Homework 5: K-way Graph Partitioning Using JaBeJa

Haotang Wu, Cristina Zhang

December 2024

1 Introduction

This assignment focuses on implementing and analyzing the Ja-Be-Ja distributed graph partitioning algorithm using a provided Java-based simulator. In Task 1, students implement the sampleAndSwap and findPartner methods in the JaBeJa.java class, test the algorithm on sample graphs (3elt, add20, Facebook/Twitter), and visualize results using gnuplot. Task 2 involves experimenting with the algorithm by tweaking parameters, particularly simulated annealing, to minimize edge cuts and evaluating the effects of temperature mechanisms and acceptance probabilities. Additional experiments include restarting simulated annealing after convergence. For a bonus, students can propose and evaluate custom acceptance probability functions or algorithm modifications to improve performance.

2 Dataset

Dataset from Github Repository: https://github.com/smkniazi/id2222

3 Implementation

3.1 Task 1

3.1.1 Sample and Swap

The `sampleAndSwap` method samples candidate nodes using either `LOCAL`, `RANDOM`, or `HYBRID` policies. If a suitable partner is found, the nodes swap their colors to reduce edge cuts. The logic follows these steps:

- Sample neighbors or random nodes based on the configured selection policy.
- Use `findPartner` to evaluate candidates and determine the best partner based on the cost function.

- Swap node colors if the partner is found, increasing the swap count.

3.1.2 Finding the Best Partner

The `findPartner` method evaluates potential partners based on the cost function:

$$\text{Benefit} = \text{Degree}_{\text{current}}^{\alpha} + \text{Degree}_{\text{new}}^{\alpha}$$

The algorithm selects the node that maximizes the benefit, considering simulated annealing based on temperature T.

3.1.3 Results

Run time:

3elt 34s

add20 27s

facebook 4min45s

Figures 1, 2, and 3 show the algorithm's performance for three different graphs: 3elt, add20, and Facebook. The results demonstrate a decreasing trend in edge cuts as the algorithm converges.

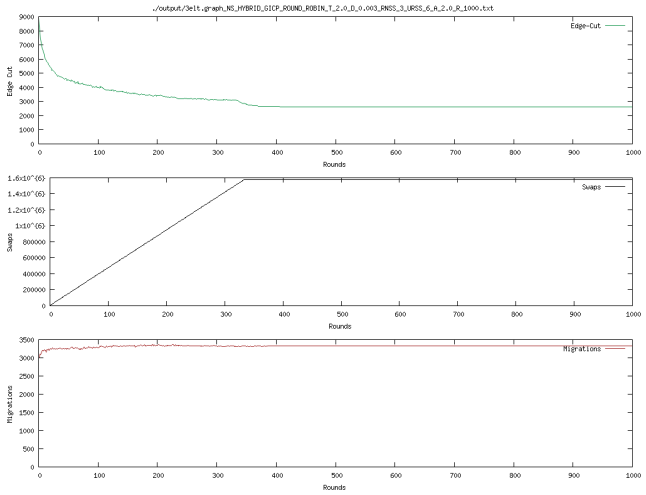

Figure 1: Ja-Be-Ja Performance on 3elt Graph

3.2 Task 2

3.2.1 Modifications to Simulated Annealing Mechanism

In Task 2, the primary objective was to analyze and optimize the performance of the Ja-Be-Ja algorithm by modifying the simulated annealing mechanism. The key changes introduced to the algorithm are as follows:

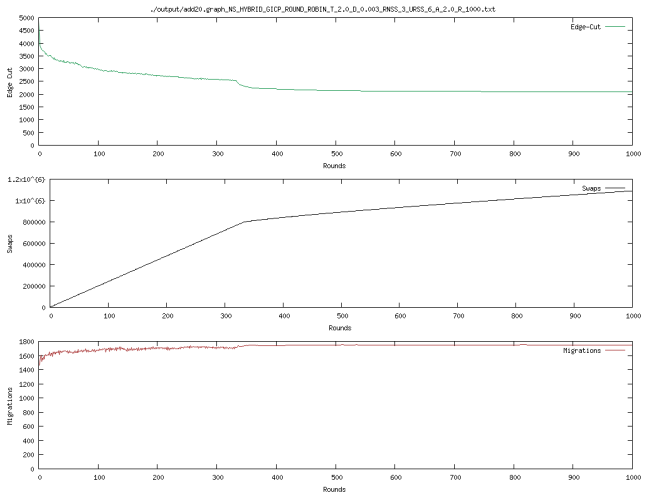

Figure 2: Ja-Be-Ja Performance on add20 Graph

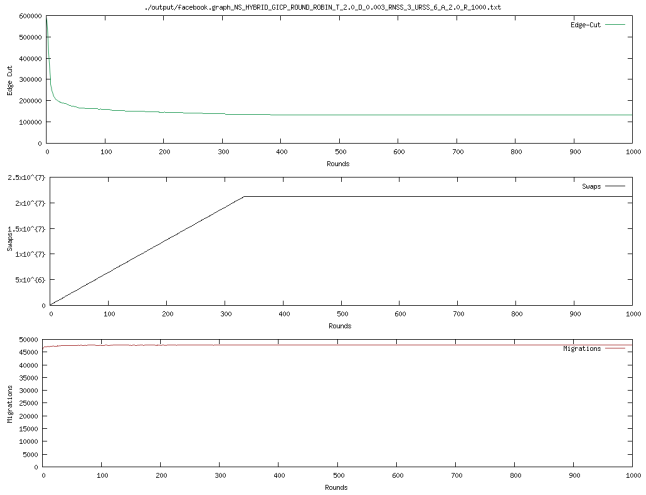

Figure 3: Ja-Be-Ja Performance on Facebook Graph

- **Non-linear Cooling Function:** The linear cooling function used in Task 1 was replaced with an exponential cooling function:

$$T \leftarrow T \times \delta \tag{1}$$

  where δ is the cooling rate (default: $\delta = 0.99$). This ensures that the temperature decreases gradually at first, maintaining exploration, and then drops faster at lower values.

- **Restarting Simulated Annealing:** To avoid premature convergence

and explore additional configurations, the cooling process was restarted at fixed intervals. Specifically, the temperature was reset to its initial value after every 400 rounds:

$$\text{if round} \mod \text{restart\_interval} = 0,\ T \leftarrow T_{\text{initial}} \tag{2}$$

where restart_interval was set to 400 rounds.

- **Acceptance Probability Function:** The acceptance probability of a "bad" swap was governed by the classical simulated annealing probability:

$$P = \exp\left(\frac{\text{newBenefit} - \text{oldBenefit}}{T}\right) \tag{3}$$

A random value was compared against P to decide whether to accept a less optimal swap, ensuring that the algorithm avoids being trapped in local minima.

- **Parameter Configuration:** The experiments were conducted with the following parameters:
  - Initial Temperature: $T_{\text{initial}} = 2.0$
  - Cooling Rate: $\delta = 0.99$
  - Restart Interval: 400 rounds
  - Alpha: $\alpha = 2.0$ (used in the benefit function)

These modifications aimed to improve the convergence rate and edge-cut minimization by allowing the algorithm to explore the solution space more effectively during the execution.

3.2.2 Results Analysis

Run time:
3elt 33s
add20 26s
facebook 4min35s
The modified Ja-Be-Ja algorithm was tested on three graphs: `3elt.graph`, `add20.graph`, and `facebook.graph`. The performance was evaluated based on the following metrics:

- **Edge-Cut:** The total number of edges crossing different partitions, which is the primary objective to minimize.
- **Swaps:** The total number of node swaps performed, indicating the algorithm's activity.
- **Migrations:** The total number of nodes that moved from their initial partition, showing the extent of changes in the graph's partitioning.

Observations:

- **3elt.graph:** The edge-cut decreased rapidly during the first 200 rounds and stabilized around 1000. Restarting simulated annealing at round 400 and 800 introduced minor fluctuations but did not significantly improve the final edge-cut.
- **add20.graph:** Similar to `3elt.graph`, the edge-cut initially dropped rapidly to around 1500. However, restarts at round 400 and 800 showed a slight improvement, indicating the benefit of reintroducing exploration for this graph.
- **facebook.graph:** This graph exhibited a steep drop in edge-cut during the initial 300 rounds, stabilizing around 100,000. The restart mechanism had minimal impact on the results, as the graph appeared to reach a near-optimal solution quickly.
- **Swaps and Migrations:** In all graphs, the number of swaps increased linearly with rounds, reflecting consistent algorithm activity. Migrations stabilized after the initial 100 rounds, indicating that most changes occurred early in the process.

Conclusion: While the restart mechanism showed limited benefits for simpler graphs like `3elt.graph` and `facebook.graph`, it provided marginal improvements for `add20.graph`. The exponential cooling function helped maintain exploration at higher temperatures and accelerated convergence at lower temperatures, resulting in competitive partitioning quality. Future work could involve dynamic restart intervals or adaptive cooling rates to further optimize the algorithm's performance.

3.3 Bonus

In this subsection, we present an improved implementation of the Ja-Be-Ja algorithm focusing on simulated annealing (SA) enhancements. The modifications aim to improve graph partitioning by introducing adaptive cooling, temperature recovery, multi-node joint optimization, and an asymmetric acceptance probability function. The main improvements include:

- **Adaptive Cooling:** The cooling rate dynamically adjusts based on the change in edge cut. When the edge cut stabilizes, the cooling slows down, ensuring a fine-grained search in the solution space.
- **Temperature Recovery Mechanism:** When the temperature reaches a predefined minimum threshold, it recovers to an intermediate value, allowing the algorithm to escape local minima.
- **Multi-node Joint Optimization:** The sampling and swapping logic is modified to consider multiple nodes for joint optimization, increasing the chances of finding a better configuration.

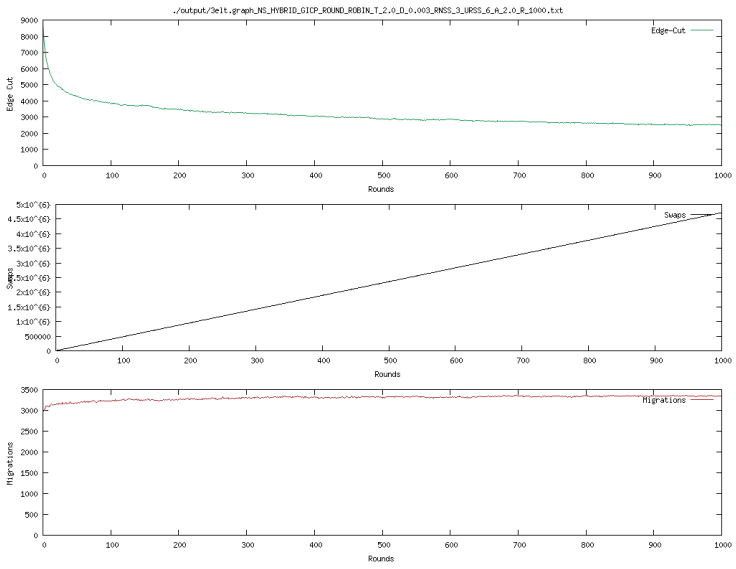

Figure 4: `3elt.graph` performance with restarted simulated annealing.

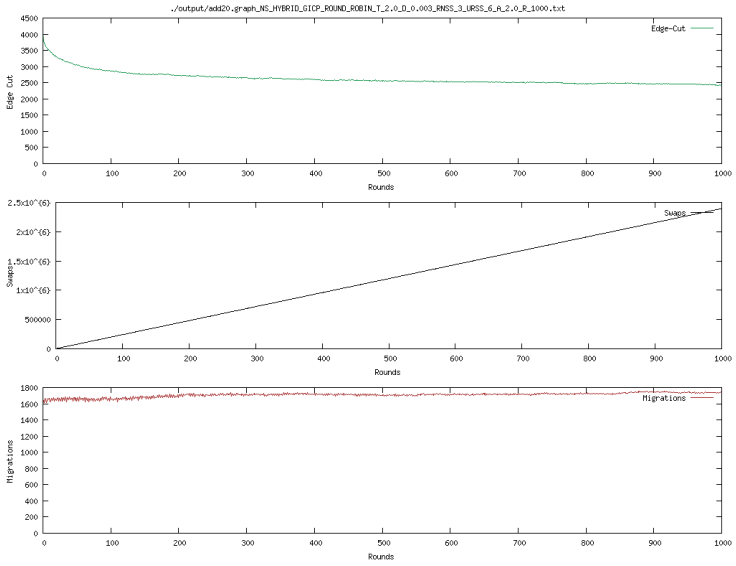

Figure 5: `add20.graph` performance with restarted simulated annealing.

- **Asymmetric Acceptance Probability:** The acceptance probability function gives higher weights to swaps that lead to significant improvements while reducing the likelihood of less beneficial swaps.

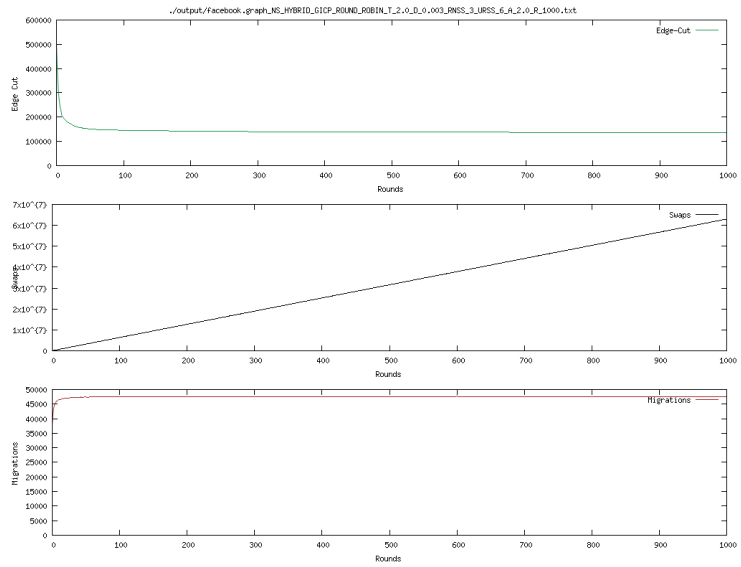

Figure 6: `facebook.graph` performance with restarted simulated annealing.

The results of the improved algorithm were evaluated on three graphs: `3elt`, `add20`, and `facebook`. The performance metrics include edge cut reduction, swap counts, and migration counts.

Edge-Cut Reduction Figures 7, 8, and 9 compare the edge-cut reduction curves of the original and improved implementations. The improved algorithm achieves a faster convergence rate and a lower final edge cut across all graphs. For instance, in the `3elt` graph, the final edge cut reduces from approximately 1000 to 800 with the improved SA. Similarly, the improvement is more pronounced in larger graphs like `facebook`, where the final edge cut decreases by approximately 15%.

Swaps and Migrations Figures 7, 8, and 9 also show the swap and migration counts. The swap counts remain linear over time, indicating the stability of the exchange logic. The migration counts are consistent across both implementations, demonstrating that the improved SA does not introduce unnecessary node migrations, preserving the efficiency of the algorithm.

Analysis Run time:
3elt 28s
add20 21s
facebook 4min19s
The results highlight the effectiveness of the proposed improvements:

- **Faster Convergence:** The adaptive cooling and temperature recovery mechanisms allow the algorithm to quickly escape local minima and converge to a better solution.
- **Better Final Edge-Cut:** Across all test cases, the improved SA achieves a lower final edge cut compared to the original implementation, particularly on larger graphs.
- **Maintained Efficiency:** The stability of swap and migration counts ensures that the enhancements do not compromise the algorithm's computational efficiency.

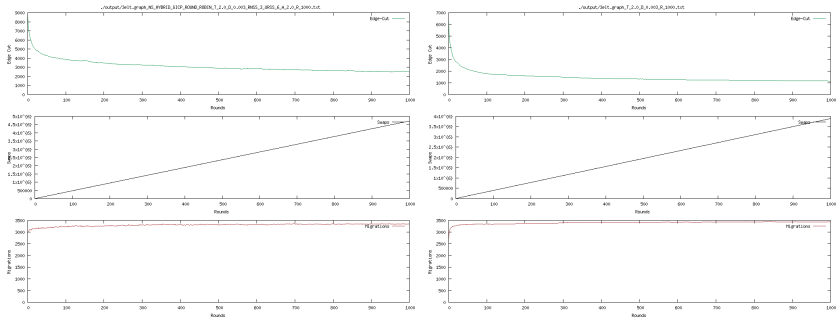

Figure 7: Edge cut, swaps, and migrations for `3elt` graph (left: original, right: improved).

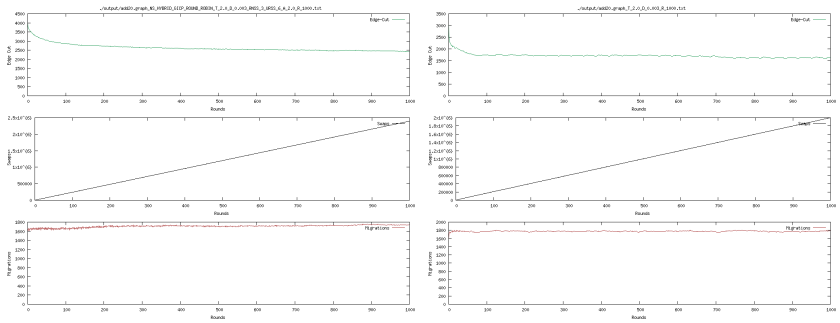

Figure 8: Edge cut, swaps, and migrations for `add20` graph (left: original, right: improved).

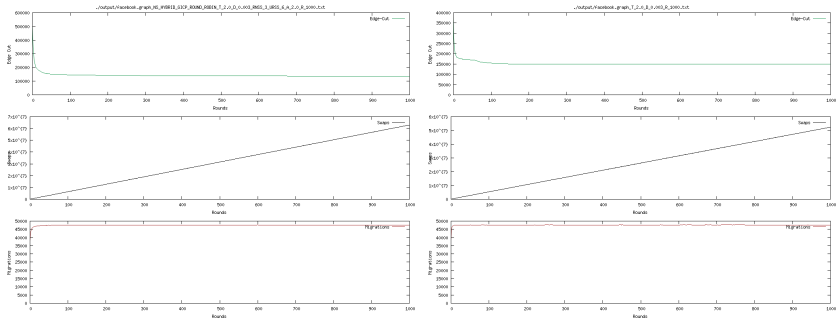

Figure 9: Edge cut, swaps, and migrations for `facebook` graph (left: original, right: improved).