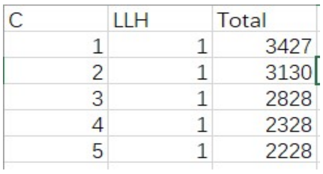
COMP5920 Scheduling Coursework 2

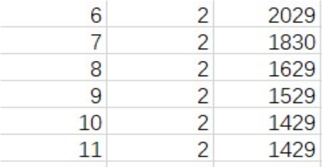
Meng Fanhui (sc19fm) 201373470

1. For this experiment, I plan to run each low-level heuristic component several times, until no apparent reduction of the solution cost then run the next LLH. For the chart below, C represents the total number of LLH activations, LLH represents which LLH is executed, "total" represents the total constraint violations.



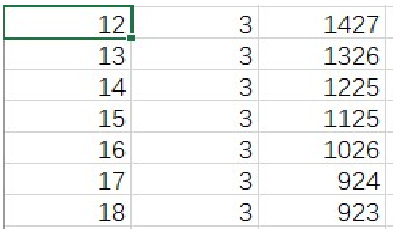
C	LLH	Total
1	1	3427
2	1	3130
3	1	2828
4	1	2328
5	1	2228

From the first to the 5th attempts, I run LLH 1, and the total constraint violations drop from initial 4027 to 2228. And I found there are only a small reduce between 4th and 5th attempts. Therefore, I think LLH 1 couldn't make the solution much better at this time. Then I run LLH 2 for the next time.



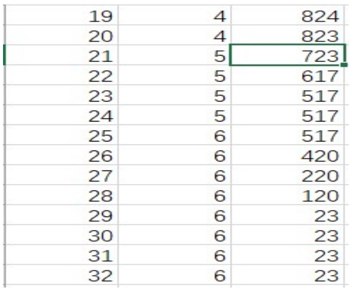
6	2	2029
7	2	1830
8	2	1629
9	2	1529
10	2	1429
11	2	1429

For LLH2, it does not make the solution much better, and for 10th and 11th attempts, the solution cost remain the same. So I move on to LLH3.



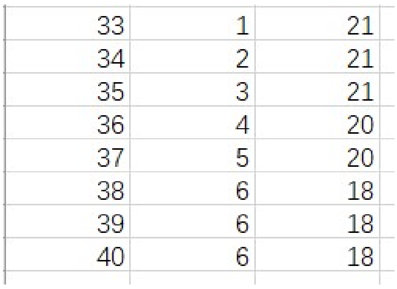
12	3	1427
13	3	1326
14	3	1225
15	3	1125
16	3	1026
17	3	924
18	3	923

For LLH3, each attempt only reduces about 100 to the solution cost, and I think this is acceptable, because the number is down to 1000, so it can't change too much. When the reduction down to single digits, move on to the LLH4.



19	4	824
20	4	823
21	5	723
22	5	617
23	5	517
24	5	517
25	6	517
26	6	420
27	6	220
28	6	120
29	6	23
30	6	23
31	6	23
32	6	23

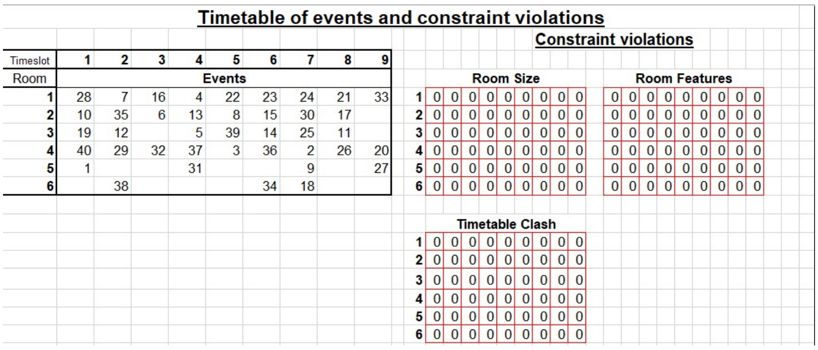
Then I do the same thing for the next few experiments. When it comes to small reduction, go for the following LLH algorithm. Finally, there is a remarkable decrease when I run LLH 6. The total constraint violations drop down to 23. And I do another few tries until the number is no longer change.



33	1	21
34	2	21
35	3	21
36	4	20
37	5	20
38	6	18
39	6	18
40	6	18

After another round of attempts, this solution's final total constraint violations remains 18, which is relatively good. Because there is no hard constraint violations in the timetable of the event as well as individual student timetable, thus, this solution satisfies every requirement (e.g. room size, room features, no clash in student timetable). All of these 18 constraint violations are in the green box, which means in the student timetable, there are three successive event violations and late event violations. And the latter two are highlighted in green, as they are soft constraints, which are not as serious as hard constraints. The experiment with my strategy gets a reasonably good solution.

The screenshots of the final solution timetable and timetable status show as below.



Timetable of events and constraint violations

Timeslot	1	2	3	4	5	6	7	8	9
Room	Events								
1	28	7	16	4	22	23	24	21	33
2	10	35	6	13	8	15	30	17	
3	19	12		5	39	14	25	11	
4	40	29	32	37	3	36	2	26	20
5	1			31			9		27
6		38				34	18		

Constraint violations

Room Size

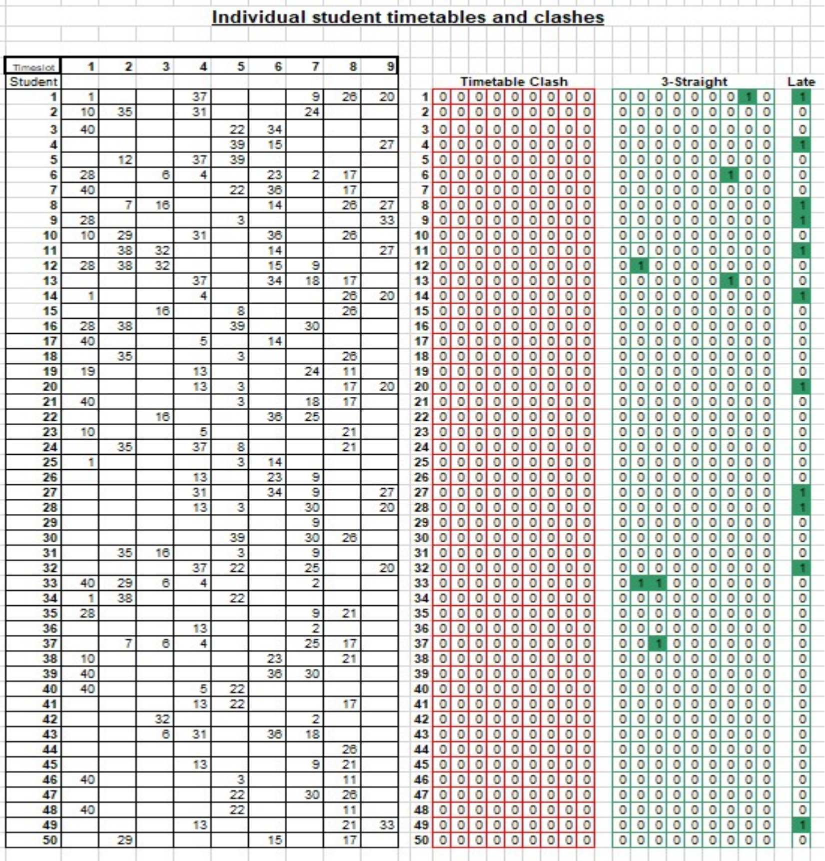
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0

Room Features

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Timetable Clash

1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0



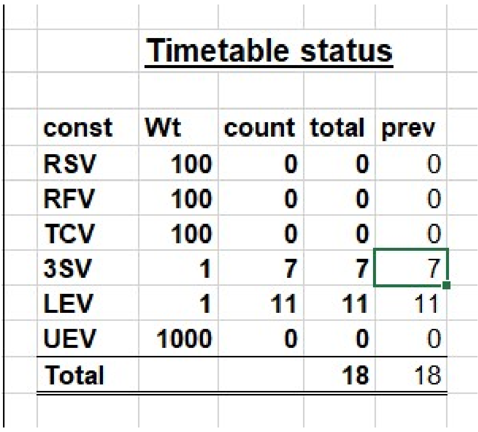
Individual student timetables and clashes

Timeslot	1	2	3	4	5	6	7	8	9
Student									
1	1			37			9	26	20
2	10	35		31			24		
3	40				22	34			
4					39	15			27
5		12		37	39				
6	28		6	4		23	2	17	
7	40				22	36		17	
8		7	16			14		26	27
9	28				3				33
10	10	29		31		36		26	
11		38	32			14			27
12	28	38	32			15	9		
13				37		34	18	17	
14	1			4				26	20
15			16		8			26	
16	28	38			39		30		
17	40			5		14			
18		35			3			26	
19	19			13			24	11	
20				13	3			17	20
21	40				3		18	17	
22			16			36	25		
23	10			5				21	
24		35		37	8			21	
25	1				3	14			
26				13		23	9		
27				31		34	9		27
28				13	3		30		20
29							9		
30					39		30	26	
31		35	16		3		9		
32				37	22		25		20
33	40	29	6	4			2		
34	1	38			22				
35	28						9	21	
36				13			2		
37		7	6	4			25	17	
38	10					23		21	
39	40					36	30		
40	40			5	22				
41				13	22			17	
42			32				2		
43			6	31		36	18		
44								26	
45				13			9	21	
46	40				3			11	
47					22		30	26	
48	40				22			11	
49				13				21	33
50		29				15		17	

Timetable Clash

1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0
38	0	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0	0
43	0	0	0	0	0	0	0	0	0
44	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0
48	0	0	0	0	0	0	0	0	0
49	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0

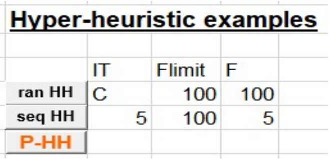
3-Straight									Late
0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1
0	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0



Timetable status

const	Wt	count	total	prev
RSV	100	0	0	0
RFV	100	0	0	0
TCV	100	0	0	0
3SV	1	7	7	7
LEV	1	11	11	11
UEV	1000	0	0	0
Total			18	18

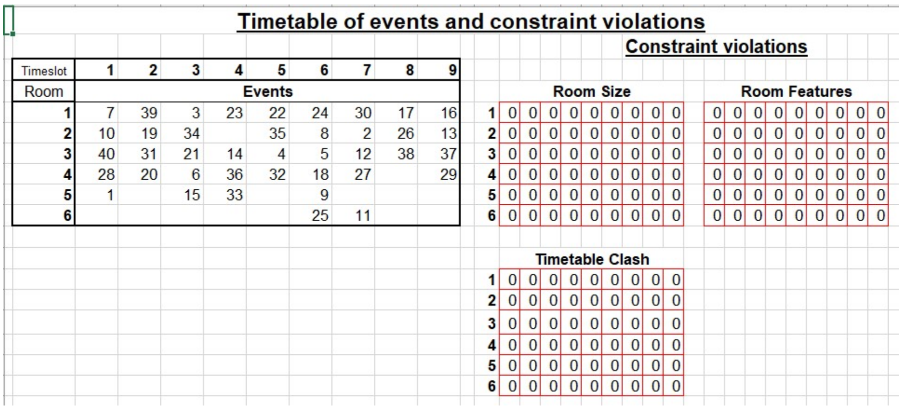
2. Change IT parameter value to C



Hyper-heuristic examples

	IT	Flimit	F
ran HH	C	100	100
seq HH	5	100	5
P-HH			

Screenshots show as below.



Timetable of events and constraint violations

Constraint violations

Timeslot	1	2	3	4	5	6	7	8	9
Room	Events								
1	7	39	3	23	22	24	30	17	16
2	10	19	34		35	8	2	26	13
3	40	31	21	14	4	5	12	38	37
4	28	20	6	36	32	18	27		29
5	1		15	33		9			
6						25	11		

Room Size

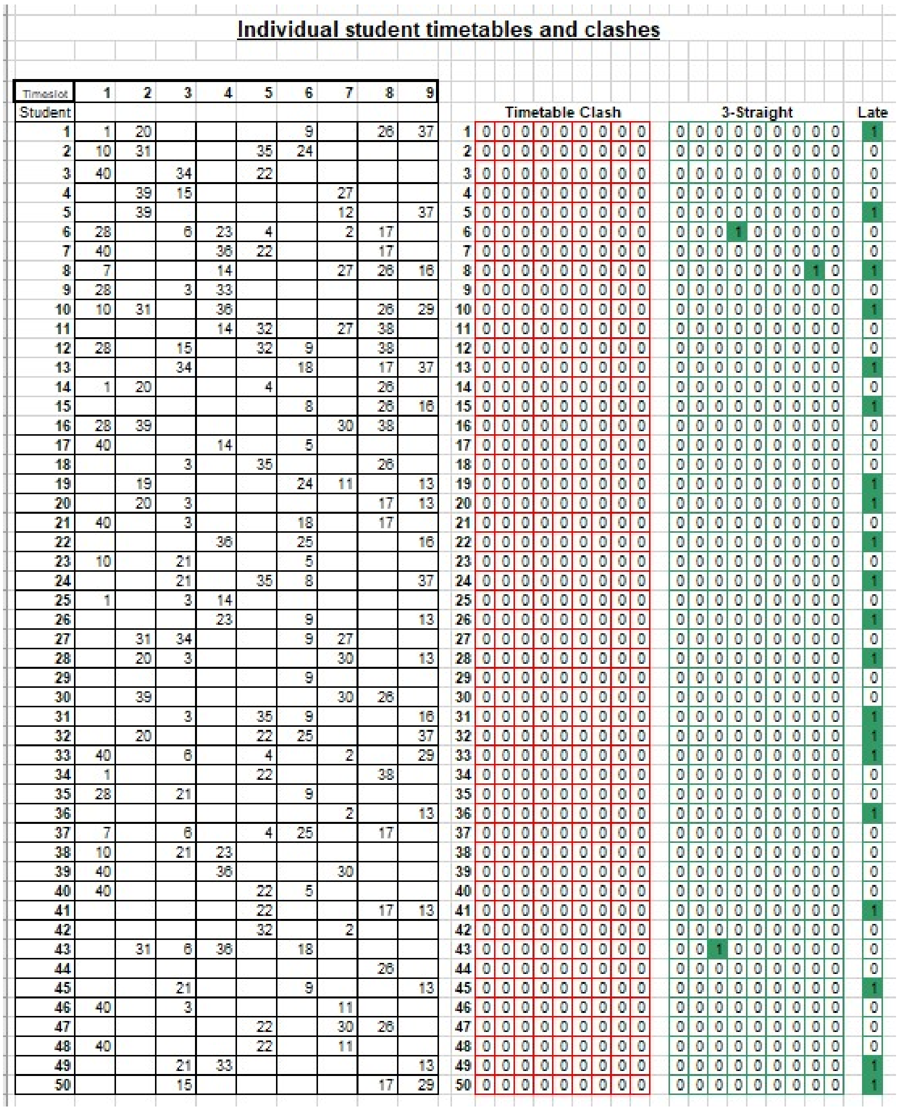
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0

Room Features

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Timetable Clash

1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0



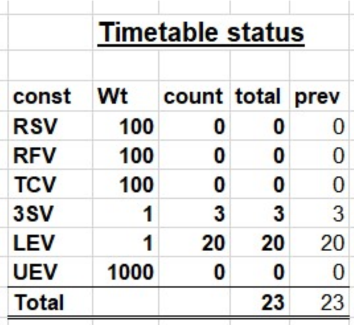
Individual student timetables and clashes

Timeslot	1	2	3	4	5	6	7	8	9
Student									
1	1	20				9		26	37
2	10	31			35	24			
3	40		34		22				
4		39	15				27		
5		39					12		37
6	28		6	23	4		2	17	
7	40			36	22			17	
8	7			14			27	26	16
9	28		3	33					
10	10	31		36				26	29
11				14	32		27	38	
12	28		15		32	9		38	
13			34			18		17	37
14	1	20			4			26	
15						8		26	16
16	28	39					30	38	
17	40			14		5			
18			3		35			26	
19		19				24	11		13
20		20	3					17	13
21	40		3			18		17	
22				36		25			16
23	10		21			5			
24			21		35	8			37
25	1		3	14					
26				23		9			13
27		31	34			9	27		
28		20	3				30		13
29						9			
30		39					30	26	
31			3		35	9			16
32		20			22	25			37
33	40		6		4		2		29
34	1				22			38	
35	28		21			9			
36							2		13
37	7		6		4	25		17	
38	10		21	23					
39	40			36			30		
40	40				22	5			
41					22			17	13
42					32		2		
43		31	6	36		18			
44								26	
45			21			9			13
46	40		3				11		
47					22		30	26	
48	40				22		11		
49			21	33					13
50			15					17	29

Timetable Clash

1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0
38	0	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0	0
43	0	0	0	0	0	0	0	0	0
44	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0
48	0	0	0	0	0	0	0	0	0
49	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0

	3-Straight									Late
1	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	1
6	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	1	0	1
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	1
14	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	1
16	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	1
20	0	0	0	0	0	0	0	0	0	1
21	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	1
23	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	1
25	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	1
27	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	1
29	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	1
32	0	0	0	0	0	0	0	0	0	1
33	0	0	0	0	0	0	0	0	0	1
34	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	1
37	0	0	0	0	0	0	0	0	0	0
38	0	0	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0	0	0	1
42	0	0	0	0	0	0	0	0	0	0
43	0	0	1	0	0	0	0	0	0	0
44	0	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0	1
46	0	0	0	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0
48	0	0	0	0	0	0	0	0	0	0
49	0	0	0	0	0	0	0	0	0	1
50	0	0	0	0	0	0	0	0	0	1



Timetable status

const	Wt	count	total	prev
RSV	100	0	0	0
RFV	100	0	0	0
TCV	100	0	0	0
3SV	1	3	3	3
LEV	1	20	20	20
UEV	1000	0	0	0
Total			23	23

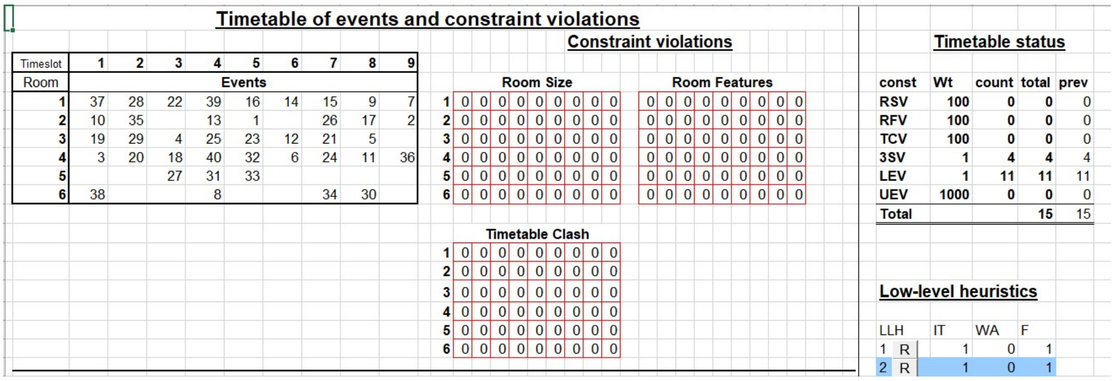
3. Pseudocode of my P-HH shows as below.

```
public class pseudocode{
    //define counter for each LLH activations and the total activation
    public static int counter = 0;
    public static int counter1 = 0;
    public static int counter2 = 0;
    public static int counter3 = 0;
    public static int counter4 = 0;
    public static int counter5 = 0;
    public static int counter6 = 0;
    //method for initialize
    public static void initialize(int[] ranking){
        //Run each LLH independently for 10 times
        total = getTotal;
        for(int i = 0; i < 10; i++){
            call LLH1;
        }
        //get the total constraint violations after 10 times LLH1
        total1 = getTotal();
        //numerical the performance of LLH1
        //Using the total constraint violations at the very begining
        //minus the current violations
        total1 = total - total1;
        //put it into the array, storing the ranking
        ranking.append(total1);
        //restart, prepare for the next LLH
        restart();

        //for LLH2
        for(int i = 0; i < 10; i++){
            call LLH2;
        }
        total2 = getTotal();
        total2 = total - total1;
        ranking.append(total2);
        restart();

        //I wouldn't write all of them, cause it's the same for the
        //rest LLHs initialization
    }

    //find the minimum, which is the best performance in the ranking
    //array
    public static int findMax(int [] a){
        int max;
        int tmp = a[0];
        for(int i = 0; i < length.a - 1; i++){
            if(a[i] > tmp){
                max = i;
                tmp = a[i];
            }
        }
        return max;
    }

    //For those LLH who has not been selected for a while
    public static void rarely(){
        //if 10 attempts and none of them are LLH1, then assign 15
       //weight to it.
        if ((counter1 / counter) < 0.1){
            total1 += 50;
        }
        else if ((counter2 / counter) < 0.1){
            total1 += 50;
        }
       //I'm not gonna write all of it in here, cause they all the same
    }
    public static void restart(){

    }
    public static void main(String[] args){

        //array for storing ranking
        int [] ranking;
        //initialize
        initialize(ranking);

        //drop out of the loop until total constraint violation is
       smaller// than 50
```

```
        do{
            //update total constraint violations
            int total = getTotal();
            //For those LLH who has not been selected for a while
            rarely();
            //find the max, which is the best performance in the
            //ranking array
            findMax(ranking);
            switch(max){
                case 0:
                    call LLH1;
                    //update it's performance
                    //to judge it's performance is to use the total1
                    //(newly total constraint violations) minus by the
                    //current violations
                   //then replace the corresponding number in the
                    //ranking array
                    total1 = total - total1;
                    ranking[0] = total1;
                    //update counter for LLH1
                    counter1++;
                    //update counter for all attempts
                    counter++;
                    break;
                case 1:
                    call LLH2;
                    //update it's performance
                    total2 = total – total2;
                    ranking[1] = total2;
                    counter2++;
                    counter++;
                    break;
                case 2:
                //the same as above
                case 3:
                //the same as above
                case 4:
                //the same as above
                case 5:
                //the same as above
            }
        }while(total > 20);
    }
}
```

4. Please see the Excel file and the log.txt file, which is my P-HH log file.
5. With my strategy for the first time, it took me 40 times of LLH activations to find the best solution, which the violations are 18.

   The result of my P-HH experiment: it took about 5 minutes and 50 attempts to find out the initialization of each LLH's performance. And it took another 3 minutes and 42 attempts to reach the best solution, which the violations are 15.



Timetable of events and constraint violations

Timeslot	1	2	3	4	5	6	7	8	9
Room	Events								
1	37	28	22	39	16	14	15	9	7
2	10	35		13	1		26	17	2
3	19	29	4	25	23	12	21	5	
4	3	20	18	40	32	6	24	11	36
5			27	31	33				
6	38			8			34	30	

Constraint violations

Room Size

1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0

Room Features

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Timetable Clash

1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0

Timetable status

const	Wt	count	total	prev
RSV	100	0	0	0
RFV	100	0	0	0
TCV	100	0	0	0
3SV	1	4	4	4
LEV	1	11	11	11
UEV	1000	0	0	0
Total			15	15

Low-level heuristics

LLH		IT	WA	F
1	R	1	0	1
2	R	1	0	1

As for random LLH, the performance of this hyper-heuristic is all about luck. I did some experiments on it, and the result can be very different. One experiment took me 10 minutes, and the final total constraint violations are 113. Another experiment also took about 10 minutes, but it reaches 17 of the violation at the first 3 minutes. In conclusion, this algorithm picks the random LLH for no reason so that it can be the worst or sometimes the best algorithm.

All in all, find and use the best performance LLH can be the most efficient way of finding the best solution. But the performance is not fixed, so keep update the performance and use the best performance at a specific time is always the best choice. Perhaps, it's not the most efficient way when it compares with other methods. Like when I do the first experiment, the attempts are less than the P-HH attempts to find the best solution. However, it's stable. It can always find a suitable solution for a short period.