Data Mining Homework 5

Ja-Be-Ja Algorithm

by

Laura Puccioni
Beatrice Insalata

Kungliga Tekniska Högskolan (KTH)

1 Introduction

In distributed computing, the optimization of graph partitioning holds paramount importance for efficiently processing and analyzing vast datasets. Graph partitioning, the division of a graph into smaller subgraphs while maintaining interconnectivity, is crucial for various applications ranging from social networks and recommendation systems to scientific simulations and network analysis. Addressing this challenge requires sophisticated algorithms that can effectively distribute and balance the graph across multiple computational nodes.

This report focuses specifically on the utilization of gossip-based peer-to-peer techniques. Gossip-based algorithms offer promising avenues for scalable and decentralized graph partitioning, enabling nodes to share information in a peer-to-peer fashion without relying on a centralized coordinator. The key algorithms under scrutiny here is Ja-Be-Ja, as detailed in the paper "A Distributed Algorithm for Balanced Graph Partitioning" (Rahimian et al., 2013).

The Ja-Be-Ja algorithm introduces an ingenious approach to partitioning large graphs in a decentralized manner. Its methodology involves the random allocation of classes (or colors) to individual nodes within the graph, followed by the calculation of local energy. The local energy metric essentially represents the degree of disorder or entropy within the node connections and their associated subset. The primary objective of the algorithm revolves around minimizing this local energy, thereby achieving a balanced partitioning of the graph. To accomplish this, the algorithm initiates a comparison process among nodes and their connected partners. This comparison involves assessing the energy associated with a node's current color assignment and the potential energy when exchanging colors with its connected partner. If this color swap leads to a reduction in energy, signifying a decrease in entropy, it indicates an improved balance between the nodes. Consequently, the algorithm implements the color exchange between the node and its partner to achieve a more balanced graph partition.

2 Task 1

To implement the Ja-Be-Ja algorithm, the first task to accomplish is modifying two methods of the Ja-Be-Ja class in the code given: sampleAndSwap() and findPartner(). The implementation follows closely the pseudo-code provided in the original paper. To avoid becoming stuck in a local optimum while executing the algorithm, Ja-Be-Ja uses the Simulated Annealing (SA) technique, which consists in introducing a temperature (T) and decrease it over time. The updated decision criterion is: $(d_p(\pi_q)^\alpha + d_q(\pi_p)^\alpha) \times T > d_p(\pi_p)^\alpha + d_q(\pi_q)^\alpha$. The annealing policy involved in this task is linear-decreasing over time as described in the paper.

The outcomes of the Ja-Be-Ja have been tested on the 3elt, add20, and Twitter graphs, and can be seen below:

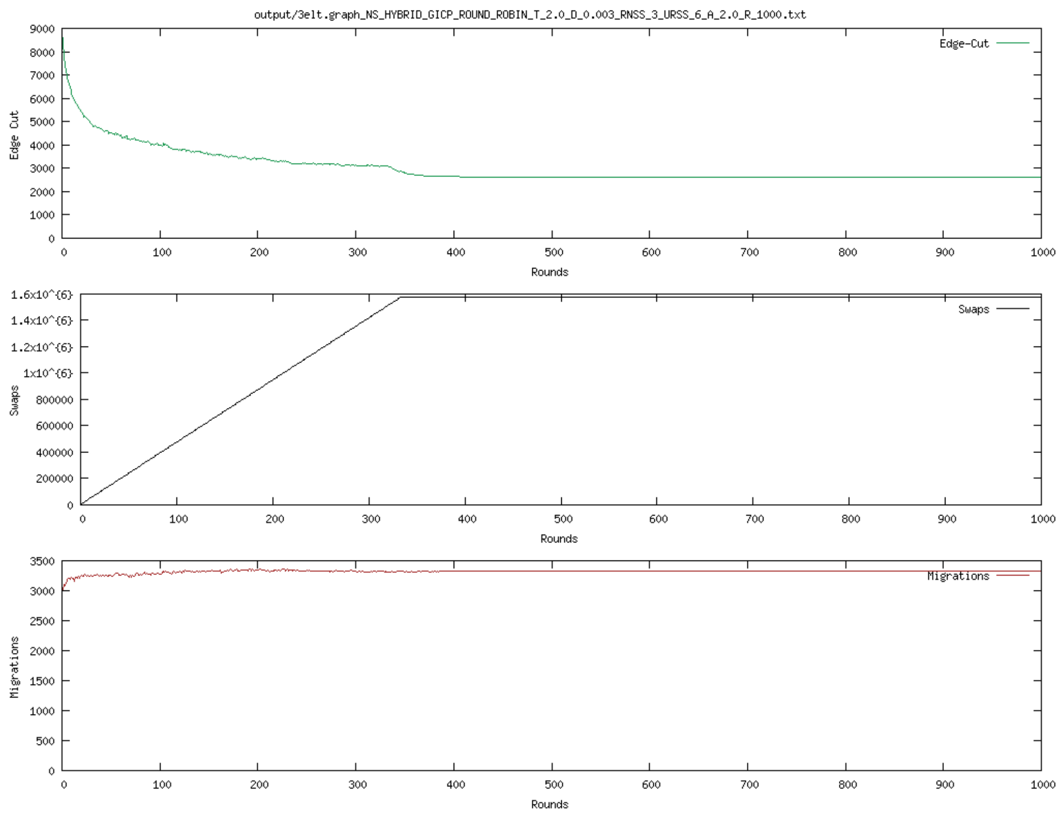


Figure 1: 3elt graph with standard Ja-Be-Ja

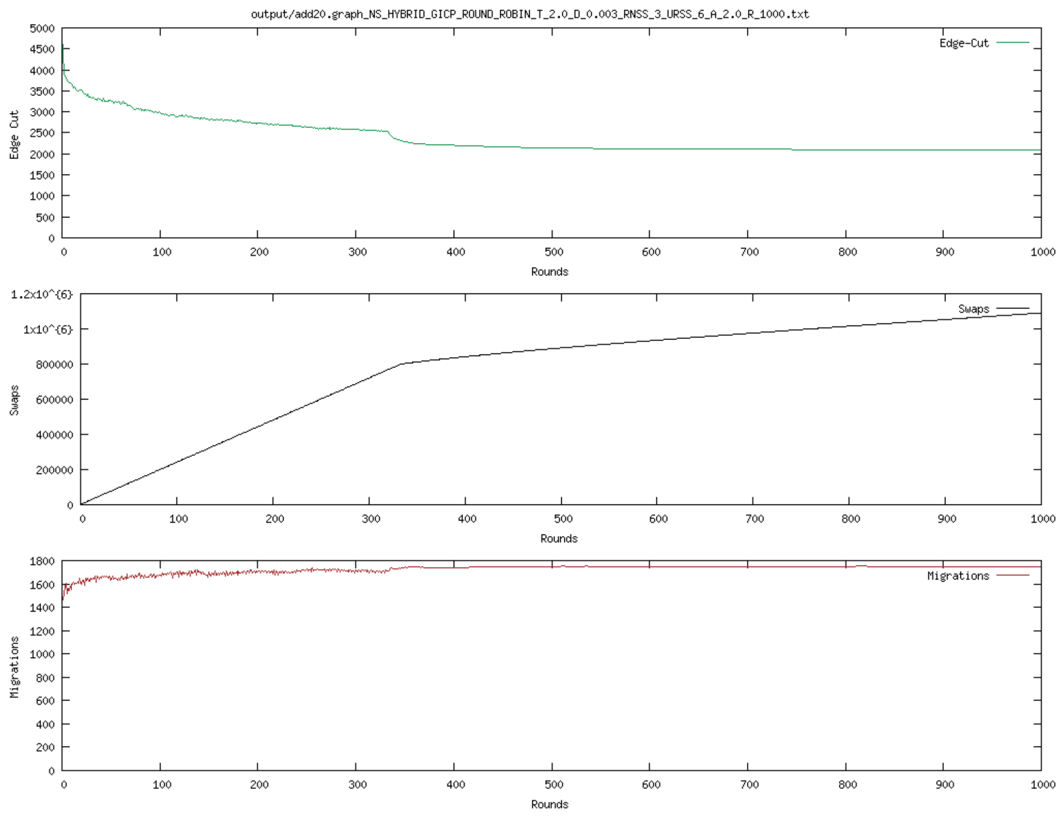


Figure 2: add20 graph with standard Ja-Be-Ja

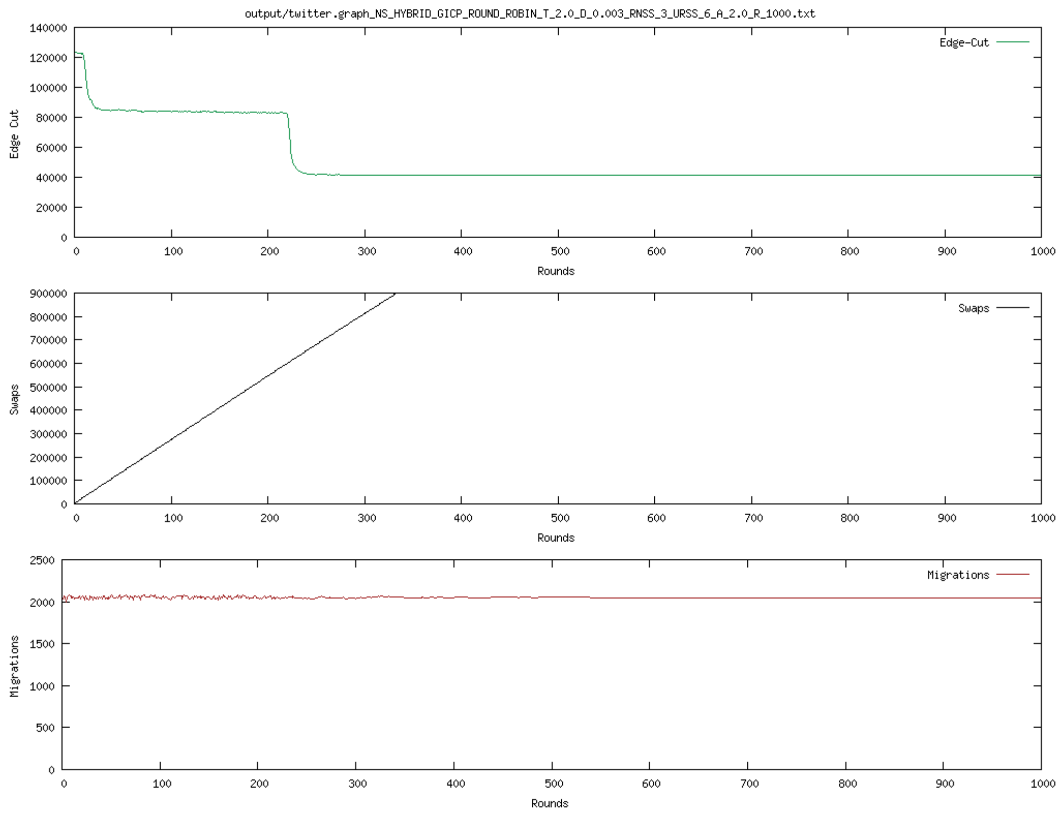


Figure 3: Twitter graph with standard Ja-Be-Ja

As can be seen, the convergence of the partitioning algorithm is dictated by the simulated annealing, which in this case updates the temperature linearly. This can be clearly seen in the graph representing the number of swaps.

The following table summarizes the results obtained in this first task.

Data	T	delta	rounds	edge-cut	swaps	migrations
3elt	2	0.003	1000	2604	1580209	3328
add20	2	0.003	1000	2095	1090263	1751
twitter	2	0.003	1000	41156	899515	2049

Table 1: Outcomes linear approach

3 Task 2

In this task, the focus is on analyzing the impact of various parameters and configurations on the performance of the Ja-Be-Ja algorithm, specifically concerning simulated annealing. At present, the algorithm employs a linear temperature decrease mechanism and multiplies the temperature by the cost function during its iterations. The objective is to explore and implement a different simulated annealing mechanism, observing its influence on the convergence rate of Ja-Be-Ja. By adjusting parameters related to simulated annealing, such as the cooling schedule and acceptance probability function, we aim to understand how these modifications affect the algorithm's convergence towards optimal edge cuts.

The new algorithm proposed consists in an exponential approach for the simulated annealing, with the main condition for switching to the new value becoming $\exp\left(\frac{\text{newValue}-\text{oldValue}}{T}\right) >$ randomValue in (0,1). For this method, the maximum initial temperature is 1 and it will be decreased by multiplying it by delta after each round. For this task we chose delta = 0.9.

Furthermore, 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 the algorithm from getting stuck in local minima, accepting more swaps and achieving lower edge cuts.

Results are shown below:

Data	T	delta	rounds	edge-cut	swaps	migrations
3elt	1	0.9	1000	1880	30283	3342
add20	1	0.9	1000	1992	667840	1768
twitter	1	0.9	1000	41656	6041	2026

Table 2: Outcomes exponential approach

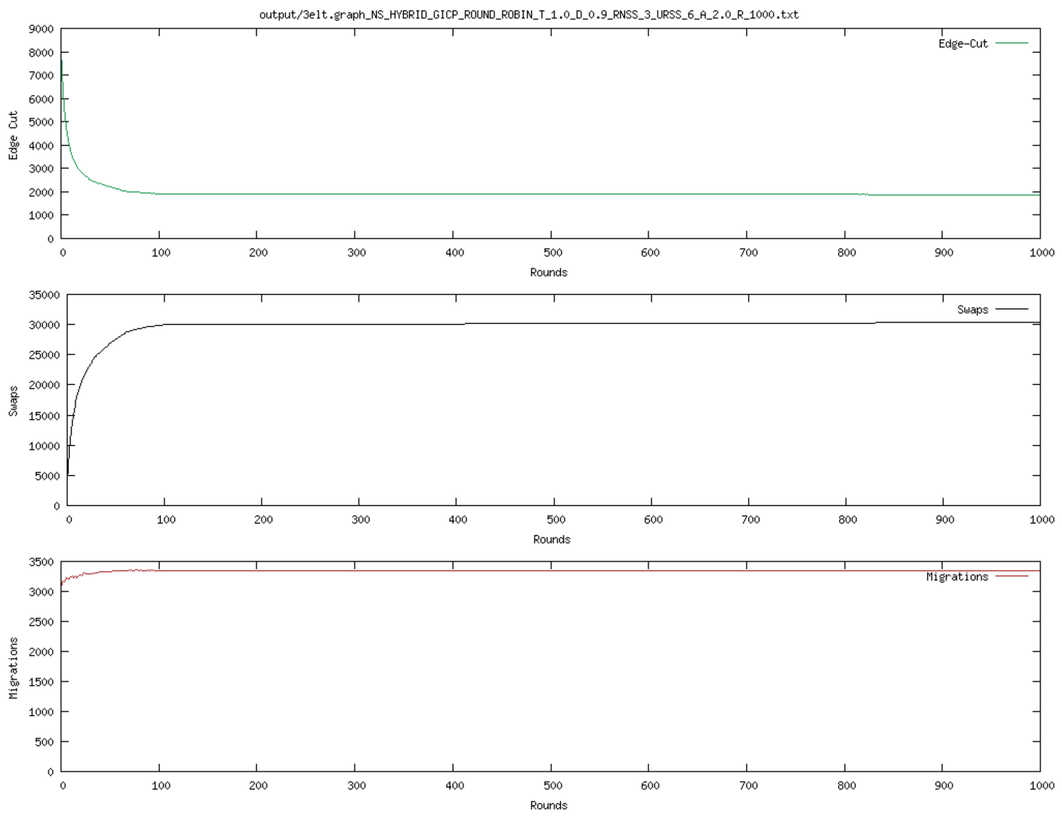


Figure 4: 3elt graph with exponential annealing

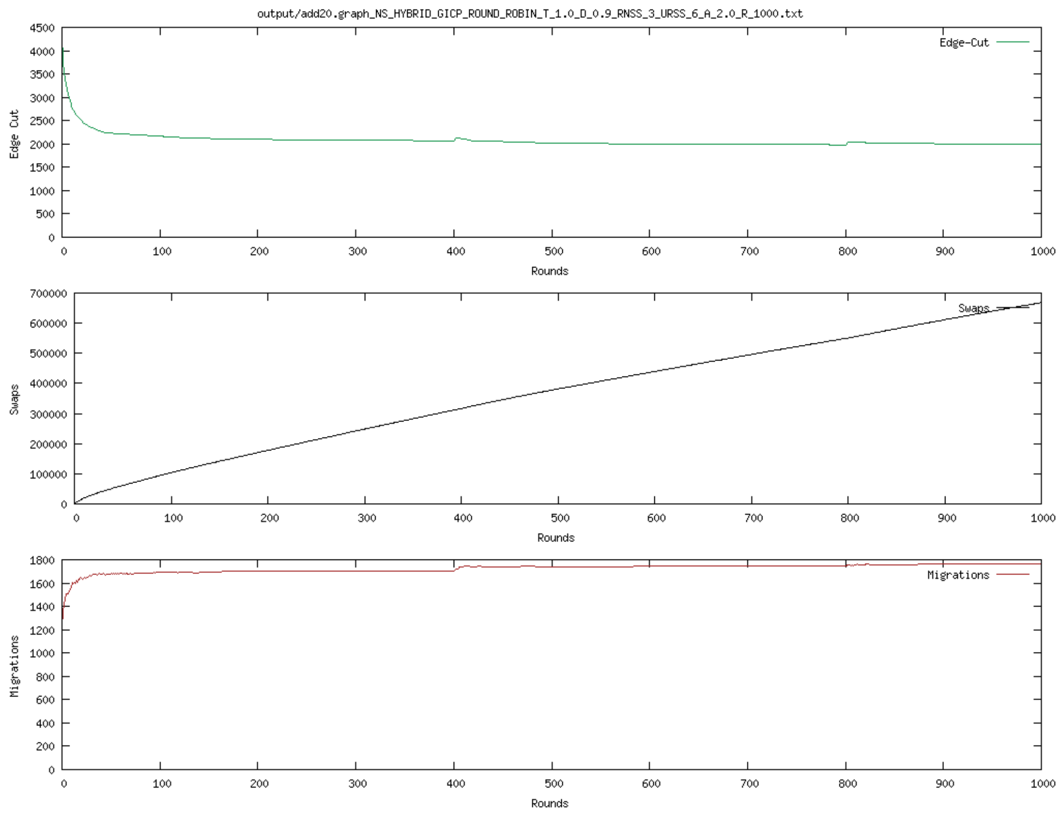


Figure 5: add20 graph with exponential annealing

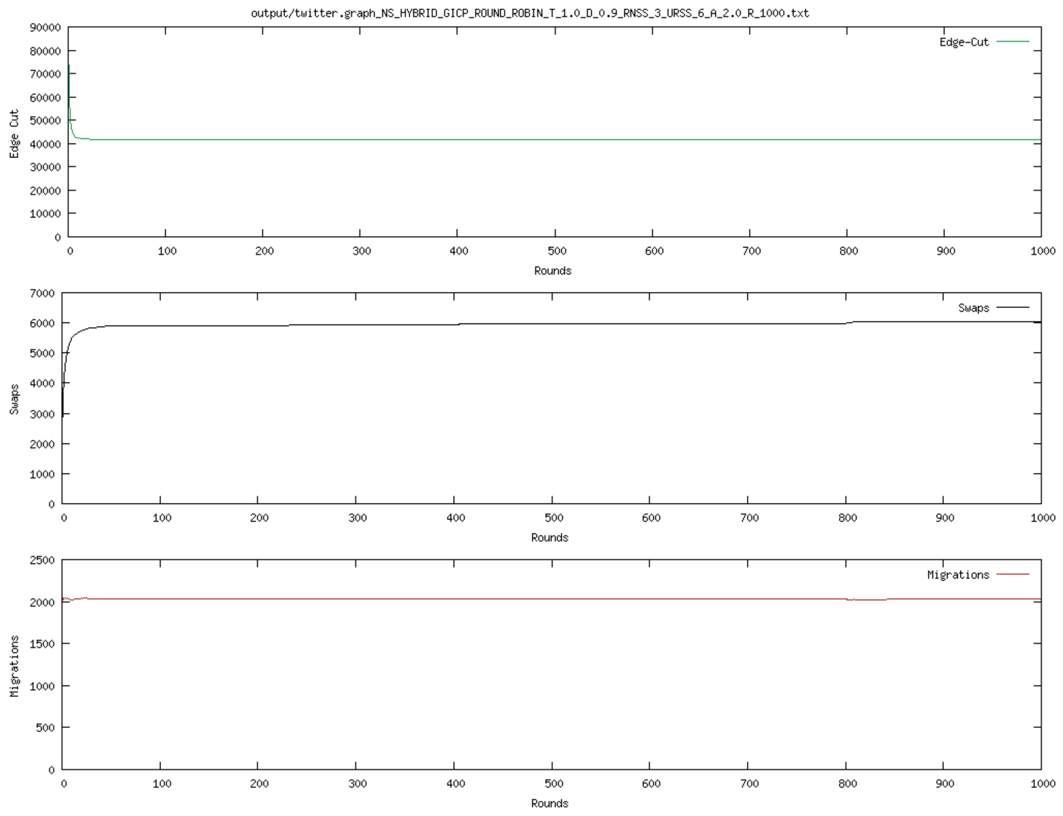


Figure 6: Twitter graph with exponential annealing

Although the add20 and twitter graphs exhibit relatively minor differences, across all three cases, fewer swaps were required to initiate convergence. Moreover, the reduction in edge cut occurred at a significantly faster rate with the exponential approach compared to the linear one.

3.1 Experiments varying the parameters

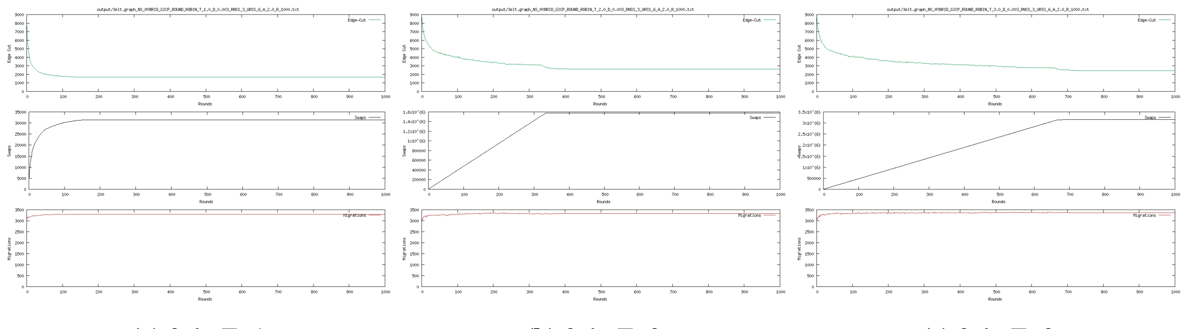
We investigate how the different parameters, T, delta and alpha, affect the results for each of the graphs. We start experimenting with the parameter T, using the linear cooling function and keeping alpha = 2, delta = 0.003 and rounds = 1000.

Data	**T**	**edge-cut**	**swaps**	**migrations**
3elt	1	**1673**	31363	3287
3elt	2	2604	1580209	3328
3elt	3	1784	30657	3326
add20	1	**1853**	582899	1697
add20	2	2095	1090263	1751
add20	3	2100	1787527	1779
twitter	1	41675	6087	2032
twitter	2	**41156**	899515	2049
twitter	3	41224	1803820	2052

Table 3: Outcomes linear approach varying T

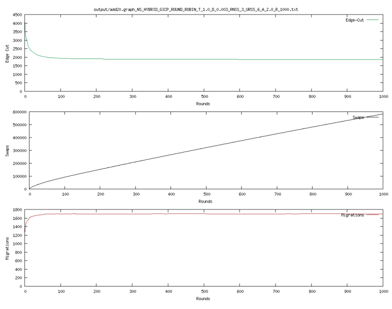
The temperature parameter significantly influences the nature of swaps made by the algorithm. For instance, a temperature (T) greater than 1 permits the inclusion of "bad swaps" to enhance the likelihood of better swaps in subsequent iterations. Observations reveal that disallowing such "bad swaps" (T=1) results in improved edge-cuts for the 3elt and add20 graphs. However, in the case of the twitter graph, the most favorable outcome is achieved at T=2. As expected, the number of swaps is impacted by T: with T=1 the number of swaps strongly decreases.

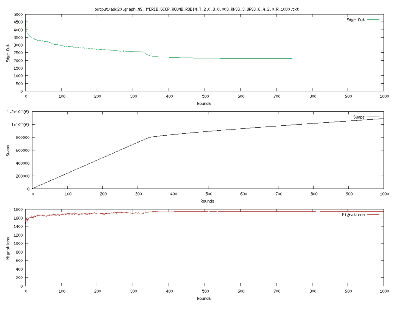
The results can be graphically seen in the following plots.



(a) 3elt, T=1 **(b)** 3elt, T=2 **(c)** 3elt, T=3

Figure 7: 3elt graph with varying T


(a) add20, T=1


(b) add20, T=2

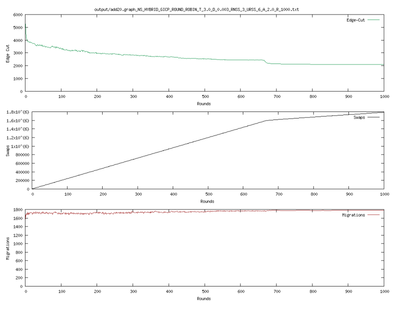
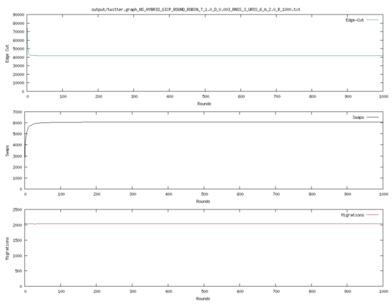
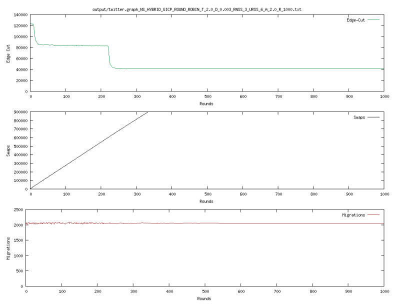

(c) add20, T=3

Figure 8: add20 graph with varying T


(a) twitter, T=1


(b) twitter, T=2

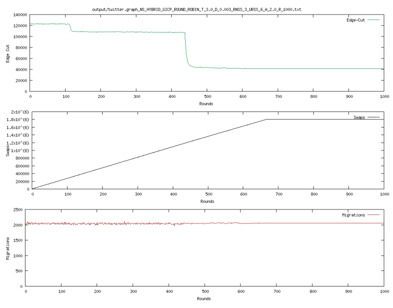
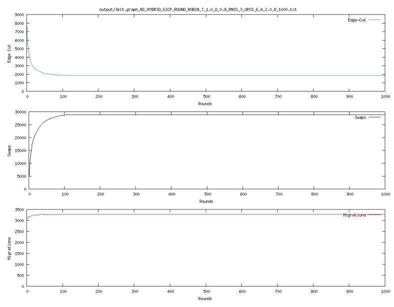

(c) twitter, T=3

Figure 9: twitter graph with varying T

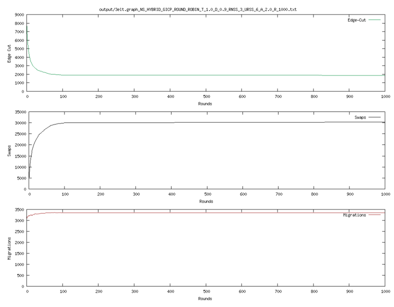
Then we experiment with different values of delta in the exponential annealing approach. Keeping T = 1, rounds = 1000 and alpha = 2, delta was varied.

Data	delta	edge-cut	swaps	migrations
3elt	0.8	1869	28888	3276
3elt	0.9	1880	30283	3342
3elt	0.99	**1752**	32830	3338
add20	0.8	**1929**	630062	1692
add20	0.9	1992	667840	1768
add20	0.99	1977	761654	1789
twitter	0.8	41719	5964	2021
twitter	0.9	**41656**	6041	2026
twitter	0.99	41725	7992	2022

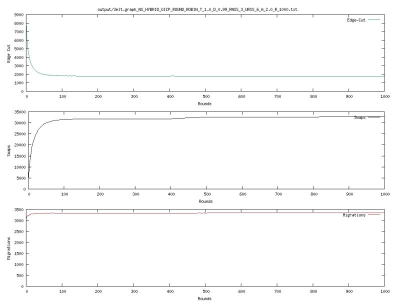
Table 4: Outcomes exponential approach varying delta



(a) 3elt, delta=0.8

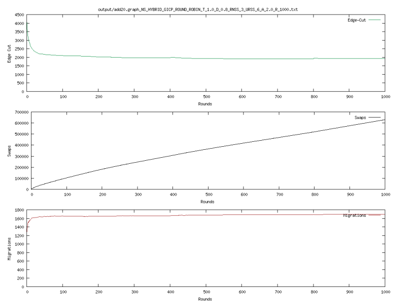


(b) 3elt, delta=0.9

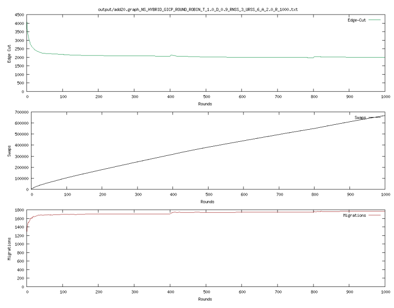


(c) 3elt, delta=0.99

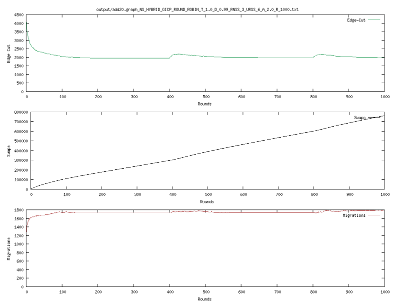
Figure 10: 3elt graph with varying delta



(a) add20, delta=0.8

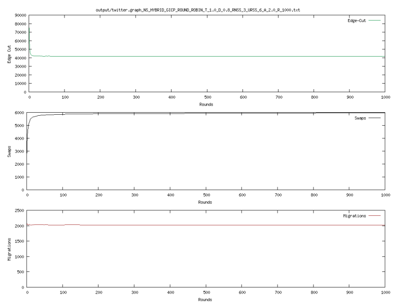


(b) add20, delta=0.9

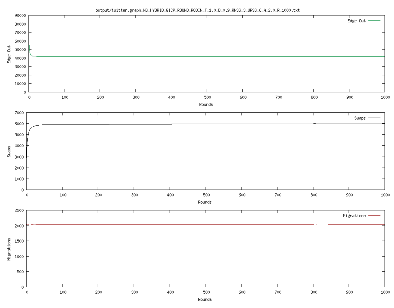


(c) add20, delta=0.99

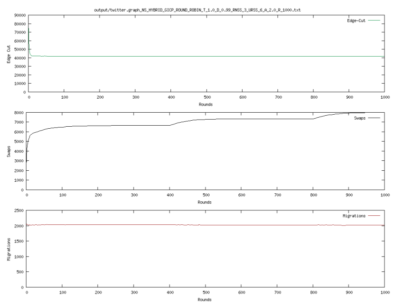
Figure 11: add20 graph with varying delta



(a) twitter, delta=0.8



(b) twitter, delta=0.9



(c) twitter, delta=0.99

Figure 12: twitter graph with varying delta

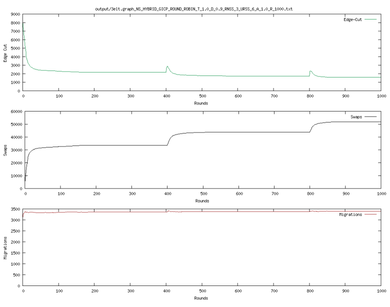
The parameter delta plays a crucial role in determining the rate at which T decreases over successive rounds. A smaller delta leads to a faster cooling rate for T. In the exponential approach, T gets updated each round by multiplying it with delta, and T influences the likelihood of swaps: a lower T corresponds to a higher probability of performing a swap.

Interestingly, in the case of the 3elt graph, with a higher delta the number of swaps is higher while the edge cut lowers. Conversely, for the other two graphs, the trend is slightly different.

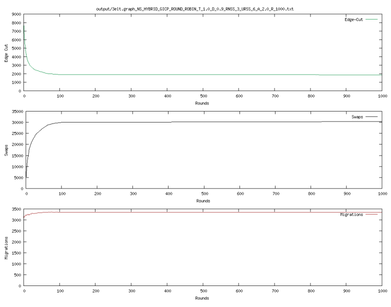
Finally we experimented with the parameter alpha, using the exponential approach and keeping T=1 and delta=0.9.

Data	alpha	edge-cut	swaps	migrations
3elt	1	**1578**	51775	3399
3elt	2	1880	30283	3342
3elt	3	1948	30064	3325
add20	1	**1711**	589749	1778
add20	2	1992	667840	1768
add20	3	1881	416999	1730
twitter	1	**40919**	6228	2030
twitter	2	41656	6041	2026
twitter	3	41887	6097	2025

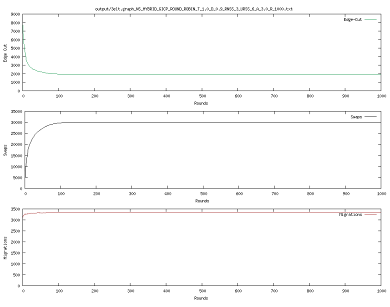
Table 5: Outcomes exponential approach varying alpha



(**a**) 3elt, alpha=1

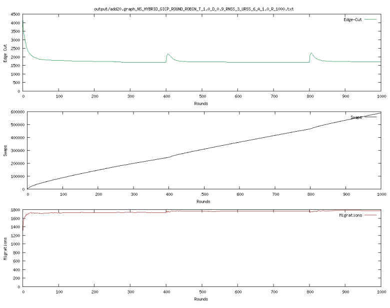


(**b**) 3elt, alpha=2

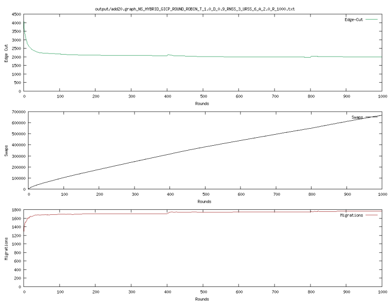


(**c**) 3elt, alpha=3

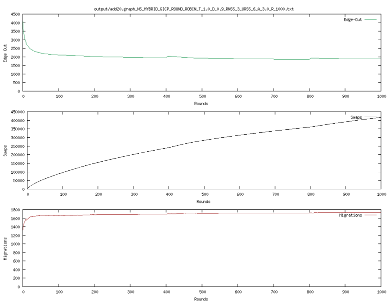
Figure 13: 3elt graph with varying alpha



(**a**) add20, alpha=1

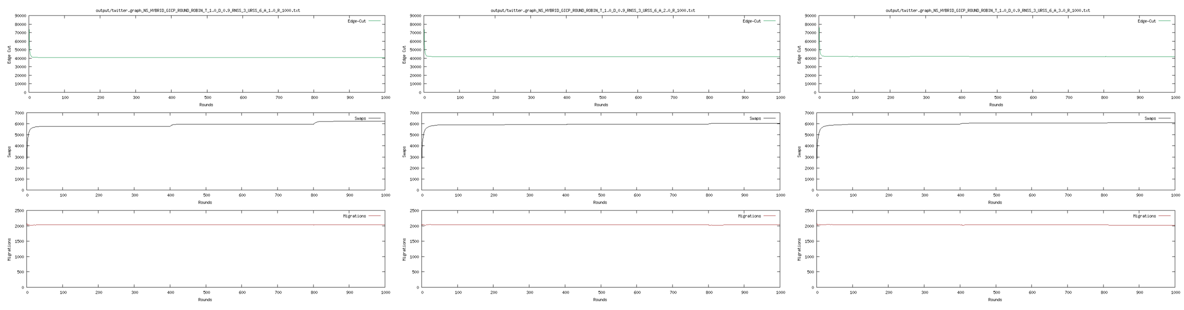


(**b**) add20, alpha=2



(**c**) add20, alpha=3

Figure 14: add20 graph with varying alpha



(a) twitter, alpha=1 **(b)** twitter, alpha=2 **(c)** twitter, alpha=3

Figure 15: twitter graph with varying alpha

The parameter alpha is instrumental in computing the values of newValue and oldValue, denoting the number of same-color nodes connected after and before swaps, respectively. When alpha exceeds 1, the difference between newValue and oldValue increases exponentially, thereby elevating the likelihood of swaps. However, this heightened propensity for swaps may result in a deterioration of the edge-cut value. This observation is evident across all graphs, where employing alpha=3 yields the worst results.

On the other hand, with alpha=1, swaps occur only when new representations denote a significant enhancement (a low difference between newValue and oldValue corresponds to a low probability of a swap).

Moreover, the plots illustrate noticeable spikes around rounds 400 and 800, coinciding with the restart of T. During this phase, when T reverts to its initial value (T=1), the allowance for bad swaps results in increased edge-cut values and swap counts.

4 Optional Task - Custom Acceptance Probability

After careful consideration, we have chosen a custom acceptance probability inspired by the sigmoid function. Our formula, $\frac{1}{1+\exp\left(-\frac{\text{newValue}-\text{oldValue}}{T}\right)}$, has been defined to promote smoother convergence and enhance partitioning stability. We observed that the previous exponential function could lead to abrupt changes in partitioning, which, although aiding faster exploration, also introduced instability and oscillation. By adopting a sigmoid-like function, our goal is to mitigate drastic changes during the partitioning process. This function constrains the acceptance probability within a bounded range, fostering a more gradual exploration of the solution space. This approach aims to strike a balance between effective exploration and maintaining stable partitions, enhancing the overall performance and reliability of our algorithm.

The algorithm was evaluated using alpha = 2, T = 1, delta = 0.9, rounds = 1000.

The application of this function resulted in slightly enhanced outcomes. Tables 6 and 7 showcase the results obtained through the exponential and custom approaches, respectively. A comparative analysis reveals that, particularly for the initial two graphs, a lower edge-cut is achieved using the exponential method. While the edge-cut metric does not exhibit improvement in the case of the twitter graph, the other two metrics demonstrate either comparable or reduced values across all three graphs.

Data	edge-cut	swaps	migrations
3elt	1880	30283	3342
add20	1992	667840	1768
twitter	41668	5956	2024

Table 6: Outcomes exponential approach

Data	edge-cut	swaps	migrations
3elt	1654	34099	3274
add20	1744	558304	1769
twitter	41682	5852	2014

Table 7: Outcomes of custom approach

5 Instructions to run the code

Before running the program, the installation of gnuplot and maven is required. When this is completed, open the terminal in the direectoy that hosts the program and follow these steps:

- execute the command **./compile.sh**
- execute the command **./run.sh -graph ./graphs/[graph_name].graph** ([graph_name] can be 3elt, add20 or twitter depending on which graph you want to use)
- Modify the name of the generated file to "result.txt", then execute the command **./plot.sh output/result.txt**

References

Rahimian, F., Payberah, A. H., Girdzijauskas, S., Jelasity, M., & Haridi, S. (2013). Ja-be-ja: A distributed algorithm for balanced graph partitioning.