CSE - Homework 4

Yacine Badiss & Romain Yon

Due to the time taken by the computation of a single graph, we decided to work on the small versions of the graphs in order to be able to run many experiments.
The machine that we used for the benchmark is a notebook having 8GB of RAM, an i7 2.0 Ghz dual-core processor and running Ubuntu x64 on an 250GB SSD.

Here are, for each graph, the best edge cut we obtained while maintaining the partition size balance constraint. All the results we obtained will be included in the final submission.

Identifier	Software used	Partitioning type	Balance of output partition	Edgecut of output partition	Running Time
Graph A	METIS	● Sorted heavy-edge matching ● no2hop=YES ● minconn=NO ● contig=NO	1.03	1500	0.236s
Graph B	METIS	● Sorted heavy-edge matching ● no2hop=YES/NO ● minconn=YES/NO ● contig=NO	1.0002	686660	11.3s
Graph C	METIS	● random matching ● no2hop=YES ● minconn=YES ● contig=NO	1.03	771467	27.984s

Since partitioning graph is a NP hard problem, the different parameters we give to Metis enable to tune the heuristics underlying the partitioning. We can see that the efficiency of each heuristics depends mostly on the shape of the graph (i.e. there is no general rule). For example, it is more efficient to partition graph C using random edge matrix rather than selecting its most heavy weight edges first. However the opposite is true for graph A and B.

That is the reason why we tried most of the possible combinations of parameters for each graph (you can refer to the python script we submitted to check how we did this testing).

For graph B, we realized that using the option “minconn” was slowing down the computation by a factor of 3 without improving the result. This is because when we minimize the edgecut, we end up with partitions that are all connected together. Hence the maximum degree of the subdomain graph is always 63 in this case and cannot be minimized any further.

We noticed that, for both graph A and C, we reached the maximum balance of output partition size in order to obtain the minimum edgecut. We then tried to relax the balance constraint and got a decrease of edgecut doing so. For graph A, we set the constraint to 1.30, getting an edgecut decrease of 10%. For graph C, we set the constraint to 1.20, getting an edgecut decrease of only 1% (still, 1% for graph C is equal to 8000 edgecut). We tried with even less tight constraints and the edgecut kept on decreasing. However, this got us further and further from equal sized output partitions.