CPEN513 A2 Placement

Summary

In this assignment, simulated annealing algorithm is used on chip placement. Initial temperature is generated from a set of random swaps. Half perimeter is used to approximate the cost. Two exit criterion are used. The first one is the temperature threshold. Annealing stops when the temperature is below the threshold. The second one is the deviation of previous costs. The deviation of n recent costs is being monitored. We exit the loop when the cost does not change much over the time. We reduce temperature by multiply it by a constant called cooling rate.

([GitHub repository])

Algorithm

Annealing algorithm is used for chip placement. To reduce time complexity, cost is calculated incrementally.

Cost

Half perimeter is used as cost here.

`half_perimeter = (max_row - min_row) + (max_col - min_col)`

Initial temperature

Initial temperature is generated from a set of random swaps. A list cost is generated by the cost of those random movements. We then multiply the standard deviation of those costs by a constant `K`

```
function initial_temp(K, num_moves):
    costs = [0] * num_moves # initial the cost list
    for i in range(num_moves):
        randoms swap sites and calculate costs
        costs[i] = cost

    return K * standard_deviation(costs)
```

Exit criterion

There are two exit criterion.

The first one is the temperature threshold, which is a constant value. The second one is the deviation of previous costs. Annealing stops when either of the criteria is met. We only keep track of n most recent cost and calculate the deviation of them. If we found that the cost does not change much or not change at all, we can just exit the annealing directly.

Here is the pseudocode for part of the implementation of the second criteria.

```
def record_n_recent_costs(n, costs, cost):
    """only keep track of n recent cost and return stdev of the n costs
    n: number of recent costs
    costs_q: deque that stores cost.
    cost: the cost you want to add
    """
    while len(costs) >= n - 1:
            costs.pop(0)
    costs.append(cost)
```

Reduce temperate

We reduce temperature by multiply it by a constant called cooling rate.

Annealing

Here is the pseudocode for annealing algorithm:

```
function annealing(cooling_rate, threshold, num_iterations,n, n_recent_costs,
stdev_threshold):
    T = initial_temp()
    cost = calculate_total_cost() # get the initial cost of the circuit
    while T > threshold or stdev(n_recent_costs) > stdev_threshold:

        cost_before_swap = 0
        cost_after_swap = 0

        pick site1, site2
        cost_before_swap += site1.update_net_cost + site2.update_net_cost # we
only update cost related to the two sites

        swap(site1, site2)
        cost_after_swap += site1.update_net_cost + site2.update_net_cost

        delta_c = cost_after_swap - cost_before_swap

        r = random(0,1) # generate a random number between 0 and 1
        if r < exp( -delta_c / T):
            # take exploratory moves
            cost += delta_cost
            record_n_recent_costs(n, n_recent_costs, cost)
        else:
            # undo moves by swap two sites back
            swap(site2, site1)
```

Test

A `pytest` test file for testing the half perimeter function

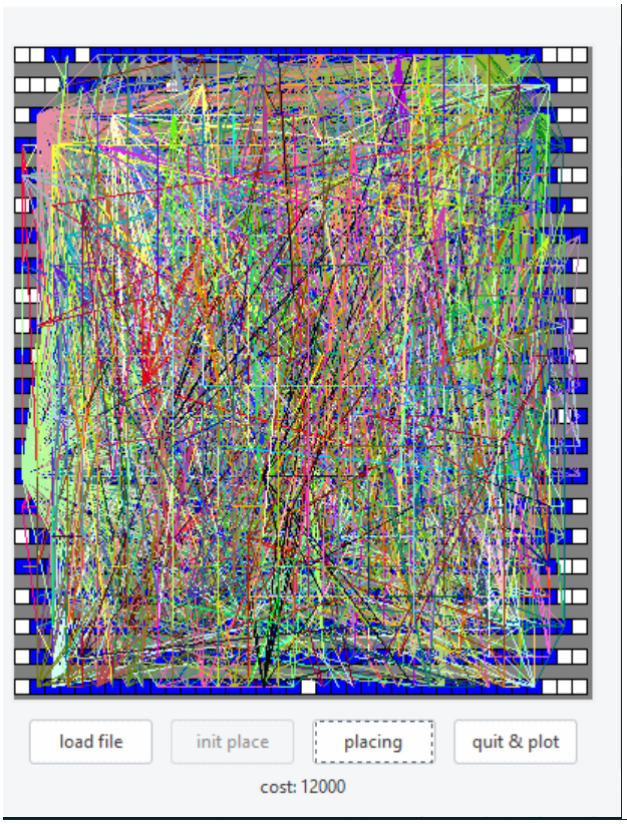
```
# lists of positions (row, col)
list0 = [(1,2), (0,0)] # half_p = (1-0) + (2-0) = 3
list1 = [(1,2), (2,3), (4,5), (6,7), (8,9), (10, 11)] # half_p = (10-1) + (11-2)
= 18
list2 = [] # Empty list # return 0
tests = [(list0, 3), (list1, 18), (list2, 0)]
@pytest.mark.parametrize("test_input, expected", tests)

def test_route_with_shuffle(test_input, expected):
  assert calc_half_perimeter(test_input) == expected
```

All tests are passed.

```
3 passed in 0.87s
```

GUI

`FuncAnimation` from `matplot` is easy to implement when draw direct data grid or chart. This time we need to draw nets connections. Drawing random shapes or lines are easier on GUI tools like `tkinker`. So both `matplot` and `tkinker` are used in this program. `tkinker` shows the real time chip placement and `matplot` is used to draw the line chart of cost and temperature after annealing.


load file
init place
placing
quit & plot
cost: 12000

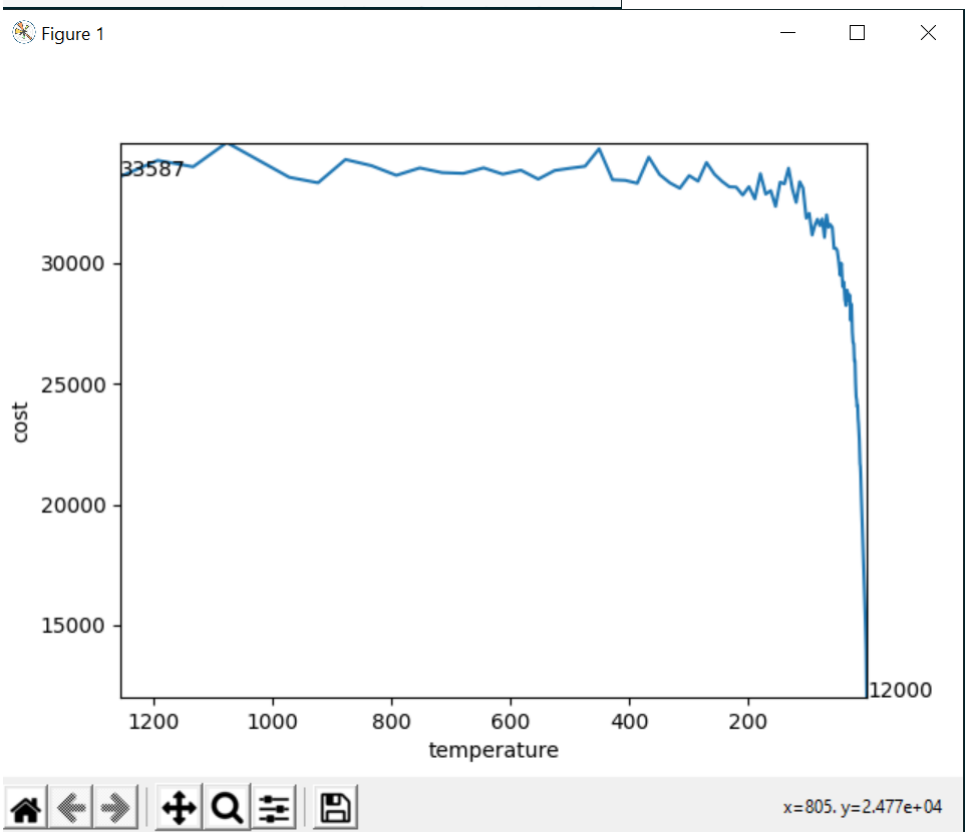


Figure 1
33587
30000
25000
20000
15000
cost
12000
1200
1000
800
600
400
200
temperature
x=805. y=2.477e+04

Results

benchmark file	initial cost	final cost	improvements
alu2	5667	1664	71%
apex1	33587	12000	64%
apex4	69314	23798	66%
c880	5781	1989	66%
cm138a	139	57	59%
cm150a	235	105	55%
cm151a	208	51	75%
cm162a	363	121	67%
cps	35769	11251	69%
e64	11284	4134	63%
paira	41435	11452	72%
pairb	72997	15397	79%

(* `improvements = -(final_cost - initial_cost) / initial_cost *100%`)