Data Mining Homework 5

K-way Graph Partitioning Using JaBeJa

Group 22 - Blanca Bastardés Climent, Alice Ciallella

Short description

The implementation of this assignment has been done in java. All the code is provided in the script **Jabeja.java** and the script **CLI.java** contains the hyperparameters used. The approach followed in this assignment consists in understanding distributed graph partitioning by means of implementing the Ja-Be-Ja algorithm.

Instructions to run the code

The following folders and files have the relevant information with the implementation of the code.

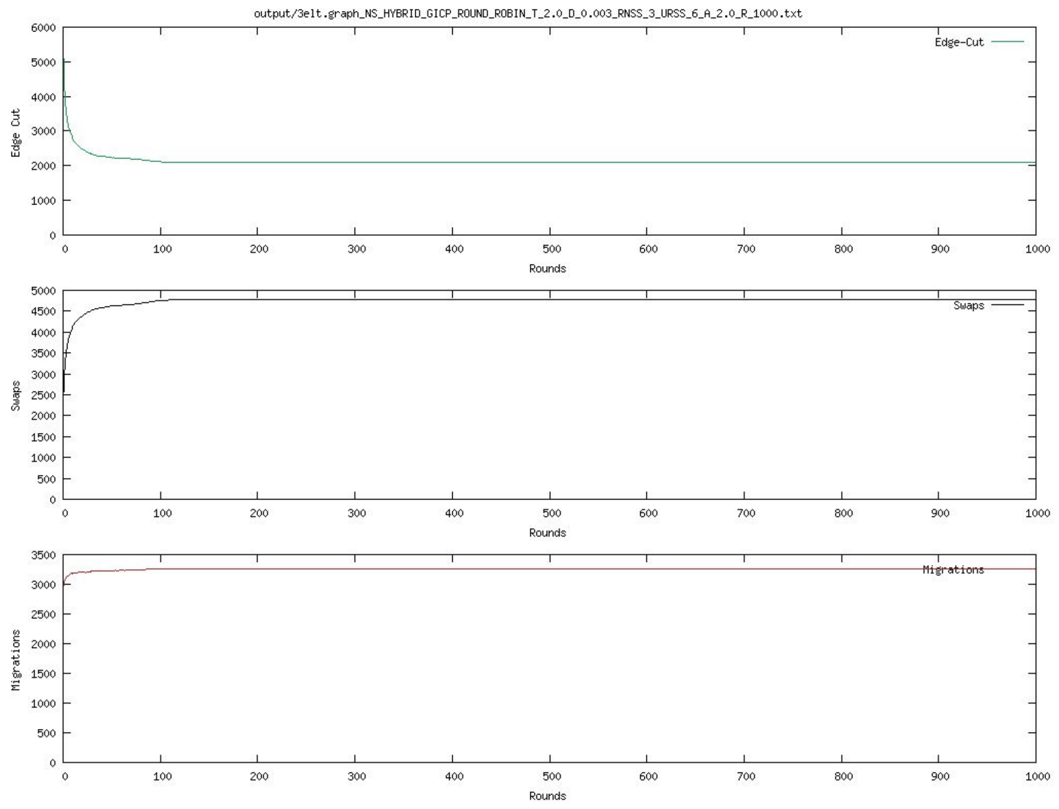
- **/src/main/java/se/kth/jabeja/Jabeja.java:** A java script containing code for JaBeJa algorithm and the different tasks assigned.

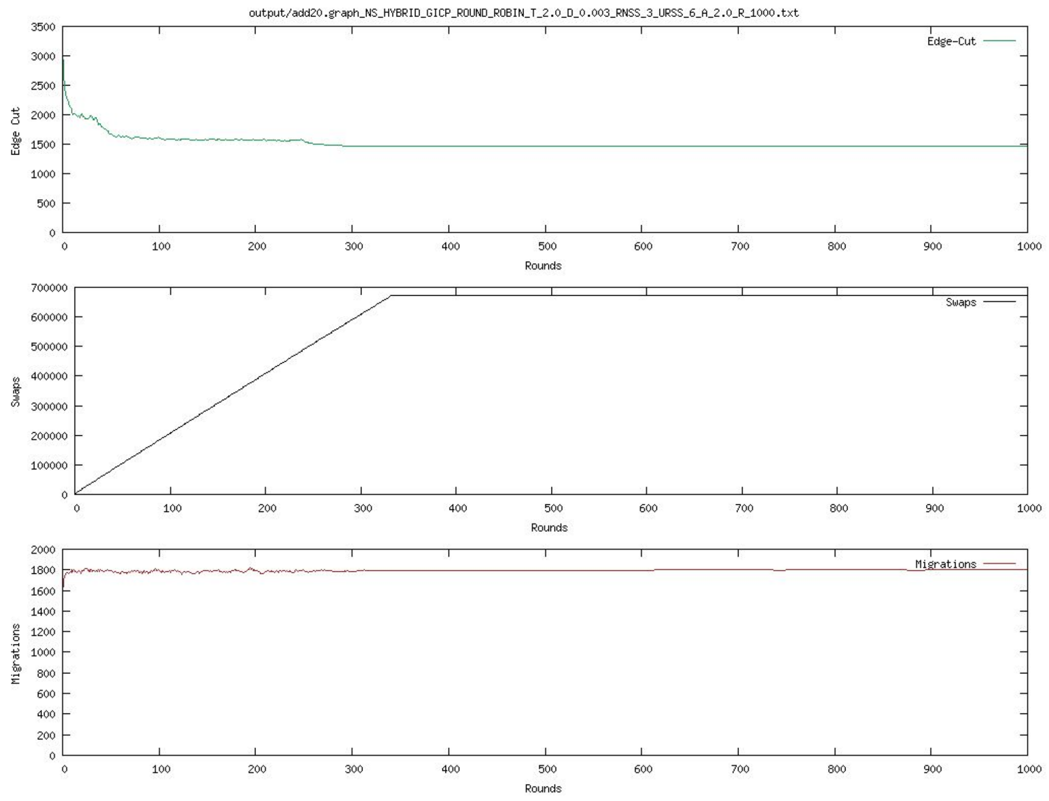
Results

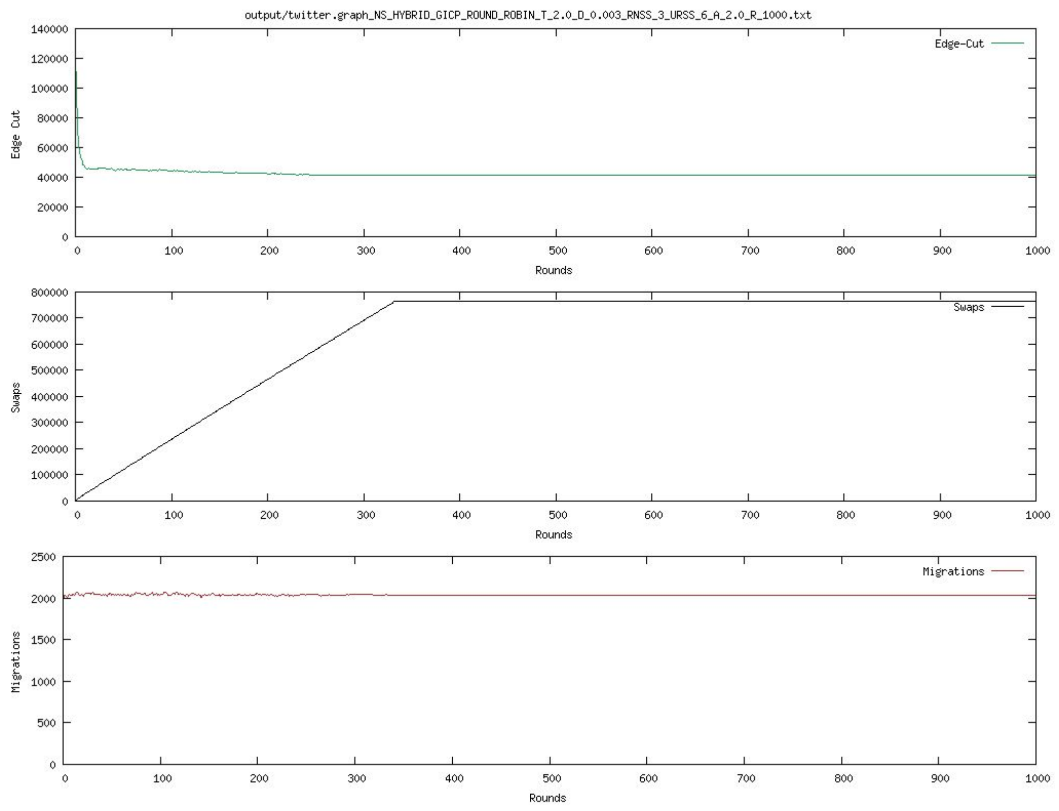
The results are divided in two sections. The three datasets studied are 3elt, add20 and twitter.

Task 1
In this task, the basic code of the algorithm is implemented following the one in the paper. Results for the three datasets with the default configuration can be seen in the next table and figures.

data	**T**	**delta**	**rounds**	**edgecut**	**swaps**	**migrations**
3elt	2	0.03	1000	1163	1307755	3369
add20	2	0.03	1000	1460	673453	1799
twitter	2	0.03	1000	41176	765501	2030


output/3elt.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_2.0_D_0.003_RNSS_3_URSS_6_A_2.0_R_1000.txt
Edge-Cut
Edge Cut
Swaps
Migrations
Rounds


output/add20.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_2.0_D_0.003_RNSS_3_URSS_6_A_2.0_R_1000.txt
Edge-Cut
Edge Cut
Swaps
Migrations
Rounds

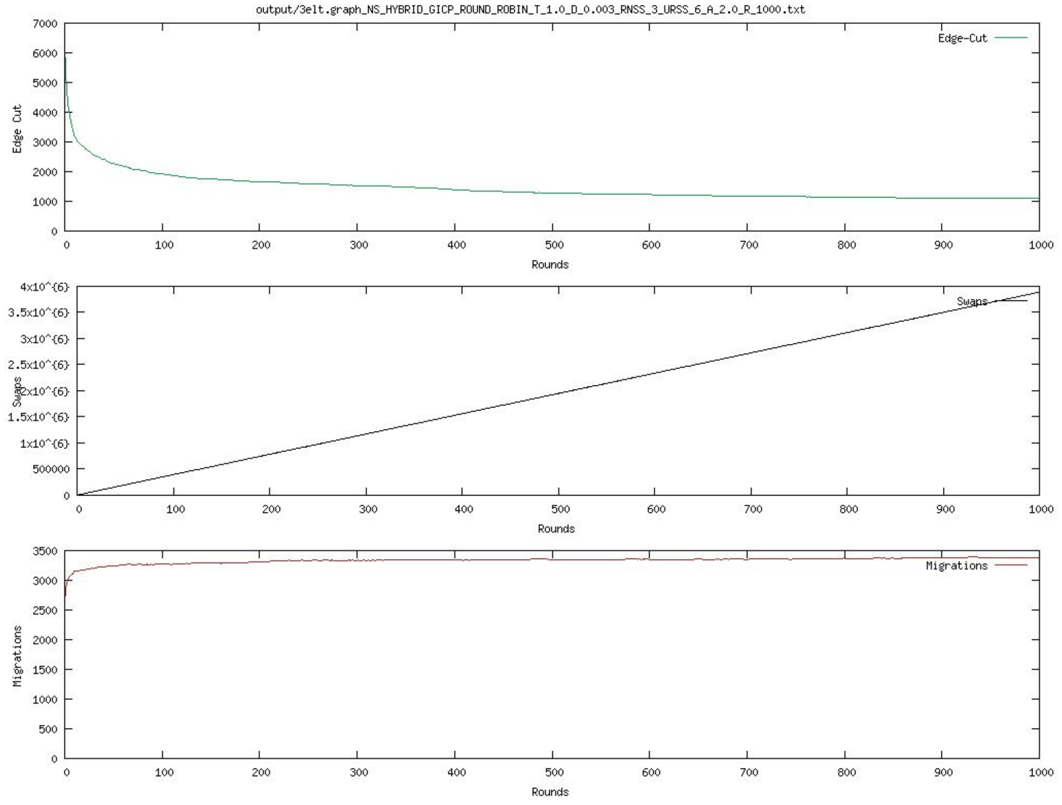


Task 2

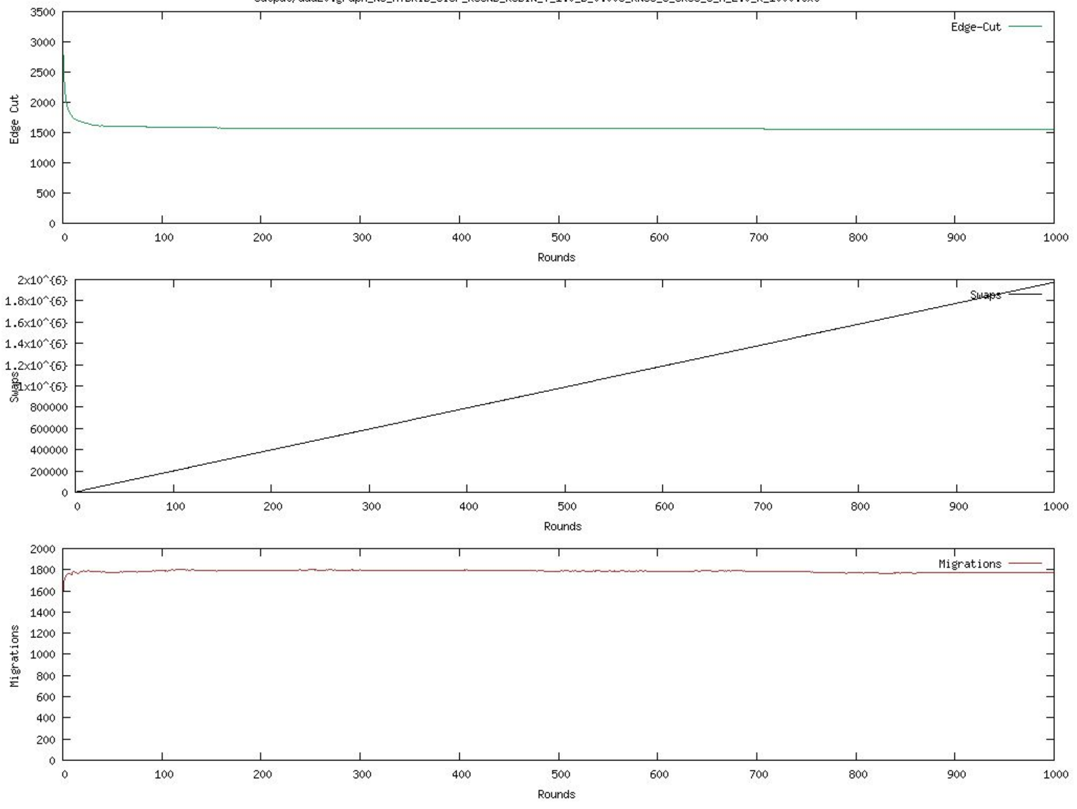
In this task, two different subtasks were done. For the first one, we explored a different mechanism to control the convergence of the algorithm. The original algorithm uses linear simulated annealing cooling function to accept a solution and keep track of the best one found so far; while in this case, we will explore an exponential approach. For this method, the maximum initial temperature is 1 and it will be decreased by multiplying it by *alpha* after each round. Typical choices for *alpha* values are between 0.8 and 0.99. We set it to 0.9.

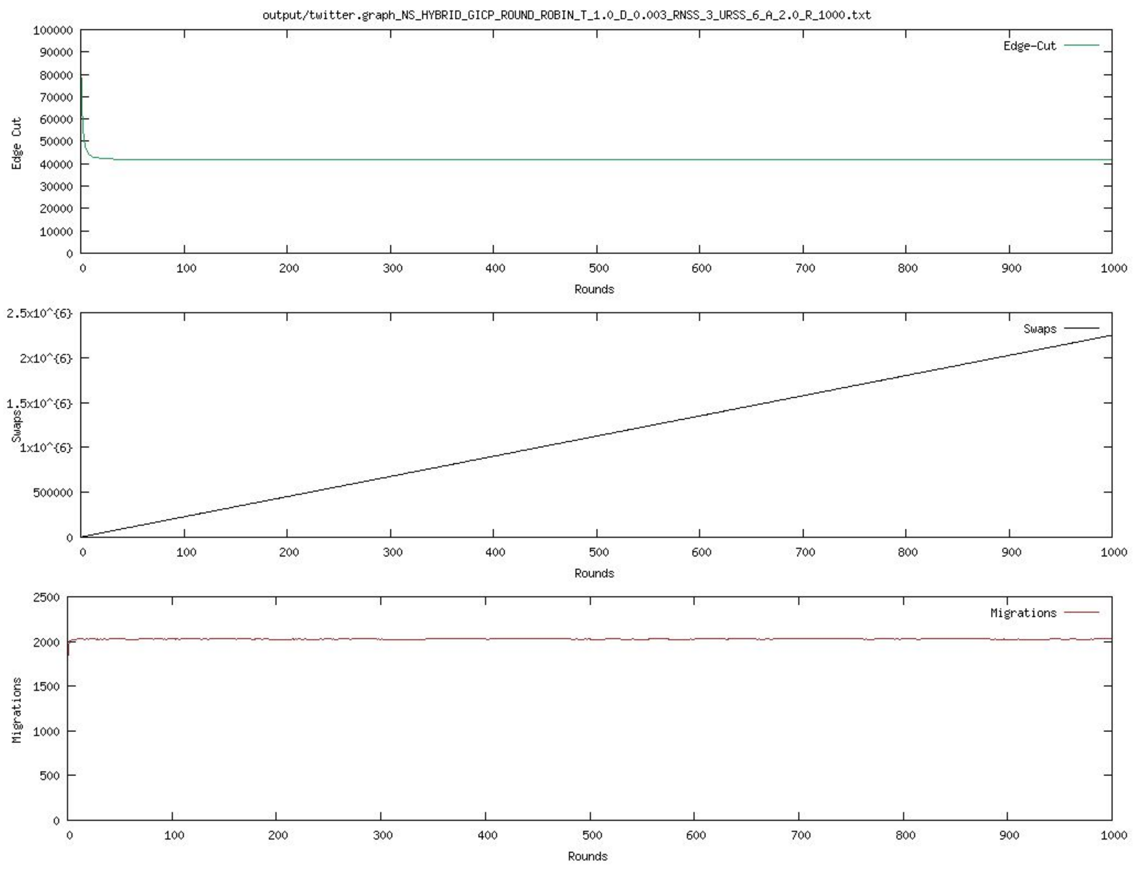
data	T	delta	rounds	edge cut	swaps	migrations
3elt	1	0.03	1000	1093	3885433	3382
add20	1	0.03	1000	1551	1972122	1771
twitter	1	0.03	1000	41793	2246781	2031

output/3elt.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_1.0_D_0.003_RNSS_3_URSS_6_A_2.0_R_1000.txt



output/add20.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_1.0_D_0.003_RNSS_3_URSS_6_A_2.0_R_1000.txt





After this new approach, we can see that although for the add20 and twitter graphs there are no big differences, in all three less swaps are needed to start reaching the convergence and for the 3elt graph, we can see how the edge cut decreased.

For the second subtask, the effect of restarting the simulated annealing after the algorithm has converged is explored. The frequency to be restarted is set to 400 rounds, preventing from getting stuck in local minima, accepting more swaps and looking for finding lower edge cuts. The results obtained are:

data	**T**	**delta**	**rounds**	**edge cut**	**swaps**	**migrations**
3elt	1	0.03	1000	1165	3888056	3371
add20	1	0.03	1000	1549	1971914	1803
twitter	1	0.03	1000	41744	2244797	2031

For the 3elt it helps avoiding convergence at a local minimum when comparing the results obtained from task1 and the same implementation but adding the reset option.

Optional
One possible approach to save computational time in case of finding no changes between several rounds consecutively, could be to set a counter to keep track of the rounds with no more swaps and after no further improvement finish the process. This could be beneficial for instance for the twitter graph since it reaches convergence very fast.