HW5 - Graph Partitioning JaBeja

Syed Arif Rahman<sarahman@kth.se>

Group 10

1. Introduction

This report outlines my personal implementation of the JaBeJa algorithm based on the given paper, which incorporates the K-way graph partitioning approach. I utilized this implementation to analyze three specific graph datasets: 3elt, add20, and Twitter. For Task 1, I successfully implemented the JaBeJa algorithm, while Task 2 involved the creation of an exponentially-decreasing simulated annealing policy and the incorporation of a mechanism for restarting the temperature.

2. How to Run the code

To run the program, follow these steps:

1. Download and Unzip the folder:
2. Move to id2222-master folder:
3. Ensure that Maven and Gnuplot are installed. Open the terminal and execute the following commands for the three graph files (3elt, add20, and twitter graphs):

```
>>./compile.sh
>>./run-graph.sh ./graphs/your-desire-graph-file.graph
>>./plot.sh output/your-desire-result-file.txt
```

3. Task1

In the execution of Task 1, the implementation of the JaBeJa algorithm demanded the modification of two crucial methods within the Jabeja class, namely sampleAndSwap() and findPartner(). The integration closely adhered to the pseudo-code outlined in the original paper. The outcomes of our JaBeJa implementation, employing HYBRID node policies follows the linear acceptance formula(lp) = (newD * T > oldD) && (newD > highestBenefit) , are detailed in below figures. Throughout this process, the parameter delta was maintained at 0.003, and default values for the initial temperature (T) at 2 and alpha at 2 were retained as per CLI.java.

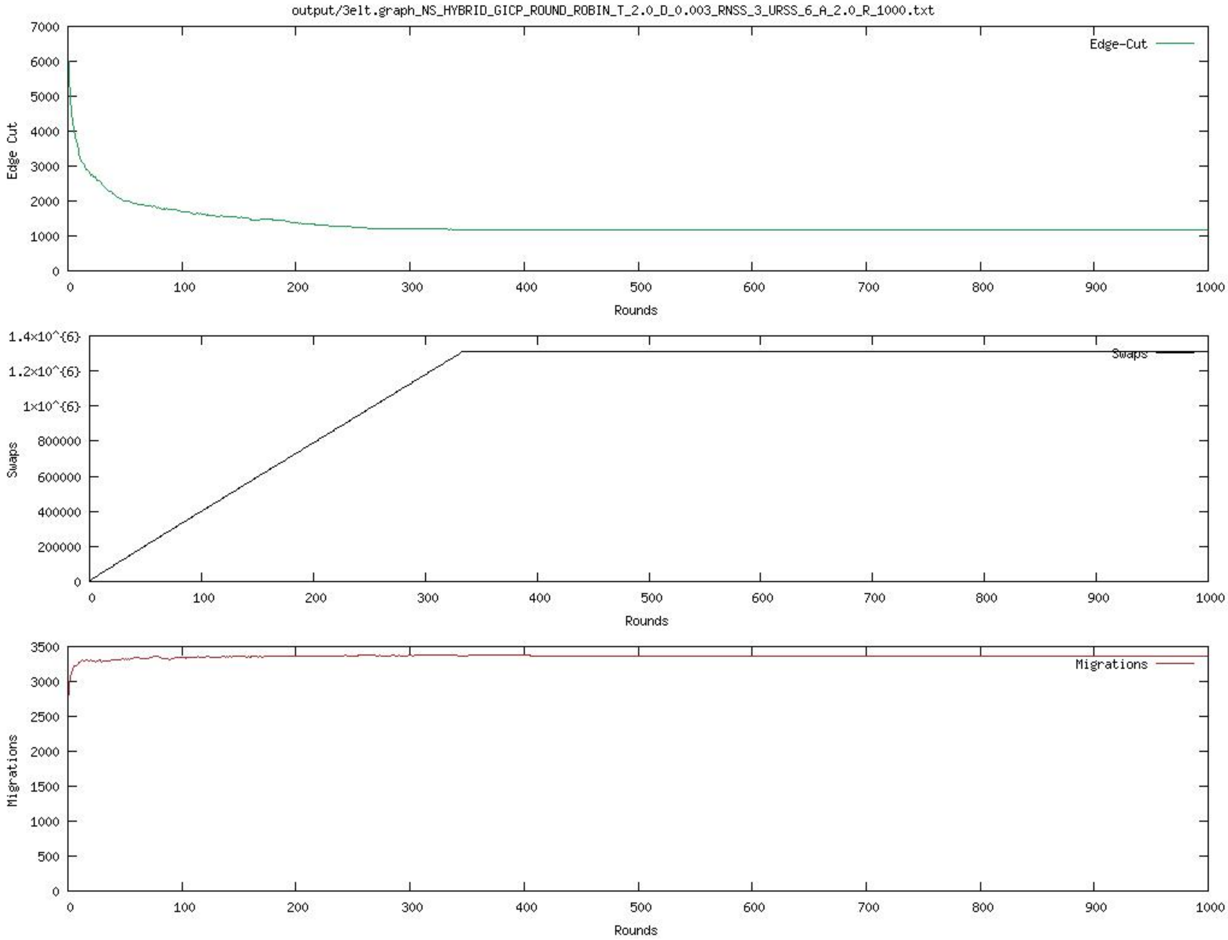


Figure 1: 3elt graph - edge cut: 1163; swaps: 1307755; migrations: 3369; time: 204 seconds;

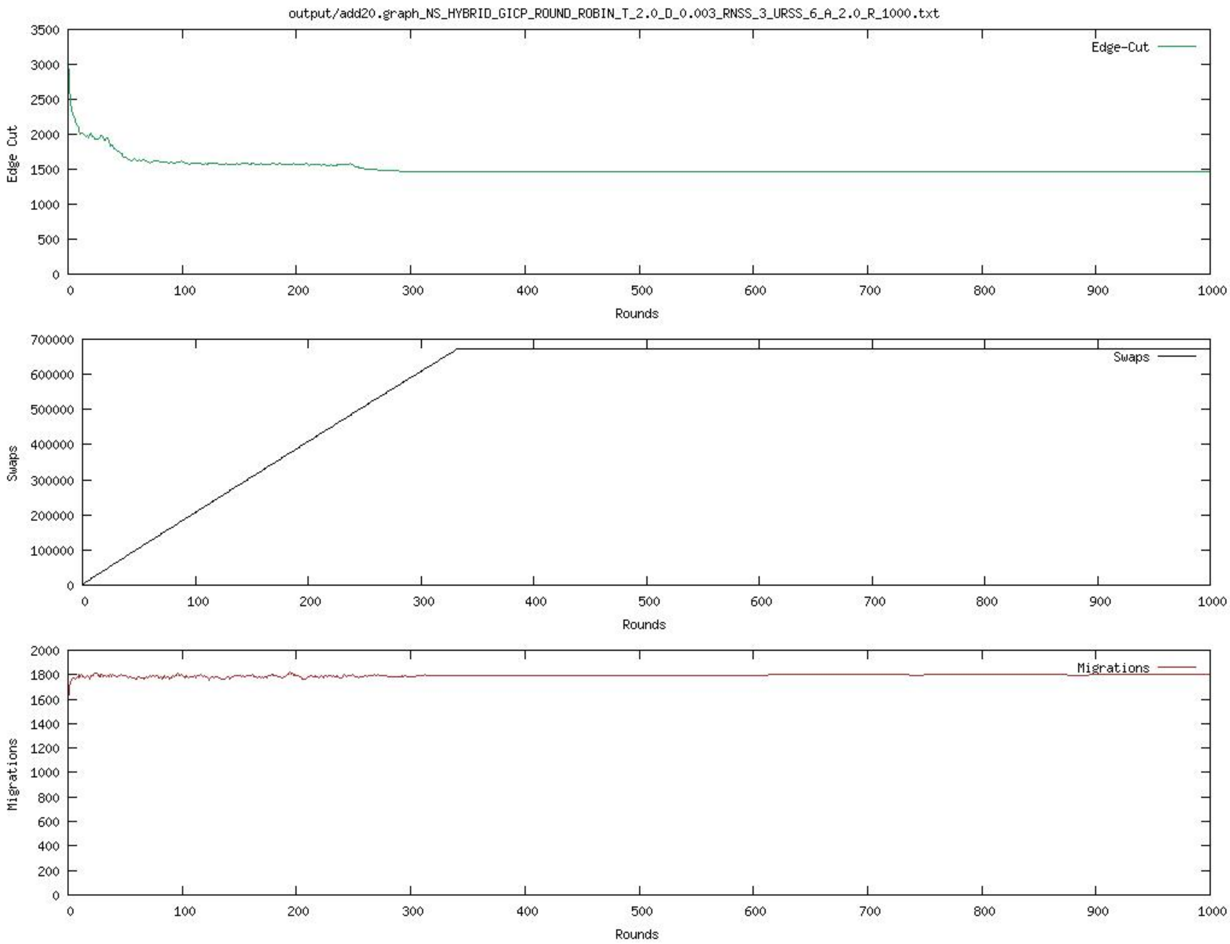


Figure 2: add20 graph - edge cut: 1460; swaps: 673453; migrations: 1799; time: 125 seconds;

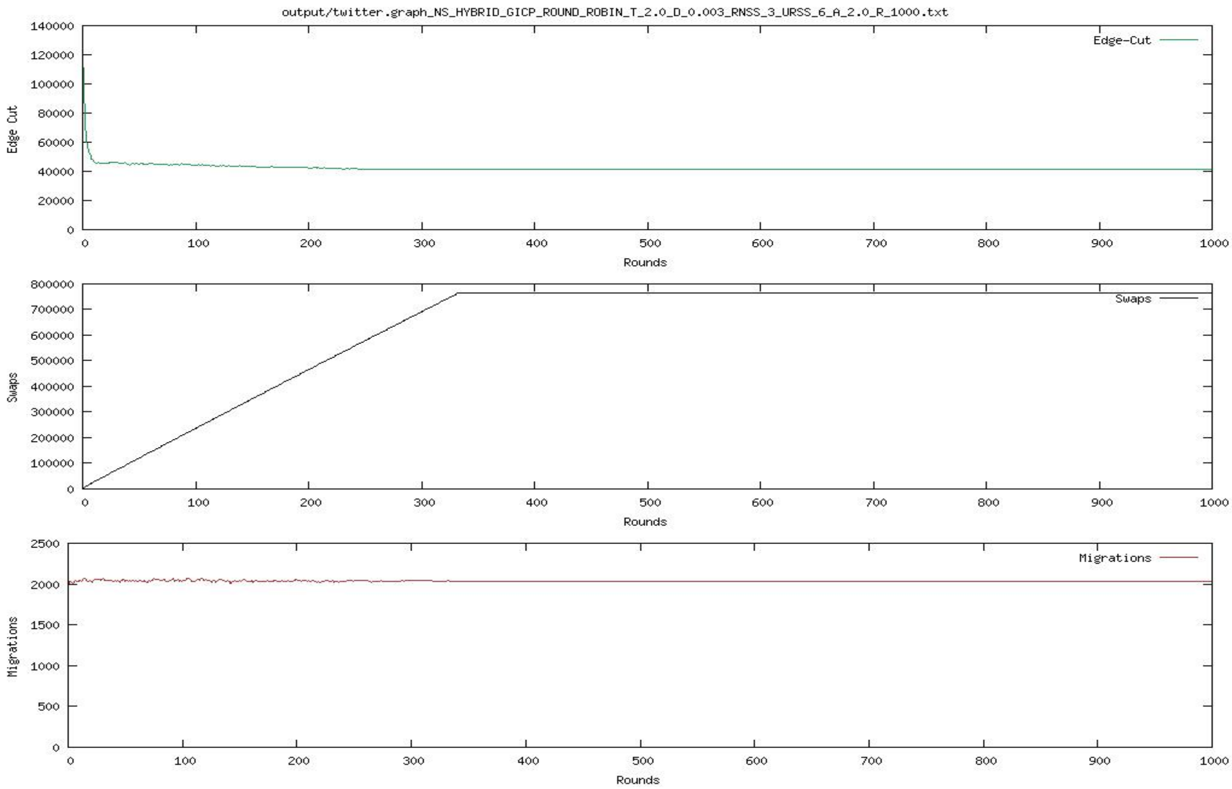


Figure 3: Twitter graph - edge cut: 41176; swaps: 765501; migrations: 2030; time: 820 seconds;

4. Task 2

At first in this task, I implemented an exponentially-decreasing annealing policy. The temperature undergoes exponential reduction with a factor of delta, starting at 1 and continuing until it reaches a minimum value of 0.001. This reduction behavior aligns with my created exponentialCoolDown() method, which accommodates exponential annealing policies. I craete a method named acceptSolution() to calculate the acceptance probability (ap) based on old and new values (oldD and newD) using the formula ap = e^((newD-oldD)/T). Upon analyzing this formula, it becomes evident that the acceptance probability (ap) decreases when the new value is smaller than the old value or when the temperature becomes larger (newD > oldD). Below I present the outcomes for various delta values(0.5 and 0.9) employing the exponential annealing policy.

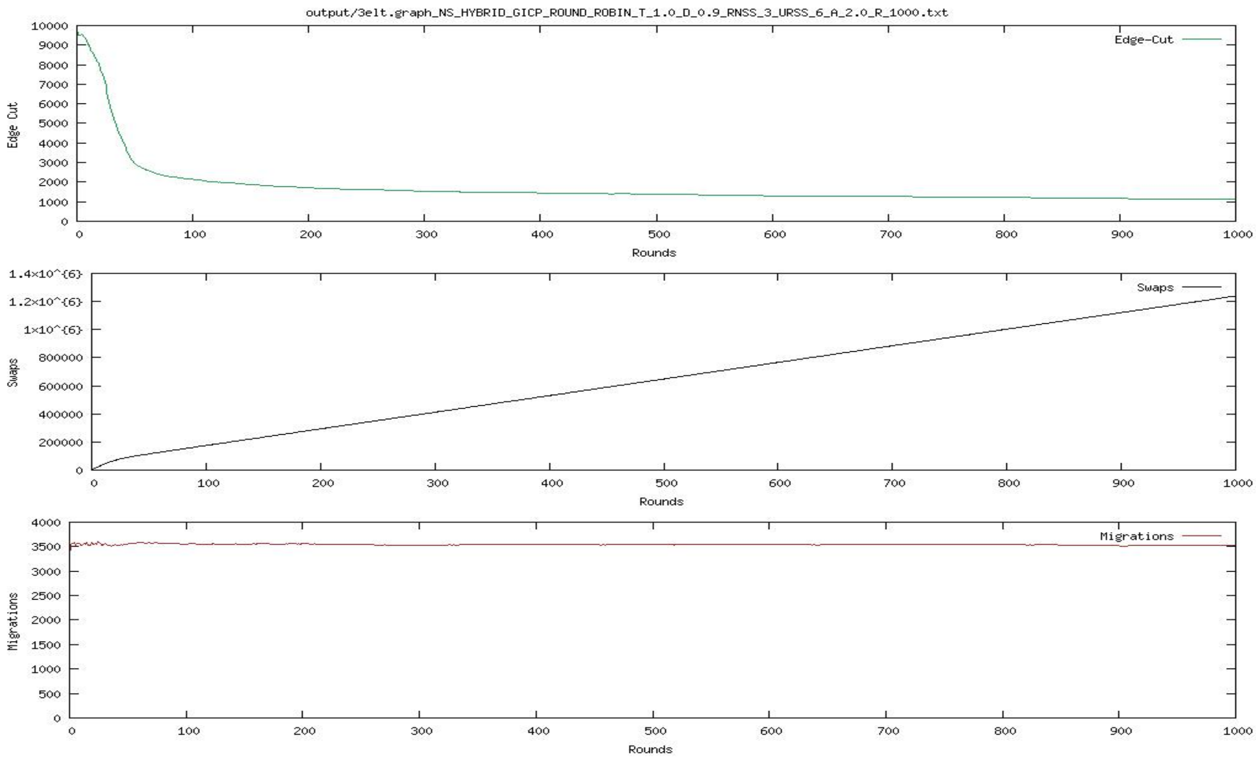


Figure 4: Exponential annealing 3elt (D: 0.9; T: 1; edge-cut: 1117; swap: 1238076; migration: 3525; time elapsed: 81 ms)

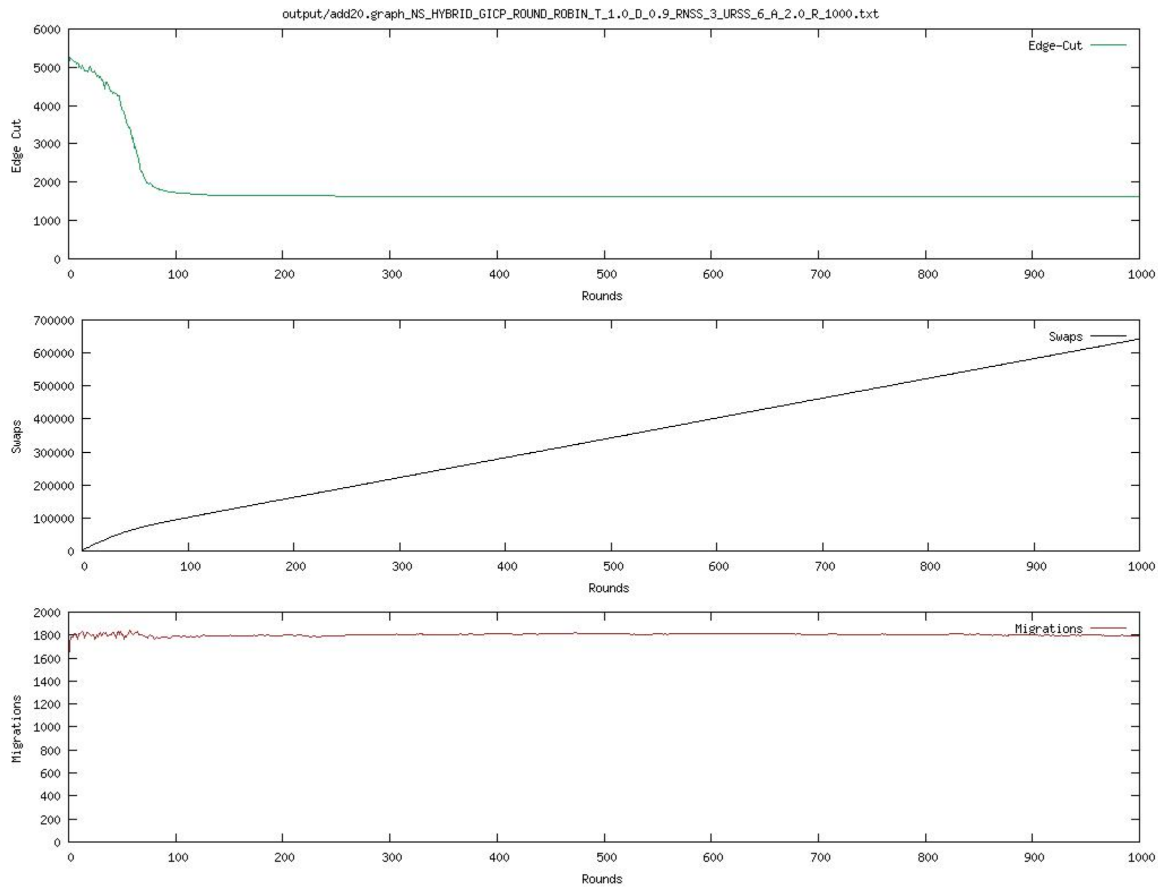


Figure 5: Exponential annealing add20 (D: 0.9; T: 1; edge-cut: 1620; swap: 641891; migration: 1798; time elapsed: 43 seconds)

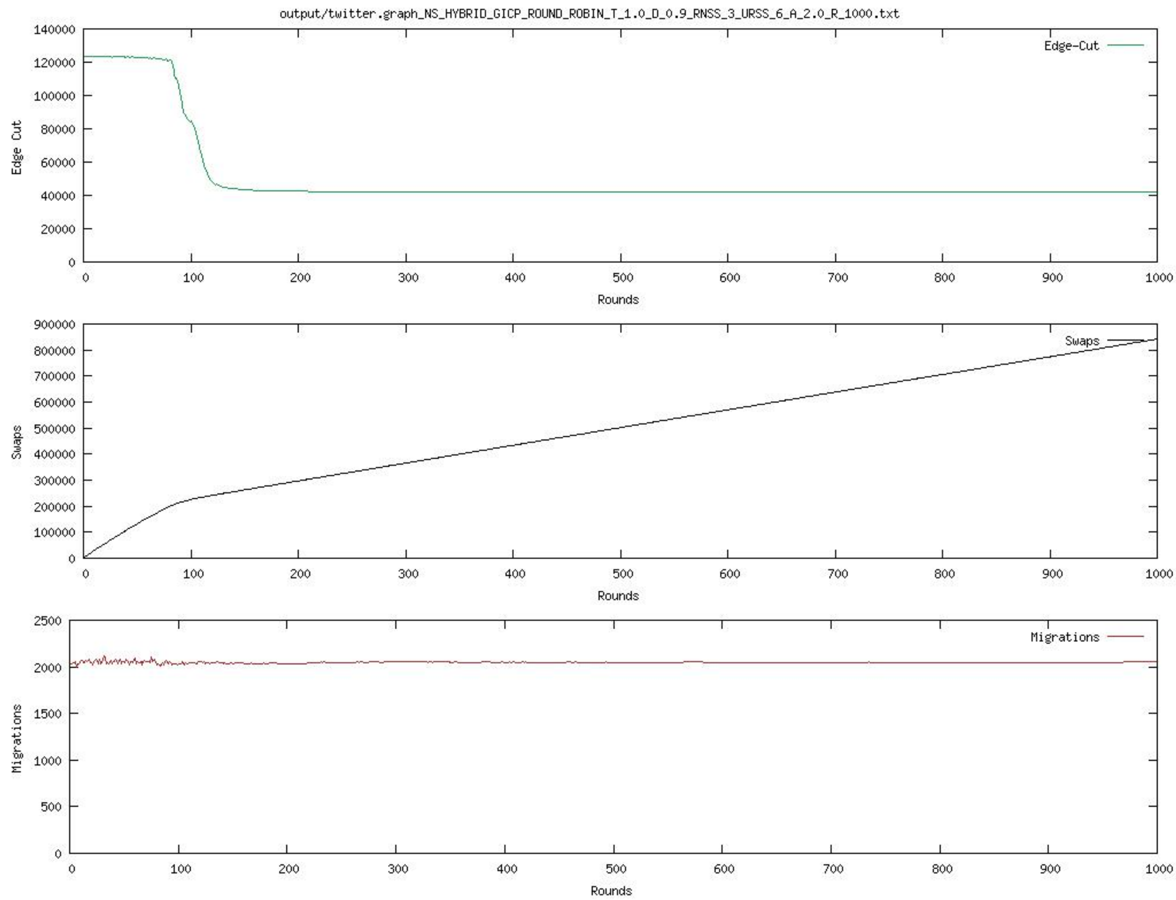


Figure 6: Exponential annealing twitter (D: 0.9; T: 1; edge-cut: 42081; swap: 841840; migration: 2051; time elapsed: 460 seconds)

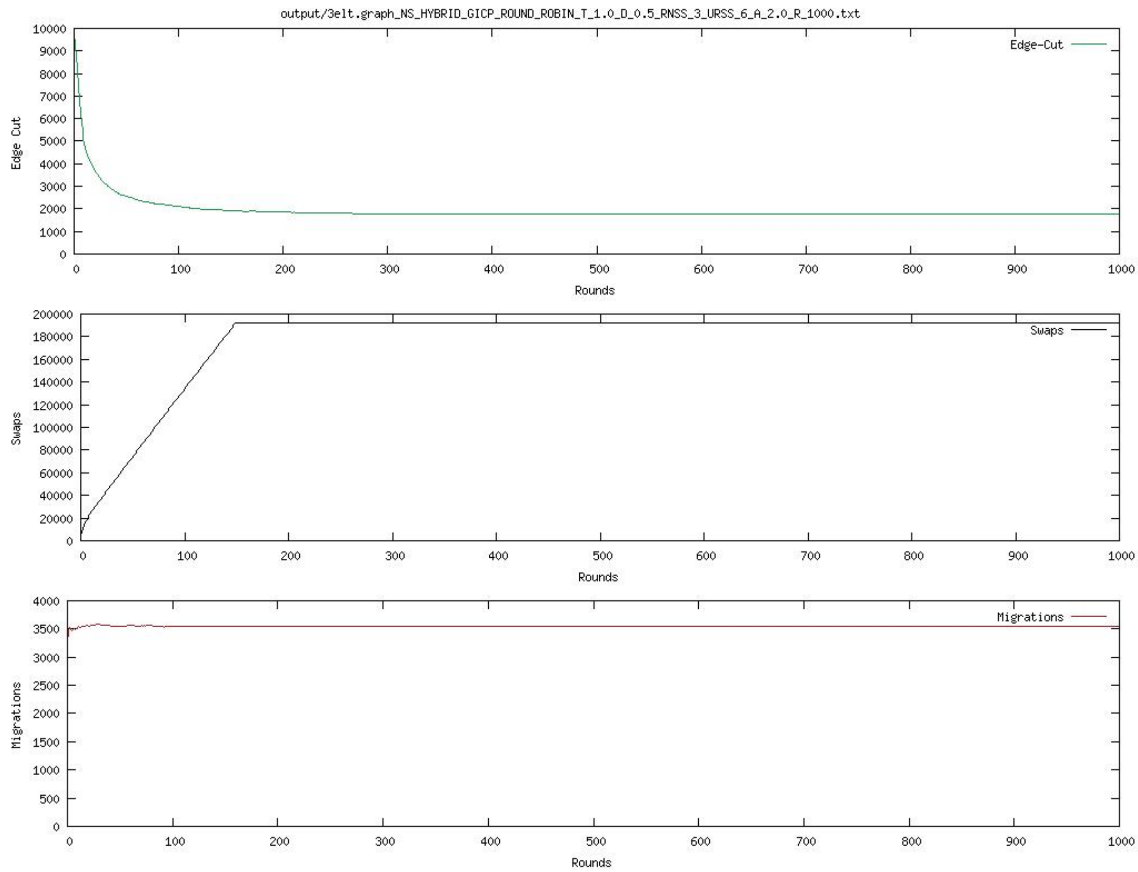


Figure 7: Exponential annealing 3elt (D: 0.5; T: 1; edge-cut: 1753; swap: 192140; migration: 3544; time elapsed: 53 seconds)

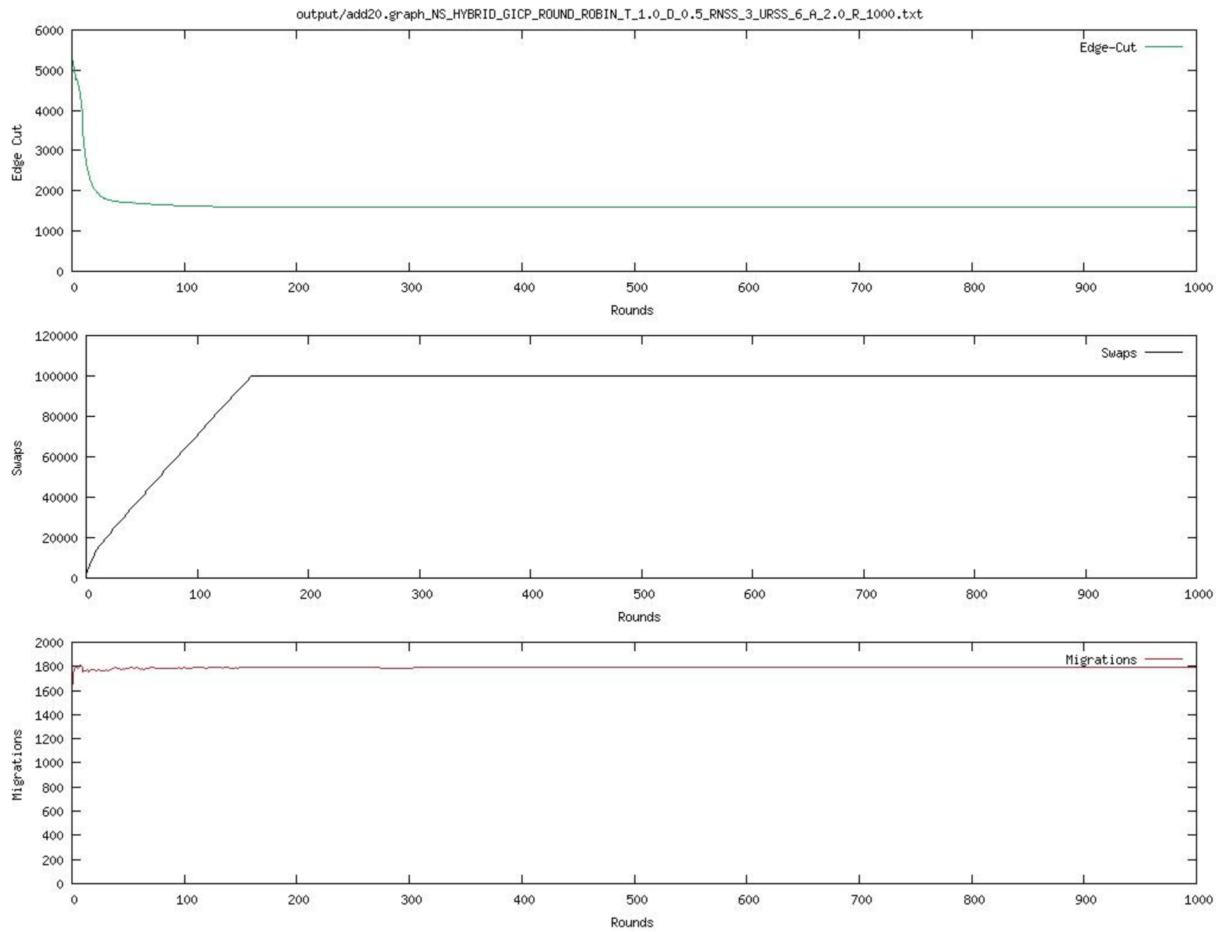


Figure 8: Exponential annealing add20(D: 0.5; T: 1; edge-cut: 1585; swap: 100277; migration: 1790; time elapsed: 58 seconds)

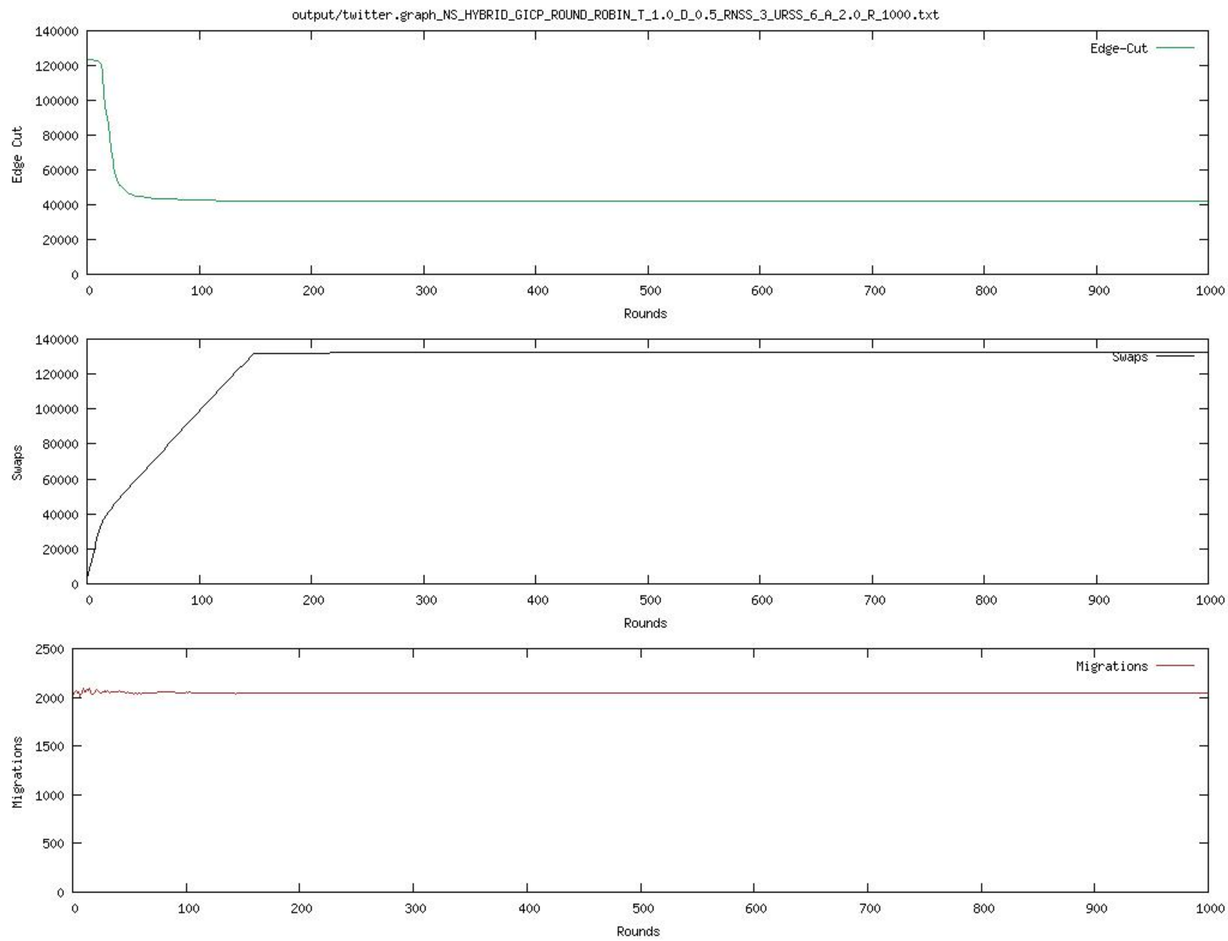
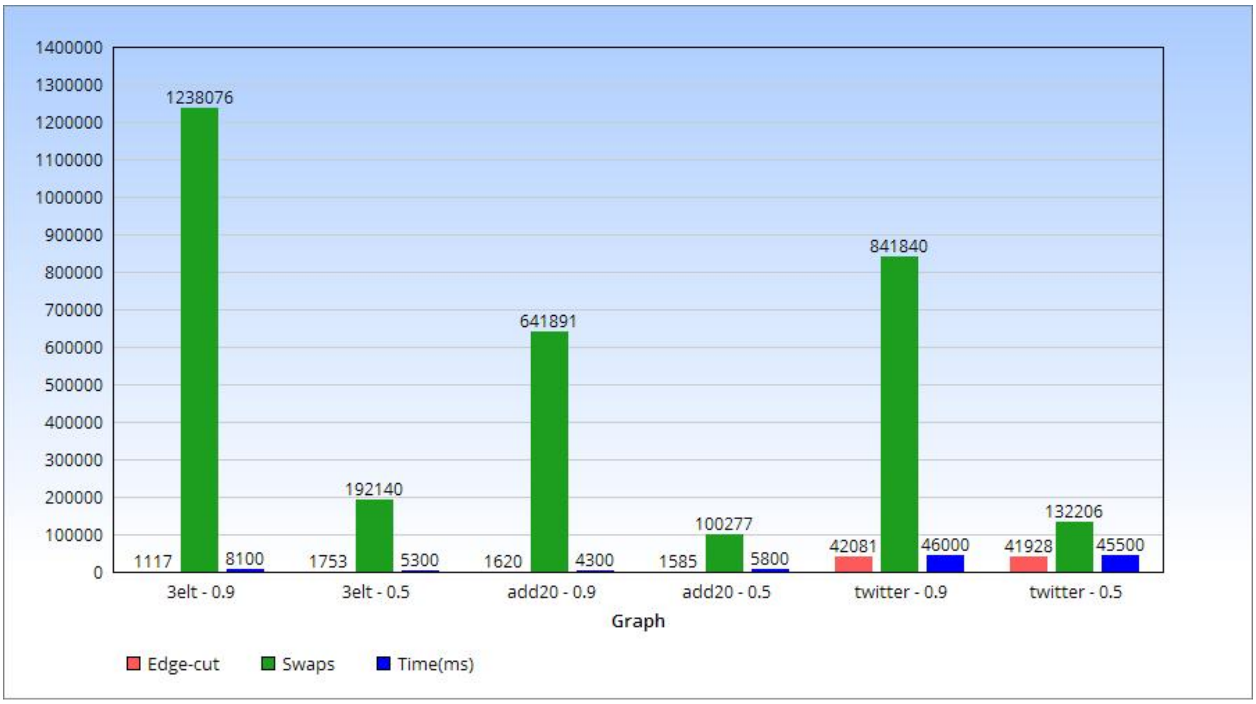
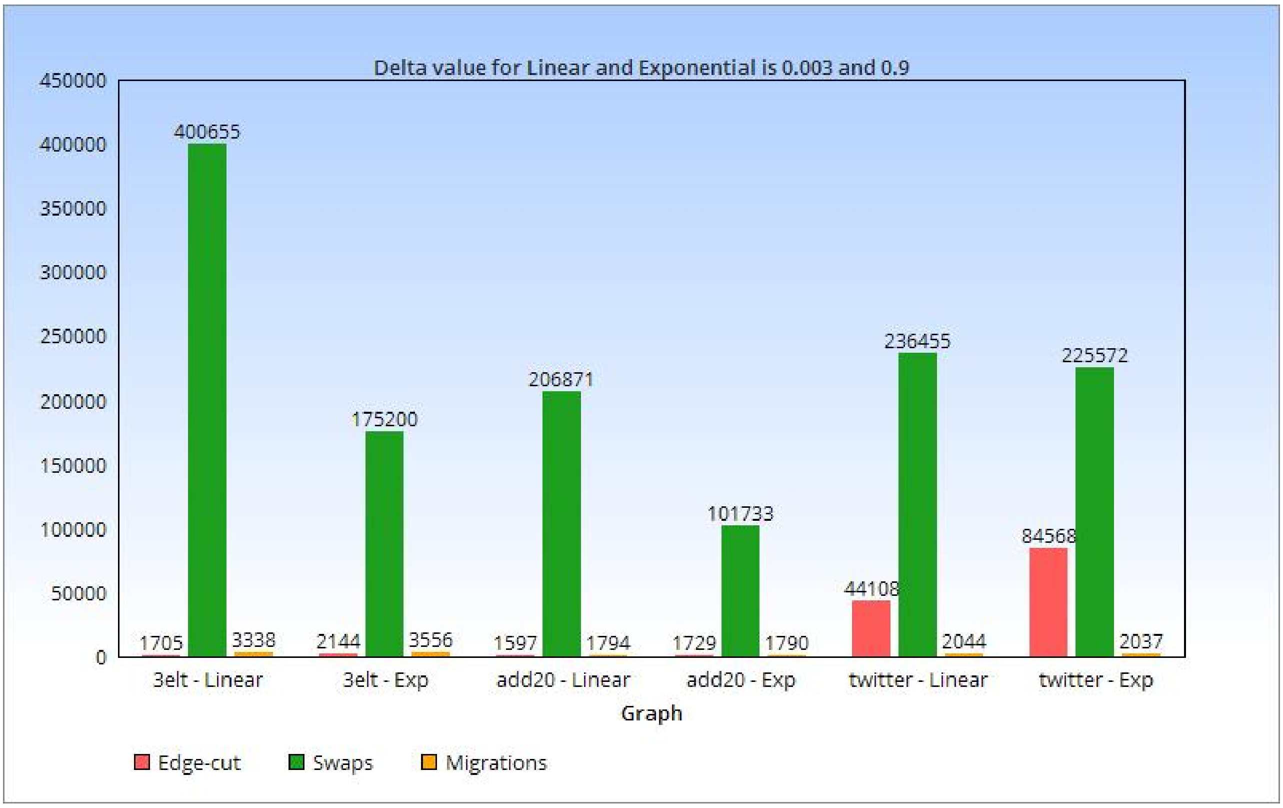


Figure 9: Exponential annealing twitter(D: 0.5; T: 1; edge-cut: 41928; swap: 132206; migration: 2043; time elapsed: 455 seconds)

Here is a bar chart to show the differences occurs on all Exponential annealing graphs while using delta value of 0.5 and 0.9 -



In the analysis of metrics, it was observed that the temperature parameter (T) could become trapped at a local minimum. Given the NP-complete nature of the problem, finding a global minimum is often impractical for real-world problems. To address this issue, an optional temperature reset mechanism was implemented after detecting convergence. The method responsible for detecting convergence and initiating the temperature reset is `resetTemperature()`. This method checks if the edge cut remains constant for a predefined number of consecutive rounds, triggering a restart to the initial temperature value. Bar chart of given below presents the results for both linear and exponential annealing policies. For the linear annealing policy, a delta value of 0.003(default) was employed, while for the exponential policy, delta was set to 0.9. The temperature reset occurs after 100 rounds of a constant edge cut. All other parameters maintain their default values in these experiments.



5. REFERENCES

- JA-BE-JA: A Distributed Algorithm for Balanced Graph Partitioning, 2013 IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems
- http://publicatio.bibl.u-szeged.hu/5295/1/taas15.pdf
- http://katrinaeg.com/simulated-annealing.html