EE5301 ProjectAssignment2 Report

Tianming Cui (cuixx327@umn.edu)

December 2019

1 Results

The test runtime results shown below are average values of 5 tests, other results are the median values chosen from 5 tests(for -a, it is the result with 3rd area, for -w, it is the result with 3rd hpwl, for -c, it is the result with 3rd area*hpwl, I use median since the average value here might be a impossible floorplan), all test results are generated on Keller-Hall Machine CSEL-KH1250-18.

The Average Initial HPWL and Area shows below, this initial floorplan is generated by randomly shuffle the positive and negative sequence, the random seed is the system time, so the initial floorplan could be different:

Ckt Name	Initial Area	Initial HPWL
n10	638685	31895
n30	723486	79050
n50	629256	158505
n100	621622	272195
n200	702780	603769
n300	1060560	1.00E+06

Figure 1: Initial Table

The Annealing result shows below:

Ckt Name	Argument	Chip Width	Chip Height	HPWL	Area	Runtime(s)
n10	-a	349	697	22557	243253	0.566396
n10	-w	672	404	13152	271488	0.783779
n10	-c	586	433	15533	253738	0.780263
n30	-a	532	451	48085	239932	3.00258
n30	-w	490	592	36145	290080	3.59585
n30	-c	466	514	41995	239524	3.619
n50	-a	408	562	113267	229296	7.73696
n50	-w	555	465	87912	258075	9.02315
n50	-c	457	517	102318	236269	9.00953
n100	-a	509	442	192341	224978	27.041
n100	-w	497	534	144535	265398	29.2903
n100	-c	474	474	166904	224676	29.259
n200	-a	518	466	443417	241388	102.551
n200	-w	534	529	347684	282486	106.832
n200	-c	480	503	368118	241440	106.901
n300	-a	669	585	734990	391365	227.716
n300	-w	719	675	558136	485325	231.299
n300	-c	645	634	615834	408930	230.76

Figure 2: Result Table

2 Implementation

2.1 Special Strategies

To reduce the runtime cost of the algorithm, the value NUM_MOVES_PER_TEMP_STEP (How many random moves within a certain iteration) is dynamically chosen in my implementation. It begin with a low value N at the beginning (picked as 6 in the implementation) and increases as 1.01N after each iteration, so that N will be increased to about 100 in the last iterations. This implementation can greatly reduce the runtime while keep a similar annealing result compares to set and fix N equals to 100 at the beginning. Since with the initially high temperature, most accepted moves are randomly and do not have a certain direction, this implementation is designed to reduce the non-directional moves and increase the directional (towards the optimal solution) moves.

Some special data structures are used in the implementation to obtain a better runtime. For each sequence-pair, two different kinds of vectors are generated, one is the original sequence (e.g. 1,4,2,3), while the other one suggests the position of each block in the sequence(e.g. for 1,4,2,3, the position vector is 0,2,3,1). By generating these position vectors, the program can quicker decide if a block is "above/below/left/right" of another block.

2.2 Parameters Selection

The initial temperature and cooling rate of the algorithm is chosen as same as the example given by the lecture slides: initial temperature is 40000, cooling rate is 95%. To obtain better annealing results, I chose freezing temperature as 0.01(1/10 of the example value) to make the annealing process longer and more complete.

The cost function with argument "-c" is chosen to be:
$cost = 0.5 * area + 0.5 * hpwl$

or
$cost = \propto *area + (1-\propto) * hpwl$ ($\propto = 0.5$)
The $\propto$ mark will be named as propto in the latter text.

The propto is chosen based on experiments. I tested propto = 0.3, propto = 0.5 and propto = 0.7, and recorded the results(I kept these options in the code and you can still test propto = 0.3 with option -cw, propto = 0.7 with option -ca).

Ckt Name	Propto	HPWL	Area	HPWL*Area
n10	0.3	19707	261230	5148059610
n10	0.5	15533	253738	3941312354
n10	0.7	19508	248368	4845162944
n100	0.3	169651	232190	39391265690
n100	0.5	166904	224676	37499323104
n100	0.7	177690	232960	41394662400
n300	0.3	589379	413952	243974615808
n300	0.5	615834	408930	251832997620
n300	0.7	653789	409716	267867813924

Figure 3: Result Table

By plotting the HPWL*Area as the overall performance (y-axis), we can get following figures, where the left bar shows propto = 0.3, the mid bar shows propto = 0.5, the right bar shows propto = 0.7:

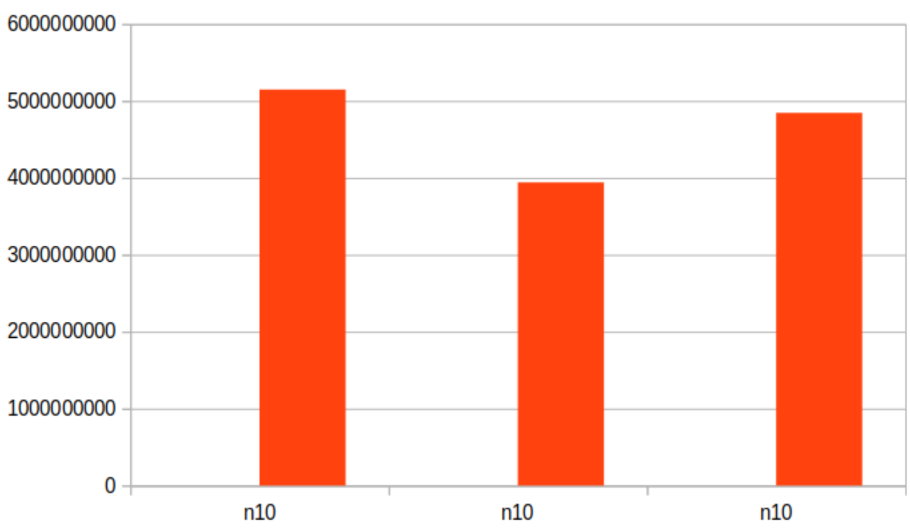

Figure 4: Result Table

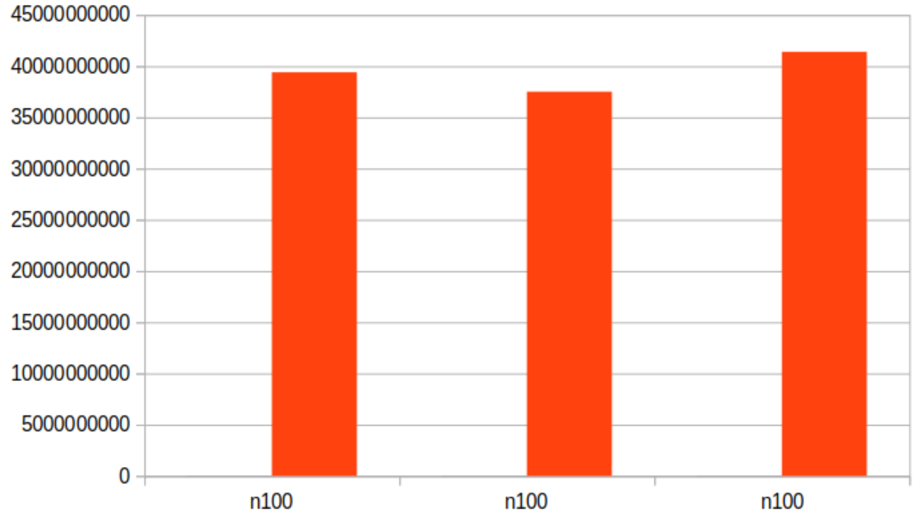

Figure 5: Result Table

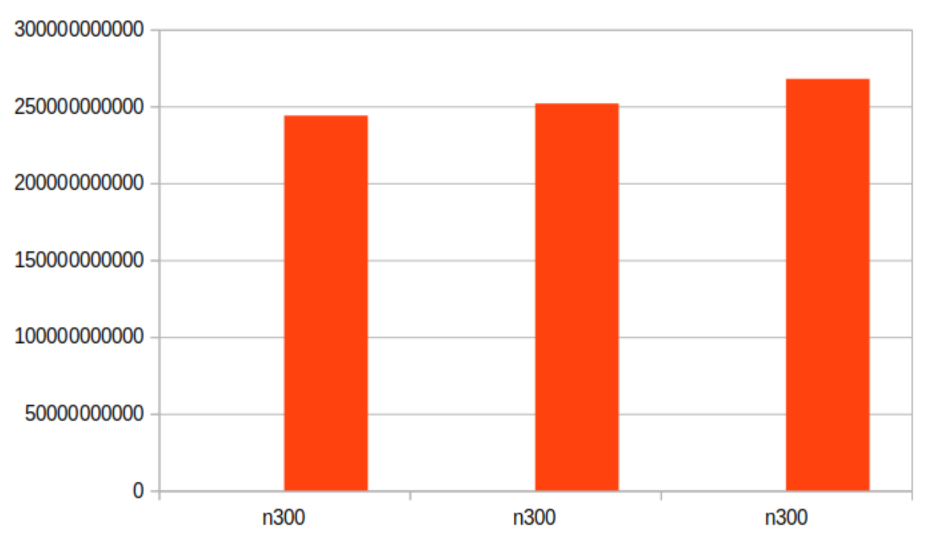

Figure 6: Result Table

The results shows that propto = 0.5 is a simple and balanced choice since this propto is always a trade-off. By set propto = 0.5, we can obtain both good AREA result and HPWL result.

The K value used in Boltzmann probability function is chosen based on the certain floorplanning file. The program runs N times random moves on the initial floorplan and records the average costs of these moves(N is proportional to the model number of the file). Then the K is computed to make 95% of the moves can be accept with the initial temperature. If we want about 100% of the moves can be accept at the beginning, K value should be closed to infinity and make the annealing too slow. The value 95% is inspired by the cooling rate and is tested to be a good choice.