CS6135 HW5 Global Routing Report

109062509 熊 磊

I. How to compile and execute?

- Compilation:

  One can go to directory, `HW5/src`, and execute `make`, the executable file, named as `hw3`, will be generated in directory, `HW5/bin`.

```
[g109062509@ic53 src]$ make
```

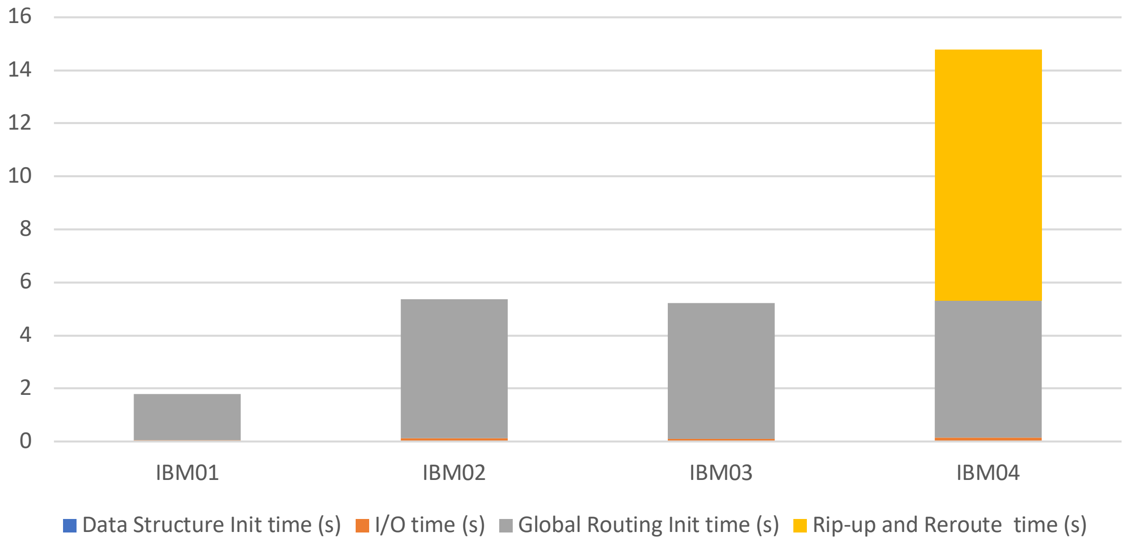
- Execution

  In directory, `HW5/bin`, execute `hw5` follow by the require files.

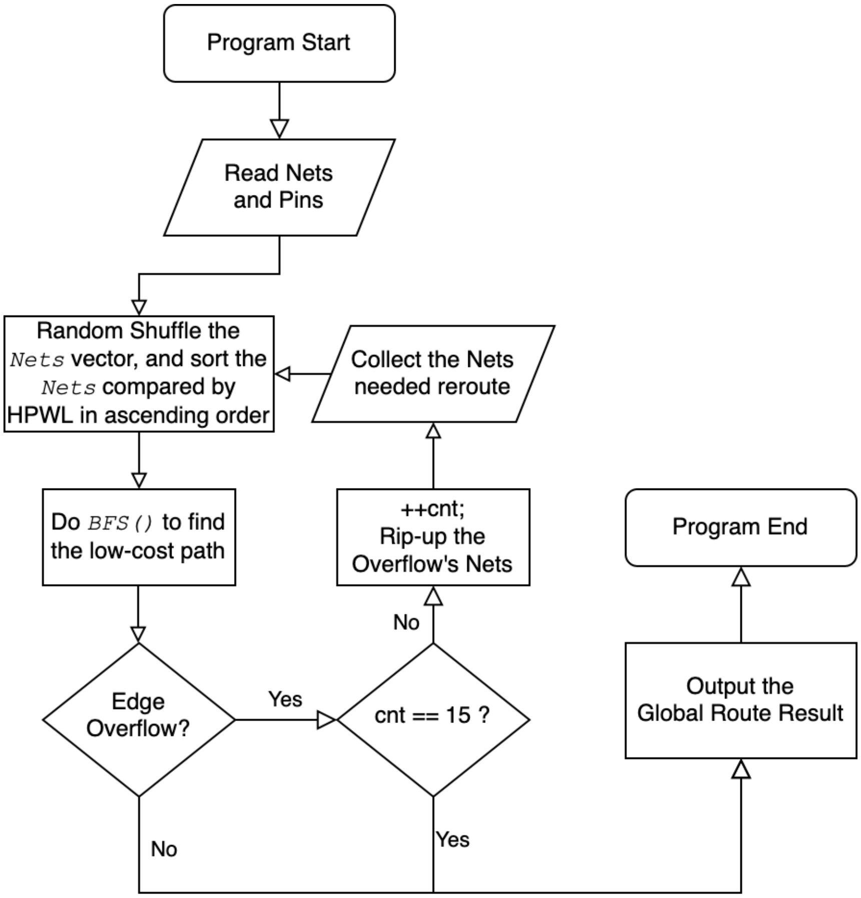
```
$ ./hw5 *.modified.txt *.result
```

II. The wirelength and the runtime of each testcase.

	IBM01	IBM02	IBM03	IBM04
Overflow	**0**	**0**	**0**	**91**
Wirelength	**60025**	**157550**	**142502**	**160660**
Data Structure Init time (s)	0	0	0	0
I/O time (s)	0.05	0.13	0.1	0.14
Global Routing Init time (s)	1.74	5.23	5.12	5.19
Rip-up and Reroute time (s)	0	0	0	9.46



III. The details of your implementation. You have to use flow chart(s) to help elaborate your algorithm, and please follow the symbols usually used in flow charts. If your method is similar to some previous works/papers, please cite the papers and reveal the difference(s).


Program Start
Read Nets and Pins
Random Shuffle the Nets vector, and sort the Nets compared by HPWL in ascending order
Collect the Nets needed reroute
Do BFS() to find the low-cost path
++cnt; Rip-up the Overflow's Nets
Program End
No
Edge Overflow?
Yes
cnt == 15 ?
Output the Global Route Result
No
Yes

Since in this lab, each net is a two-pin-net, hence the method I implemented is to perform BFS from the *start*-pin to the *end*-pin. As maze solving algorithm, while here we use max-heap (`std::prioity_queue`), instead of normal queue, to consider the lowest-usage edge for routing by calculating each path current's cost. While this idea is similar as Lee Algorithm[1] and NTHU Route 2.0[2].

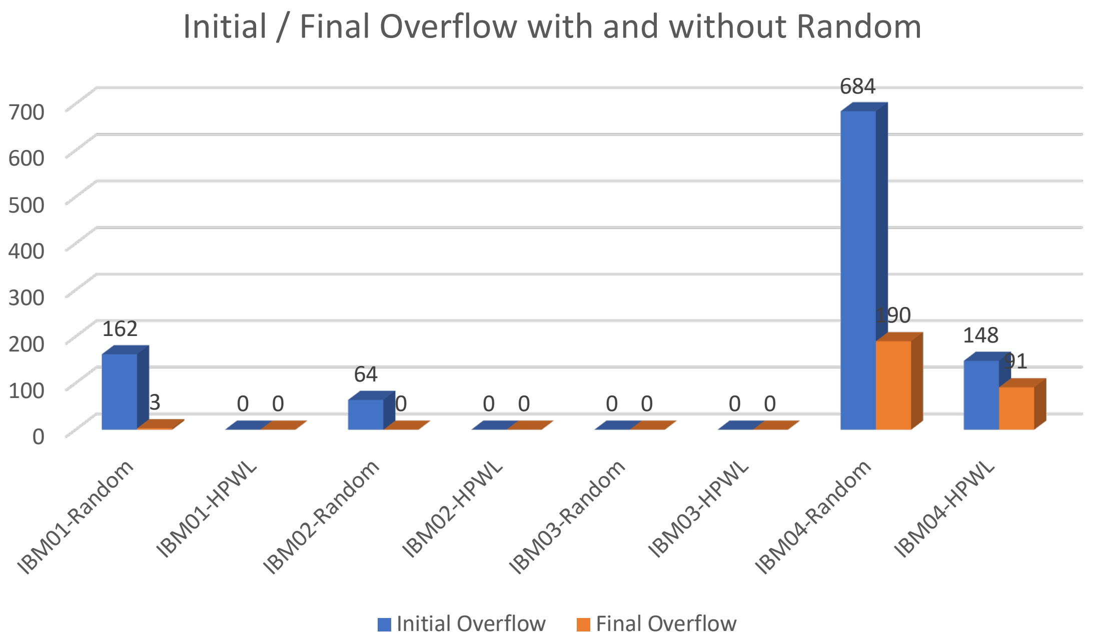
After the initial route, it is possible to have overflow for some edges, we then perform the Rip-up and Reroute for at most *num_reroute* times until the *total_overflow*`==0`. Since we do some tricks in the initial stage of routing, detail explanation in the next section, I can easily generate zero-overflow route in initial stage.

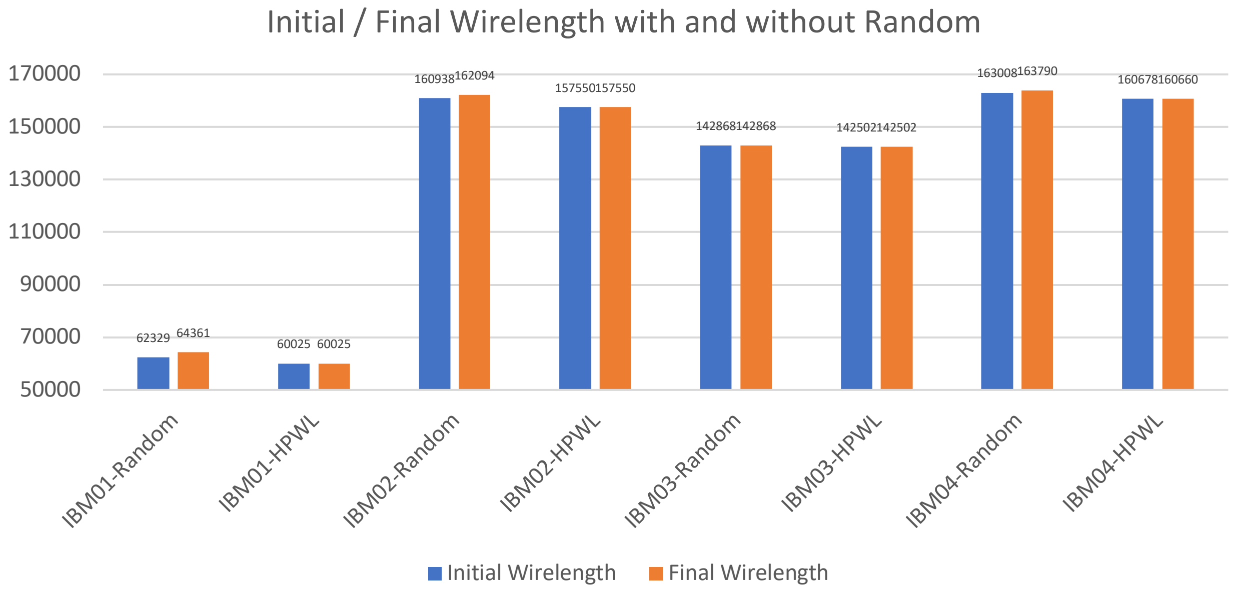
[1] C. Y. Lee, "An Algorithm for Path Connections and Its Applications," *IEEE Trans. Electron. Comput.*, vol. EC-10, no. 3, pp. 346–365, Sep. 1961, doi: 10.1109/TEC.1961.5219222.

[2] Y. Chang, Y. Lee and T. Wang, "NTHU-Route 2.0: A fast and stable global router," 2008 IEEE/ACM International Conference on Computer-Aided Design, San Jose, CA, 2008, pp. 338-343, doi: 10.1109/ICCAD.2008.4681595.

IV. What tricks did you do to speed up your program or to enhance your solution quality? Also plot the effects of those different settings like the ones shown below.

1) Shuffle the Nets vector, and sort it in ascending order of its HPWL. This can let the router perform routing on net having small bounding box first.





2) I perform ***parallelization*** at the initial stage of routing, and use the lowest overflow routing in the next stage.

3) Save random seed in order to reproduce the empirical best result.

V. What have you learned from this homework? What problem(s) have you encountered in this homework?

In the beginning, I did not know which data structure to use to effectively record the edges and nodes used, and it was a bit messy to get the id of the edge. Similarly, how to calculate the cost for BFS to find the path correctly is also challenging, penalty is determined by empirical testing.

In this homework, I found that if start routing with nets having small HPWL, the program can quickly find the result with a small overflow, sometimes even zero, and its wirelength is also relatively short.

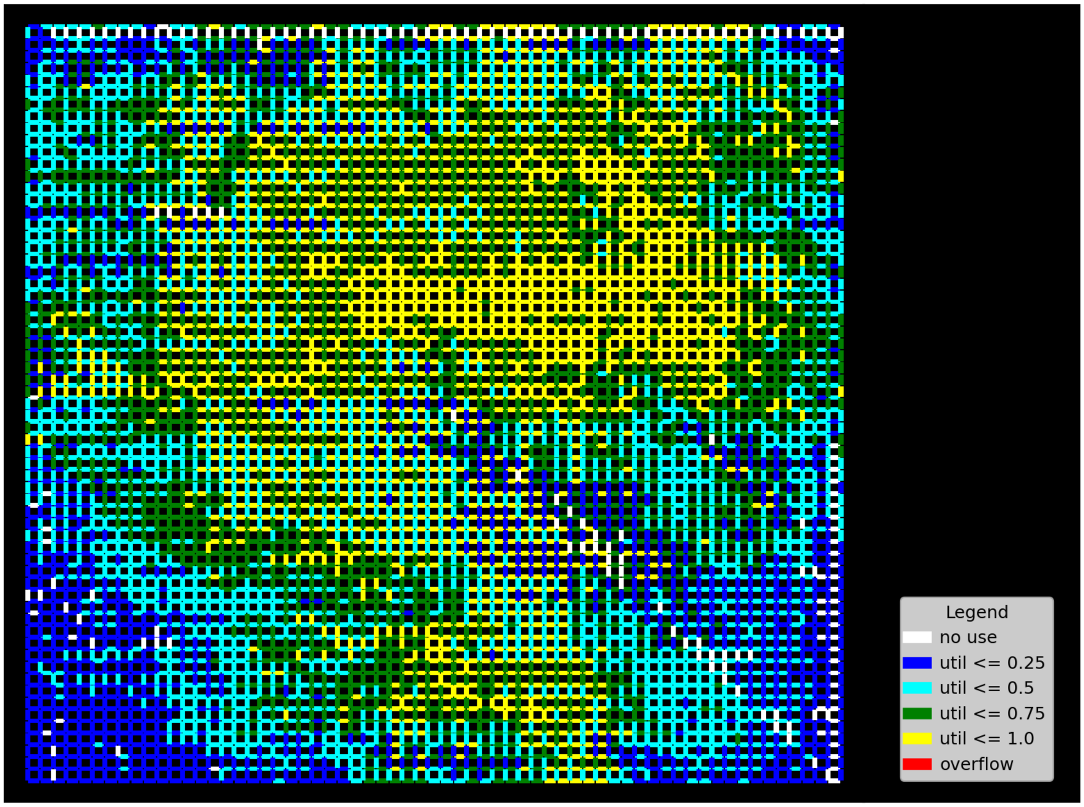
VI. History Comparison

Case	**IBM-01**			**IBM-02**			**IBM-03**			**IBM-04**		
Sol.	**OvF**	**WL**	**Time**	**OvF**	**WL**	**Time**	**OvF**	**WL**	**Time**	**OvF**	**WL**	**Time**
1st	0	59371	8.89	0	156556	14.41	0	142680	3.57	71	160342	158.51
1st	0	59519	3.91	0	157394	6.4	0	143000	2.91	63	159136	55.571
3rd	0	63495	6.57	0	166148	8.83	0	143894	13.21	101	162460	122.5
Lee Yellow	0	61271	33	0	161245	113	0	147086	91	285	161728	148
Romulus	0	62529	107.24	0	161486	350.2	0	144522	87.82	222	159408	590
Laplaceyc	0	60391	4.39	0	160066	16.58	0	143594	3.39	119	160888	55
Lei Hsiung	**0**	**60025**	**1.03**	**0**	**157550**	**3.04**	**0**	**142502**	**3.40**	**91**	**160660**	**12.73**

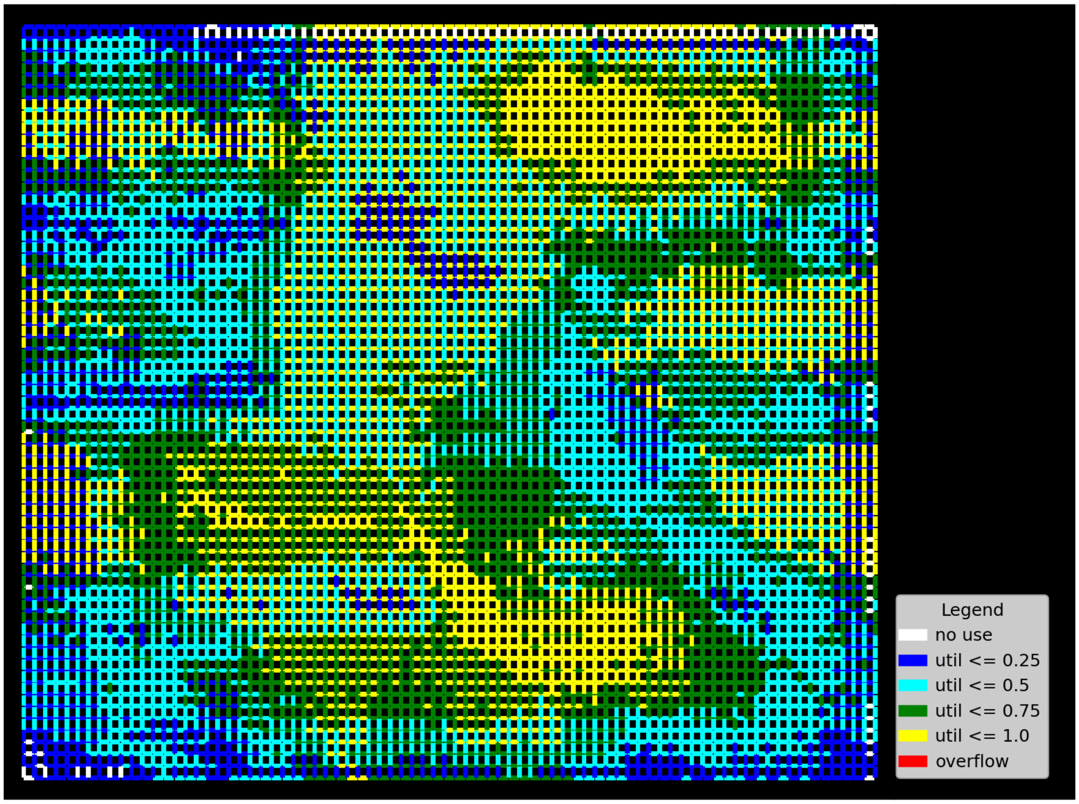
VII. Bonus (Parallelization)

In my program, one can specify the thread numbers to do the initial routing, it will remain the lowest overflow, first, and the smallest wirelength, second, for next stage, rip-up and reroute if there exists overflow.

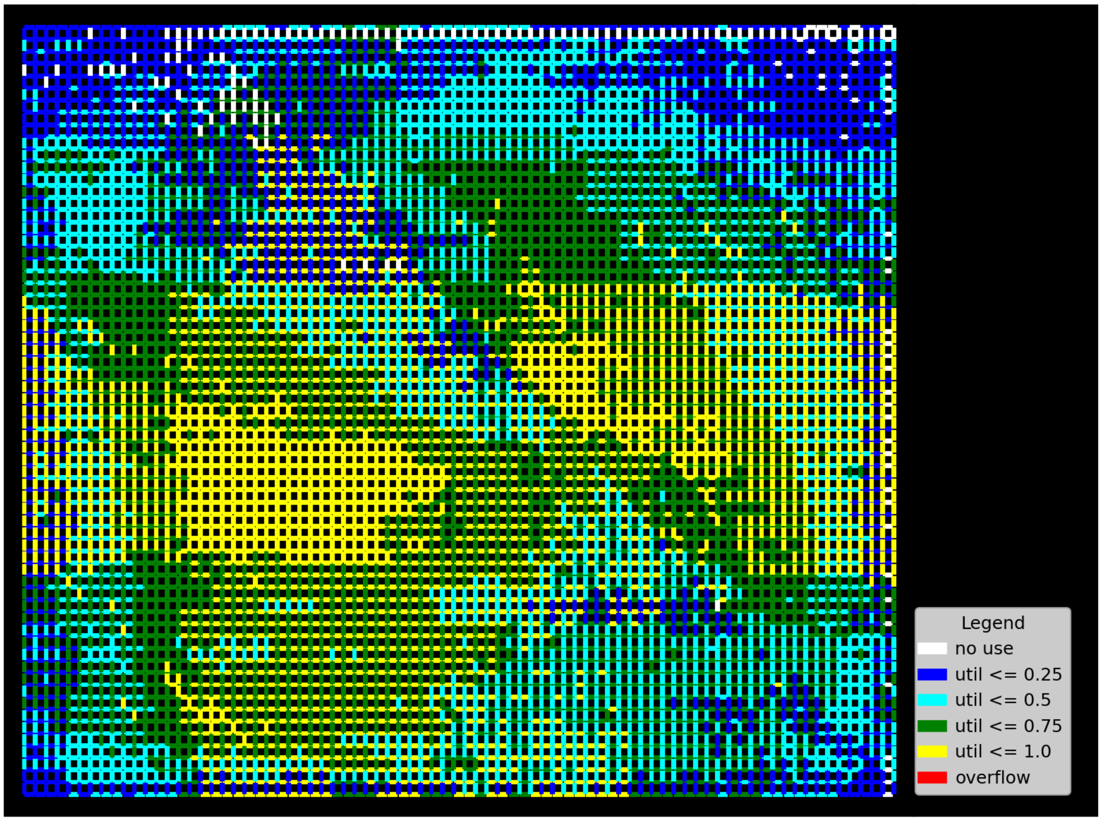
VIII. Bonus (Congestion Map)



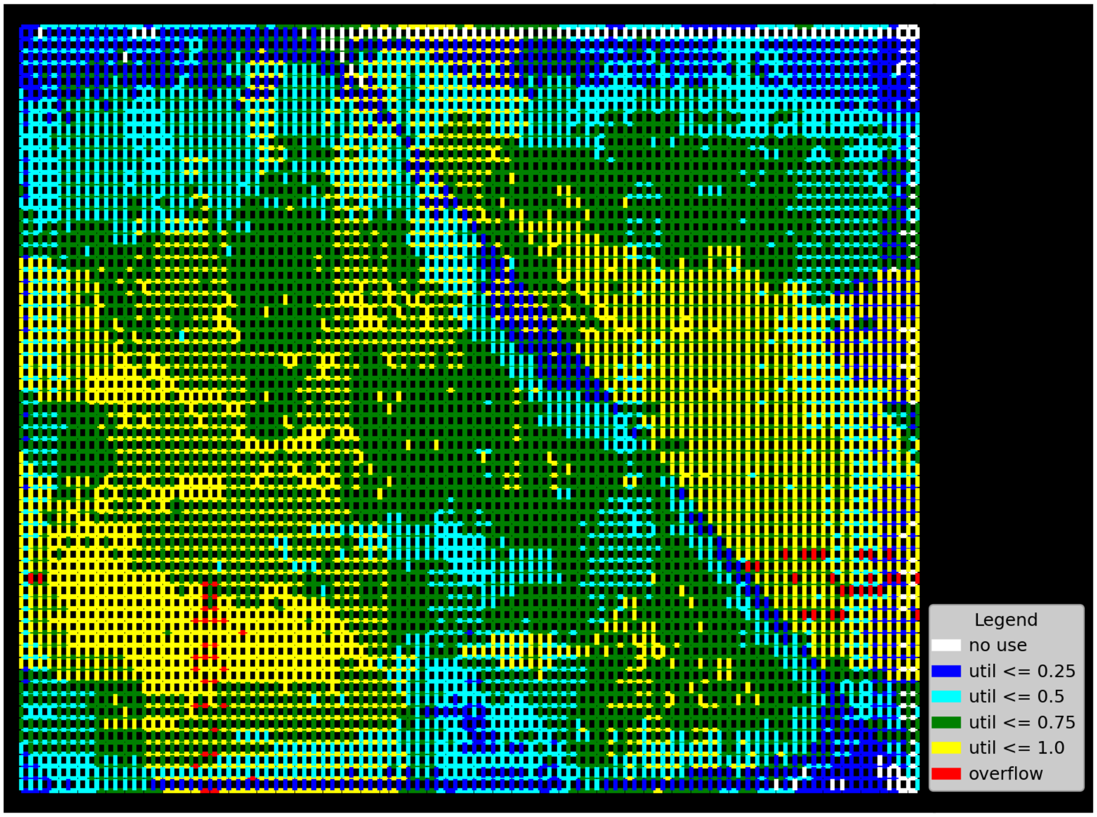
IBM-01



IBM-02


Legend
no use
util <= 0.25
util <= 0.5
util <= 0.75
util <= 1.0
overflow

IBM-03


Legend
no use
util <= 0.25
util <= 0.5
util <= 0.75
util <= 1.0
overflow

IBM-04