Report 5: K-way Graph Partitioning Using JaBeJa

Group 34: Sofia Krylova, Iosif Koen

December 10, 2022

1 Introduction

In this home assignment, we were to understand distributed graph partitioning using gossip-based peer-to-peer techniques, such as JaBeJa.

In order to accomplish this, we needed to perform the following steps:

1. Implement the Ja-Be-Ja algorithm, using a scaffolding source code for Ja-Be-Ja simulation for the one-host-one-node model;
2. Tweak different JaBeJa configurations to find the smallest edge cuts for the given graphs.
3. Extra effort: define our acceptance probability function or change the Ja-Be-Ja algorithm (to improve its performance) and evaluate how our changes affect the performance of graph partitioning.

We did not use any dedicated data frameworks. The task is implemented in Java.

2 Data source

To test our implementation, we used the 3elt, add20, and Twitter graphs, that were providing in a GitHub repository.

3 Implementation

3.1 Task 1 - Basic implementation

We have implemented the *sampleAndSwap* method and the *findPartner* method described in the F.Rahimian, et al., JA-BE-JA: A Distributed Algorithm for Balanced Graph Partitioning, SASO2013 paper.

This algorithm supports three possible ways of selecting the candidate set:

1. Local (L): every node considers its directly connected nodes (neighbors) as candidates for color exchange;

2. Random (R): every node selects a uniform random sample of the nodes in the graph;

3. Hybrid (H): in this policy first the immediate neighbor nodes are selected (i.e., the local policy). If this selection fails to improve the pair-wise utility, the node is given another chance for improvement, by letting it to select nodes from its random sample (i.e., the random policy).

We also checked that a potential energy is not of the same value as the current one. Otherwise, it can swap infinitely.

Outputs for the given graphs are presented below.

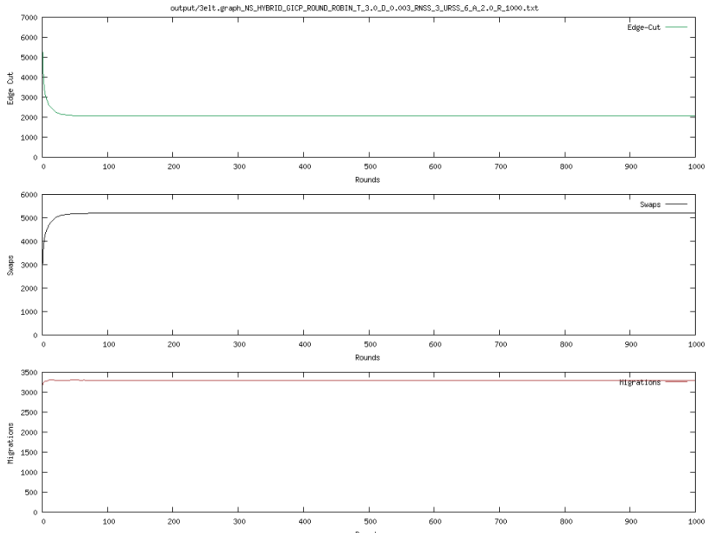

Figure 1: 3elt

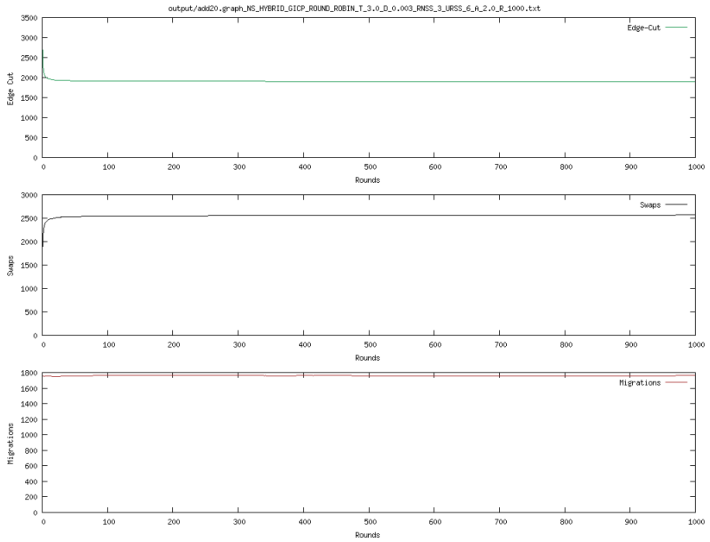

Figure 2: Twitter

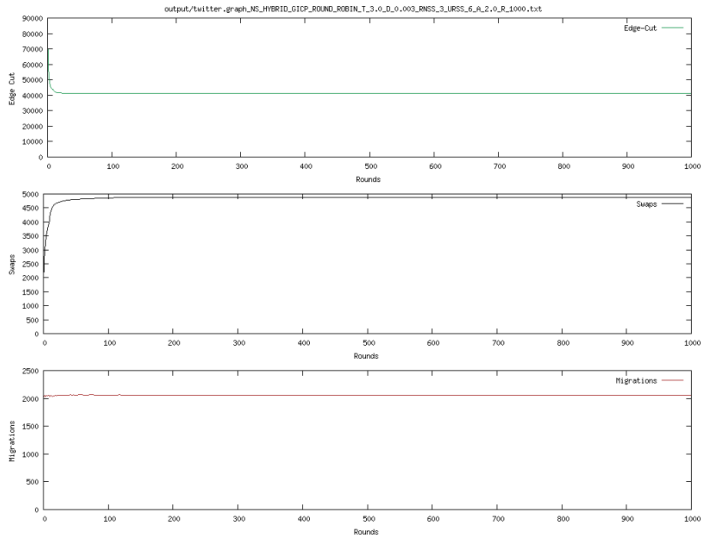

Figure 3: add20

The performance was measured for the *3elt* graph by the following metrics:

1. Edge-cut: the number of interpartition edges;
2. Time to converge;
3. Swaps: the number of swaps that take place between different hosts during run-time (that is, swaps between graph nodes stored on the

same host are not counted).

Changeable parameters are node selection algorithm, temperature and delta.

Set-up	Number of swaps	Time to converge, s	Minimum edge cut
3elt NS H T 5.0 D 0.003	2646819	33	**439**
3elt NS H T 4.0 D 0.003	1352681	36	**470**
3elt NS H T 4.0 D 0.002	1995321	30	**415**
3elt NS H T 3.0 D 0.003	276998	17	**431**
3elt NS H T 3.0 D 0.002	1995321	40	**415**
3elt NS H T 2.0 D 0.003	26742	23	693
3elt NS H T 1.5 D 0.003	9847	3	1207
3elt NS L T 3.0 D 0.003	1992622	9	3059
3elt NS L T 2.0 D 0.003	712436	8	3199
3elt NS L T 1.5 D 0.003	235803	7	3296
3elt NS R T 5.0 D 0.003	1357223	49	**422**
3elt NS R T 4.0 D 0.003	3972697	24	**449**
3elt NS R T 4.0 D 0.002	1978818	21	**405**
3elt NS R T 3.0 D 0.003	297433	8	**406**
3elt NS R T 3.0 D 0.002	435579	14	**428**
3elt NS R T 2.0 D 0.003	28342	4	710
3elt NS R T 1.5 D 0.003	9856	2	1274
Twitter NS H T 3.0 D 0.003	364006	285	41256
Twitter NS H T 3.0 D 0.002	549032	322	41198
Twitter NS R T 3.0 D 0.003	806301	148	**41177**
Twitter NS R T 3.0 D 0.002	1192300	177	41194
Add20 NS H T 3.0 D 0.003	248788	17	1440
Add20 NS H T 3.0 D 0.002	359466	17	**1426**
Add20 NS R T 3.0 D 0.003	249312	9	1453
Add20 NS R T 3.0 D 0.002	369598	10	1452

Table 1: Algorithm performance

As we can conclude from the given table, the minimal edge cut can be achieved with the *Random* approach, *Temperature = 4.0* and Delta = 0.002.

3.2 Simulated Annealing

Simulated annealing is a method for finding a good (not necessarily perfect) solution to an optimization problem. Simulated annealing's strength is that it avoids getting caught at local maxima - solutions that are better than any others nearby, but aren't the very best.

First, we updated our algorithm with the suggestions from the article and tested this changes.

Set-up	Number of swaps	Time to converge, s	Minimum edge cut
3elt NS H T 1.0 D 0.003	4877	3	2041
3elt NS H T 0.8 D 0.003	4917	4	2004
3elt NS H T 0.8 D 0.002	4692	3	1965
3elt NS H T 0.4 D 0.003	4709	4	2021
3elt NS H T 0.4 D 0.002	4792	4	1997
3elt NS H T 0.2 D 0.003	4424	1	2195
3elt NS H T 0.2 D 0.002	4520	2	2110
3elt NS R T 1.0 D 0.003	4732	2	2012
3elt NS R T 1.0 D 0.002	4699	2	2025
3elt NS R T 0.8 D 0.003	4881	3	1914
3elt NS R T 0.8 D 0.002	4774	3	2138
3elt NS R T 0.6 D 0.003	4836	3	2040
3elt NS R T 0.6 D 0.002	4705	1	1876
3elt NS R T 0.4 D 0.003	3279	3	1869
3elt NS R T 0.4 D 0.002	4997	4	**1683**
3elt NS R T 0.2 D 0.003	4985	6	1730
Twitter NS H T 0.8 D 0.003	4805	143	41201
Twitter NS H T 0.8 D 0.002	4959	247	**41138**
Twitter NS R T 0.8 D 0.003	4303	93	41482
Twitter NS R T 0.8 D 0.002	4143	54	41495
Add20 NS H T 0.8 D 0.003	2059	3	1906
Add20 NS H T 0.8 D 0.002	2033	17	1875
Add20 NS R T 0.8 D 0.003	2084	9	**1806**
Add20 NS R T 0.8 D 0.002	2152	12	1862

Table 2: Algorithm performance with simulated annealing

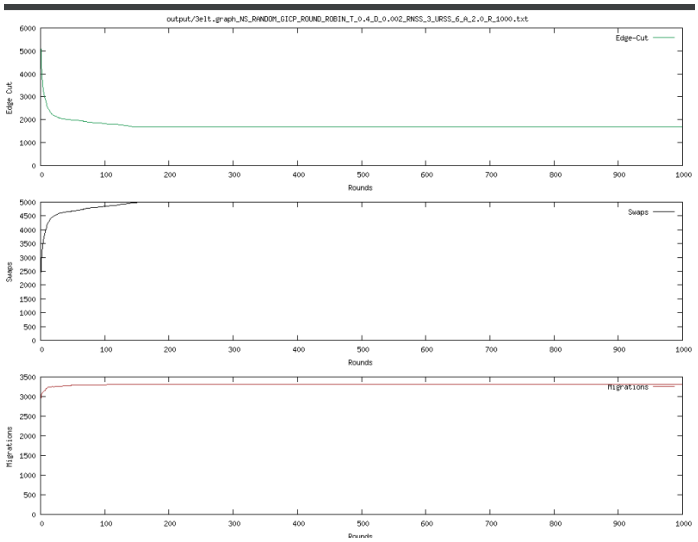

Figure 4: 3elt NS R T 0.4 D 0.002

Next, we added a temperate restart at the 400 round.

Set-up	Number of swaps	Time to converge, s	Minimum edge cut
3elt NS H T 0.8 D 0.003	4518	3	2284
3elt NS H T 0.8 D 0.002	4639	3	2059
3elt NS H T 0.4 D 0.003	4452	3	2215
3elt NS H T 0.4 D 0.002	4763	3	1991
3elt NS H T 0.2 D 0.003	4630	2	2034
3elt NS R T 0.8 D 0.002	4732	2	2205
3elt NS R T 0.6 D 0.003	4867	73	1934
3elt NS R T 0.6 D 0.002	5267	6	**1550**
3elt NS R T 0.4 D 0.003	4710	3	2094
3elt NS R T 0.2 D 0.8	4928	1	1803
3elt NS R T 0.2 D 0.003	4936	3	1789
Twitter NS H T 0.8 D 0.003	4810	111	41225
Twitter NS H T 0.8 D 0.002	4732	172	41180
Twitter NS R T 0.8 D 0.003	4909	69	**41164**
Twitter NS R T 0.8 D 0.002	4198	126	41523
Add20 NS H T 0.8 D 0.003	2014	6	1904
Add20 NS H T 0.8 D 0.002	2013	17	**1879**
Add20 NS R T 0.8 D 0.003	2041	11	1881
Add20 NS R T 0.8 D 0.002	2066	3	**1887**

Table 3: Algorithm performance with a temperature restart

We can see that the temperature restart lowered the number of edge-cuts

but increased the convergence time.

3.3 Extra effort - Algorithm improvements

We are proposing a bit different acceptance probability formula, which can give a better algorithm performance.

$$e^{\frac{(\frac{1}{c_{old}} - \frac{1}{c_{new}})}{T}}. \quad (1)$$

Also, since we use smaller start T for the simulation annealing, we propose a more frequent temperature restart. Moreover, we restart with temperature, calculated by the following expression:

$$T_{restart} = T_{initial} * \delta. \quad (2)$$

Set-up	Number of swaps	Time to converge, s	Minimum edge cut
3elt NS H T 0.8 D 0.9	336138	1	1559
3elt NS H T 0.8 D 0.8	155131	1	1779
3elt NS H T 0.4 D 0.9	267736	1	1451
3elt NS H T 0.4 D 0.8	122615	1	1625
3elt NS H T 0.2 D 0.9	157552	1	**1308**
3elt NS R T 0.8 D 0.9	385442	1	1357
3elt NS R T 0.8 D 0.8	178697	1	1472
3elt NS R T 0.4 D 0.9	268457	1	1565
3elt NS R T 0.4 D 0.8	122126	1	1684
3elt NS R T 0.2 D 0.9	157682	1	1564
Add20 NS H T 0.8 D 0.9	42144	5	1524
Add20 NS H T 0.8 D 0.8	42324	6	1351
Add20 NS R T 0.8 D 0.9	83292	5	**1223**
Add20 NS R T 0.8 D 0.8	40366	5	1533

Table 4: Algorithm performance with a temperature restart and improved calculations

Note: time for conversion is counted for a restart cycle, if any.

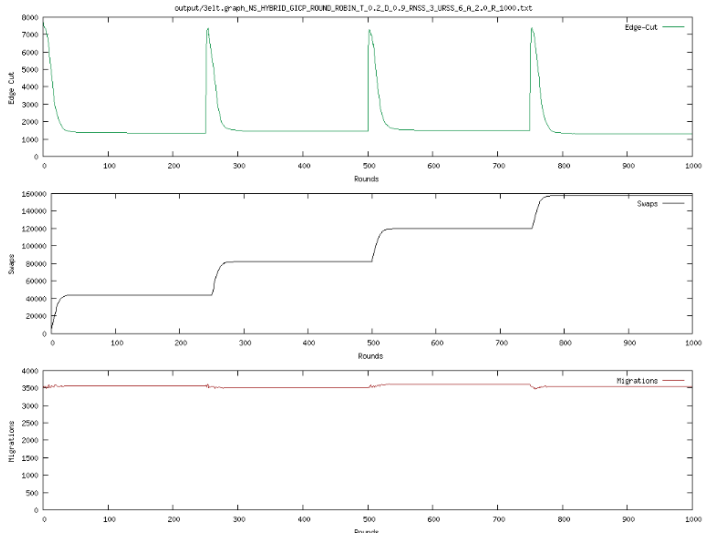

Figure 5: 3elt NS H T 0.2 D 0.9

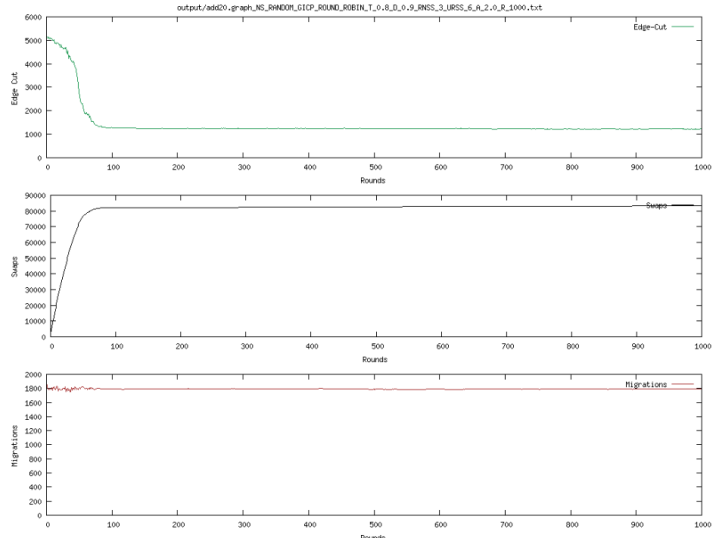

Figure 6: add20 NS R T 0.8 D 0.9

4 Conclusions

In this homework assignment, we got familiar with a Ja-Be-Ja algorithm. We also implemented 3 versions of probability calculations and compared their performance on 3 different graphs.