Homework2 Report

(1) name: 唐梧遷　　student ID: 1100624267

(2)

```
--How to compile:

    In this directory, enter the following command:

    $ make

    It will generate the executable files "hw2" and "hw_parallel" in "HW2/bin"


    If you want to remove all of them, please enter the following command:

    $ make clean
```

```
--How to Run

    In "HW2/bin/", enter the following command:

    Usage: ./<exe> <net file> <cell file> <output file>

    e.g.:

    $ ./hw2 ../testcases/p2-1.nets ../testcases/p2-1.cells ../output/p2-1.out

    $ ../bin/hw2_parallel ../testcases/p2-1.nets ../testcases/p2-1.cells ../output/p2-1.out
```

(3)

testcase	cutsize	runtime	status
p2-1	6	0.02	success
p2-2	332	0.04	success
p2-3	5904	0.84	success
p2-4	42407	11.52	success
p2-5	122724	71.15	success

(4)

	P2-1	P2-2	P2-3	P2-4	P2-5
Final cut size	6	332	5904	42407	122724
runtime	0.02	0.04	0.84	11.52	71.15
I/O time	0	0.03	0.61	1.09	2.34
Comp time	0.02	0.01	0.23	10.43	68.81

可能是因为计算的速度太快，在小测资的情况下，IO time 甚至会超过 computation time。

(5)
I.

single thread：

演算法思路是相似的，但是具体实作上有差异。比如我使用了了退火算法。

当找到当前最优解之后，拿出所有 Gain 为 0 的 Cells 随机分配到集合 A 或 B。观察是否会出现 Gain>0 的 Cells 并移动。我在此设定一个 max 退火 depth=30000，并反复迭代。

multithread:

我在两个地方使用了平行处理:

一、 在 initial partition 完成之后，平行计算各个 Cell 的 Gain 值。

二、 在 update Gain 值时，使用了多线程计算被移动 Cell 的所有相关 Nets 是否为 ritical net。

II.可以算是有，但和课堂中 slide 中的不完全一样。因为 cells 的最大 number 和 nets 的最大 number 都小于 200000，所以直接把 number 当 index 来索引，用来加速。

```
struct Net{
    int cntL=0;
    vector<int> icells;
}nets[maxn];
```

```
struct Cell{
    int size,isleft=0,gain=0,ismoved=0;
    vector<int> inets;
}cells[maxn];
```

III.

每一次找到 max gain 的点，去判断是否可以移动，若可以，则移动，并和最佳 result 比较，如果超过了最佳 result 则用同样的资料结构复制一份保存。

IV.

一、 使用了大量的 bit operator 代替了 if 和 else。

二、 提前为 cells 和 nets 分配了连续的记忆体。

三、 用 maxheap 来取得 max gain 的 cell。

四、 对 initial partition 做了特殊处理，用 bfs 按 nets 去走访每个 cells 来分割 cells，尽量在 initial partition 时减少 cut size。

V.

一、 在 initial partition 完成之后，平行计算各个 Cell 的 Gain 值。并且使用了 C++内建的 atomic 对每个 cells 的 gain 限制了原子操作。

```
struct Cell{
    int size,isleft=0,ismoved=0;
    atomic<int> gain=0;
    vector<int> inets;
}cells[maxn];
```

二、 在 update Gain 值时，使用了多线程计算被移动 Cell 的所

有相关 Nets 是否为 critical net，如果是，则依然在多线程中更新此 net 中的相关 cells 的 gain 值。使用了 c++内建的 mutex 用来为 max heap 加锁。

```
function calcGain=[](int cidx){
        for(int nidx:cells[cidx].inets){
            int cntL=nets[nidx].cntL,cntR=nets[nidx].icells.size()-
nets[nidx].cntL;
            if(cells[cidx].isleft&&cntL==1||!cells[cidx].isleft&&cn
tR==1)++cells[cidx].gain;
            if(cells[cidx].isleft&&cntR==0||!cells[cidx].isleft&&cn
tL==0)--cells[cidx].gain;
        }
        mtxMaxGain.lock();
        if(cells[cidx].gain>=0)maxGain.emplace(cidx,cells[cidx].gai
n);
        mtxMaxGain.unlock();
    };
```

实验结果：

	P2-1	P2-2	P2-3	P2-4	P2-5
Final cut size	77	1056	22433	57717	133365
runtime	0.03	0.24	4.43	7.79	27.61
I/O time	0.01	0.02	0.17	0.31	0.47
Comp time	0.02	0.22	4.26	7.48	27.14

在大测资的时候，速度明显有了提高。

由于平行化版本，我没有对 initial partition 进行优化。所以 min cut size 上会略差。

(6)

testcase	cutsize	runtime	status
p2-1	6	0.02	success
p2-2	332	0.04	success
p2-3	5904	0.84	success
p2-4	42407	11.52	success
p2-5	122724	71.15	success

	Cut size					Runtime (s)				
Ranks	p2-1	p2-2	p2-3	p2-4	p2-5	p2-1	p2-2	p2-3	p2-4	p2-5
1	6	191	4441	**43326**	**122101**	0.01	0.07	3.05	5.01	42.06
2	6	**161**	1065	43748	125059	0.01	0.1	3.11	9.84	18.77
3	6	358	2186	45430	122873	0.04	0.78	21.21	115.38	59.78
4	**5**	302	1988	46064	124862	0.03	0.17	7.04	6.93	8.22
5	6	411	**779**	46356	125151	0.01	0.16	5.49	12.31	13.57

p2-1 和去年第二持平。

p2-4 赢了去年最佳解，并且时间上相差不大。

总体来说可以和去年的前五个最佳解答不相上下，可能是我加入了退火算法的原因。

(7)

熟悉了多线程操作，比如 mutex，atomic 的用法。

了解了退火算法。对于资料结构的规划有了更清楚的认知。