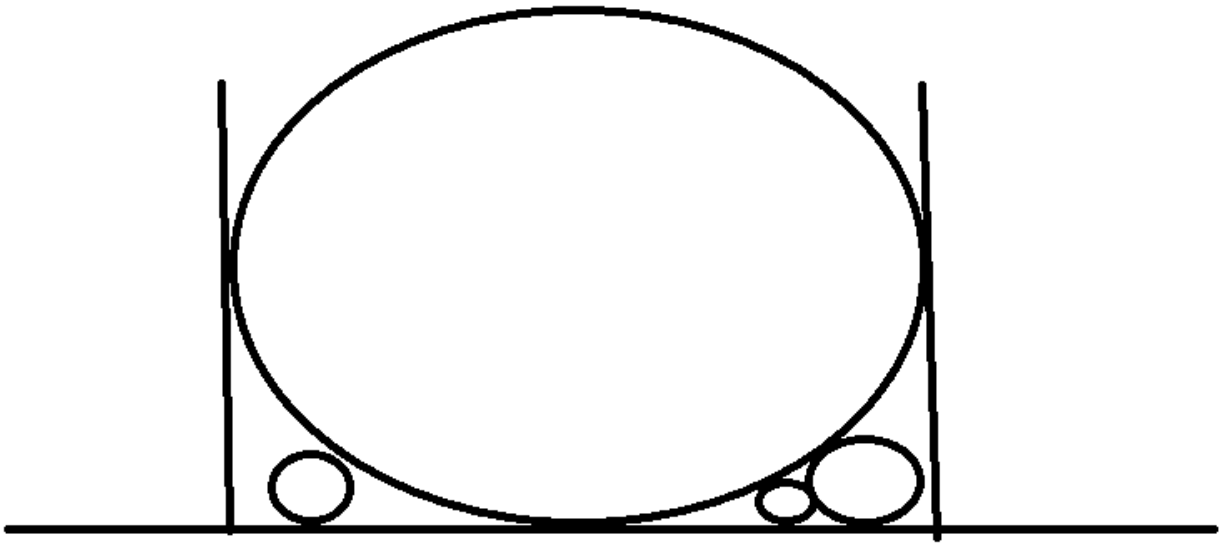
Instance	Best	Average	Worst	Average Time
Instance1	198.99	198.99	198.99	1ms
Instance2	1088.69	1088.69	1088.69	3ms
Instance3	4560.31	4580.08	4587.17	103.2ms
Instance4	9574.58	9621.97	9660.17	615.6ms
Instance5	18425.4	18489.8	18577.3	4710.4ms

Tabu Search

N/A means algorithm timed out. Algorithms were executed 10 times to find the avg time and avg, best, worst solutions.
Computer Specs: Ryzen 5 2700 3.6GHz, 16GB Ram GTX 960

Instance Sizes:
Instance1: 5
Instance2: 10
Instance3: 50
Instance4: 100
Instance5: 200

There's a problem with my calculation of the length of the line where I am always assuming the first and last circles are always the ones that on the edge which may not be case.



cases like these give false results.