Homework 5: K-way Graph Partitioning Using JaBeJa

Konstantin Sozinov, sozinov@kth.se
Kim Hammar, kimham@kth.se

December 12, 2017

1 Introduction

The main algorithm can be found in the *Jabeja.java* file. We implemented the Ja-Be-Ja distributed graph partitoning algorithm in Java, following the given template code as well as the pseudo-code in the paper. This report includes evalautions of the algorithm on three graphs: add20, elt3 and twitter. The evaluations indicate that different parameters of the algorithm are suitable for each of the graphs in a number of metrics: convergence time, edge-cut, swaps, and node-migrations. Finally, we implemented our own extension to the simulated annealing of Ja-Be-Ja and demonstrate some interesting results.

The algorithm is vertex-parallel. Each vertex performs local search to improve the partitioning by minimizing the edge cuts locally. Formally, the local search of node p tries to minimize $arg\min_c \sum_{v \in N_p} d_p - d_p(c)$, where N_p is the neighborhood of p, $d_p = |N_p|$, and $d_p(c)$ is the number of neighbors with color c. Each vertex only has access to its local view of nodes and edges in the graph, and attempts to swap colors with its neighbors to improve its local situation, and (hopefully) also improve the global partitioning. The global edge-cut size is also denoted as the energy of the system. Ja-Be-Ja is an heuristic algorithm based on a portion of randomness when using simulated annealing. Ja-Be-Ja does not provide any upper or lower bound guarantees on the resulting partitions. Nodes only swap colors if it reduces their local energy. There are three policies to select nodes for swapping: random, local, and uniform. A node will go through all its selected nodes and see which one is best to swap with for each iteration.

To escape local minimas, Ja-Be-Ja utilizes simulated annealing. The basic simulated annealing outlined in the paper uses a temperature factor T, when $T > 1$ swap-decisions are biased towards swapping rather than not-swapping, even if it could increase the energy of the system. T is reduced over time until it reaches 1. The second version of simulated annealing uses the technique outlined in a blog post [1]. To summarize, this approach to simulated annealing goes over all neighboring solutions and selects the best solution using the following formula for acceptance probability, $ap = e^{\frac{c_{old} - c_{new}}{T}}$. Furthermore, we also implemented this technique using restarts, meaning that when $T = 0$ it is reset back to 1.

Finally, our own version of simulated annealing got inspiration from the Momentum technique, known to improve Gradient Descent convergence time. Formally we use momentum as follows:

$$momentum = max(0, \mu \cdot (c_{new}^{(t)} - c_{new}^{(t-1)}))$$

$$ap = e^{\frac{c_{old} - (c_{new} - momentum)}{T}}$$

Where μ is the momentum coefficient. When using momentum we did not use any restarts of T.

2 Evaluation and results

For all tasks we used the hybrid selection policy as that was presented as the best policy in the paper. Additionally we tried to stick to the values of α and δ that performed best in the paper.

[1] http://katrinaeg.com/simulated-annealing.html

2.1 Task 1 - Linear Simulated Annealing, no restarts, no randomness

graph	delta	T	edge-cut	rounds	swaps	migrations	partitions	converge	alpha	policy
add20	0.003	2	2095	1000	1090263	1751	4	yes	2	hybrid
3elt	0.003	2	2604	1000	1580209	3328	4	yes	2	hybrid
twitter	0.003	2	41156	1000	899515	2049	4	yes	2	hybrid

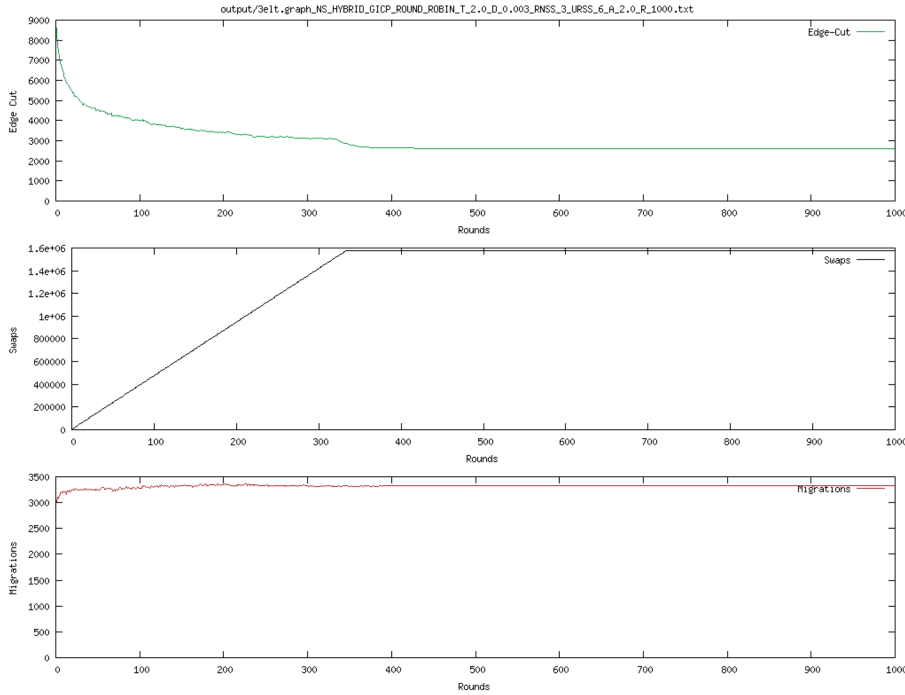

Figure 1: 3elt

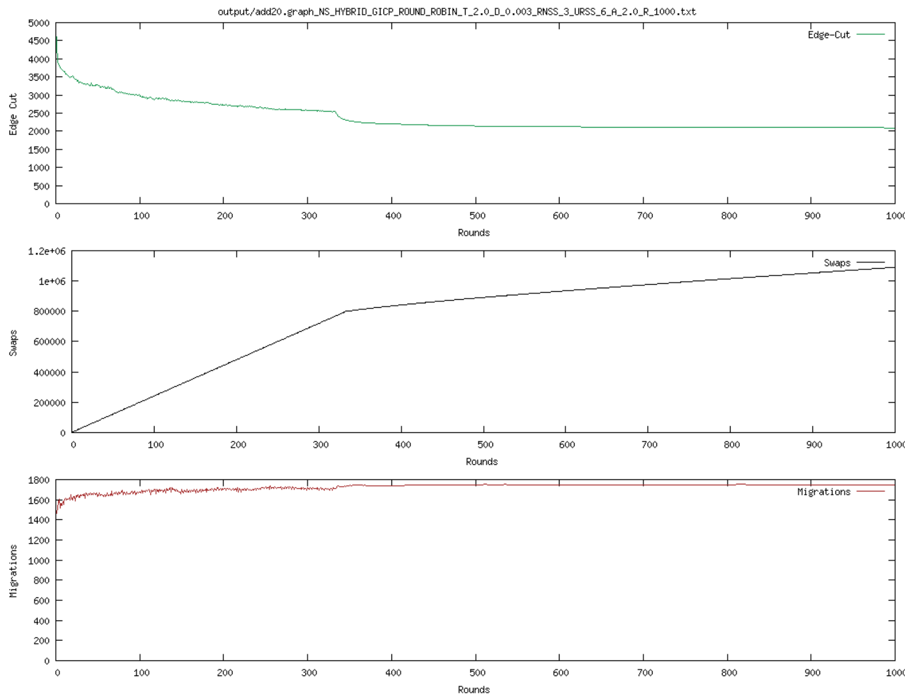

Figure 2: add20

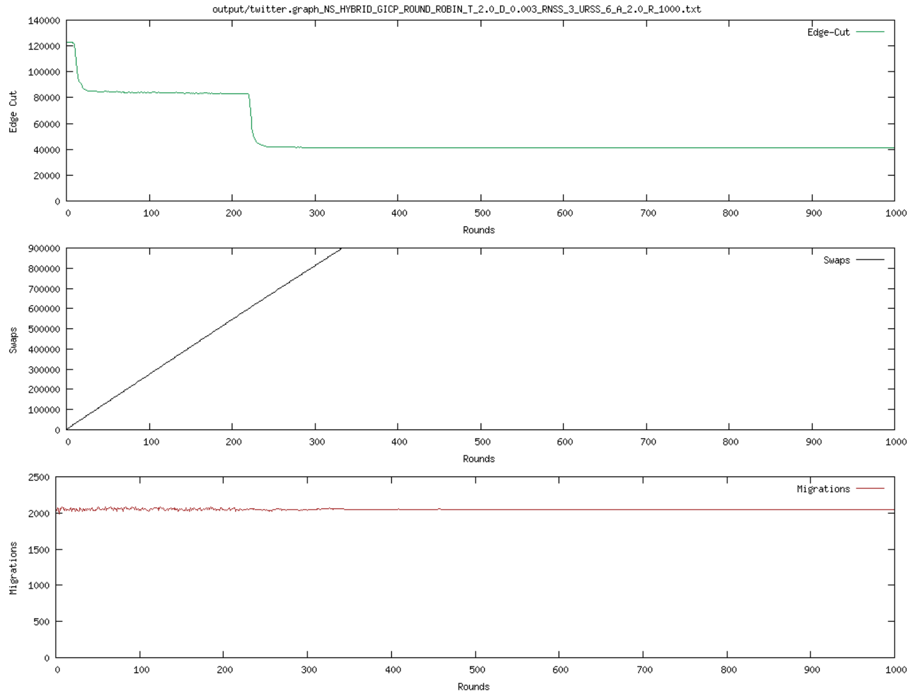

Figure 3: twitter

What can be noted about these results are that the algorithm converged early and to a bad solution (local optima) in all three cases.

2.2 Task 2.1 - Linear Simulated annealing, no restarts, randomness and acceptance probability

graph	delta	T	edge-cut	rounds	swaps	migrations	partitions	converge	alpha	policy
3elt	0.003	1	2190	1000	103586	3274	4	yes	2	hybrid
add20	0.003	1	2060	1000	373826	1745	4	yes	2	hybrid
twitter	0.003	1	41115	1000	48804	2046	4	yes	2	hybrid

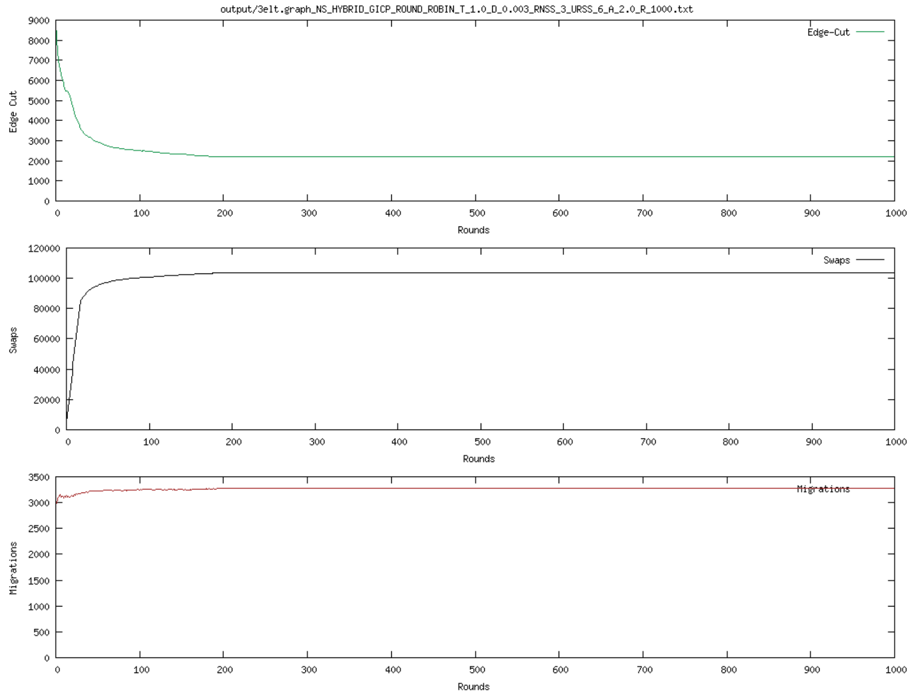

Figure 4: 3elt

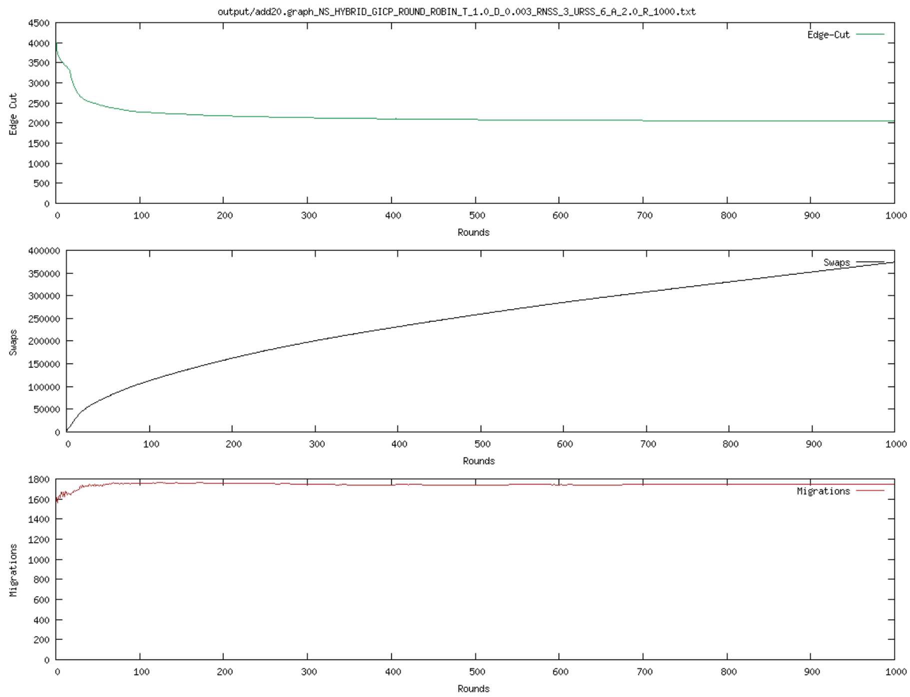

Figure 5: add20

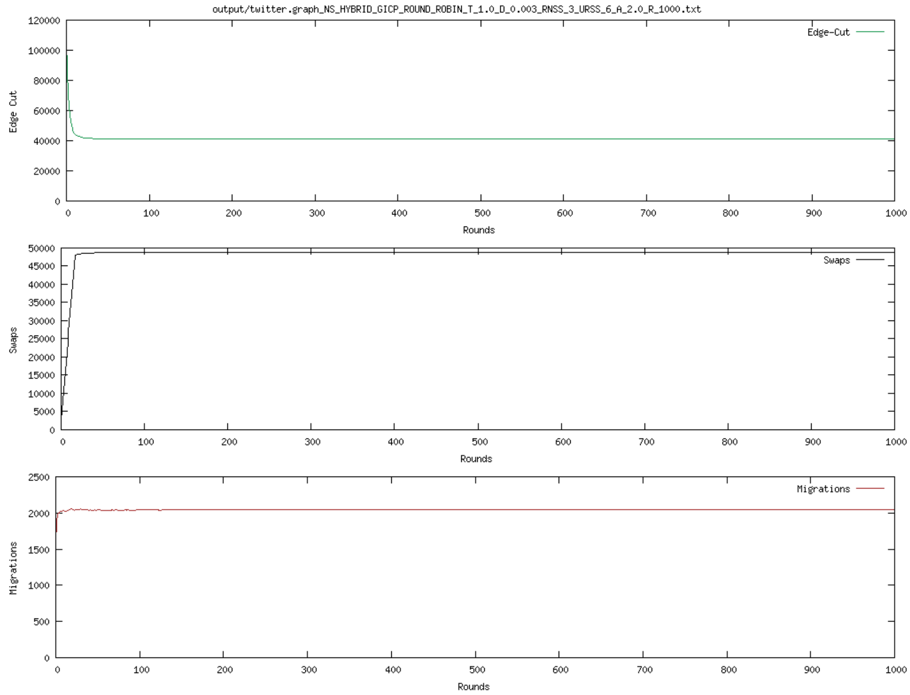

Figure 6: twitter

What can be noted about these results are that the new simulated annealing technique gave a lot better results on 3elt, but almost the same results on add20 and twitter graphs. Furthermore the algorithm converged in all cases (local optima problem again).

2.3 Task 2.2 - Linear Simulated annealing with restarts, randomness and acceptance probability

graph	delta	T	edge-cut	rounds	swaps	migrations	partitions	converge	alpha	policy	restart
3elt	0.003	1	2037	1000	4463446	3296	4	no	2	hybrid	1
add20	0.003	1	2348	1000	2303961	1746	4	no	2	hybrid	1
twitter	0.003	1	41147	1000	2494681	2049	4	yes	2	hybrid	1

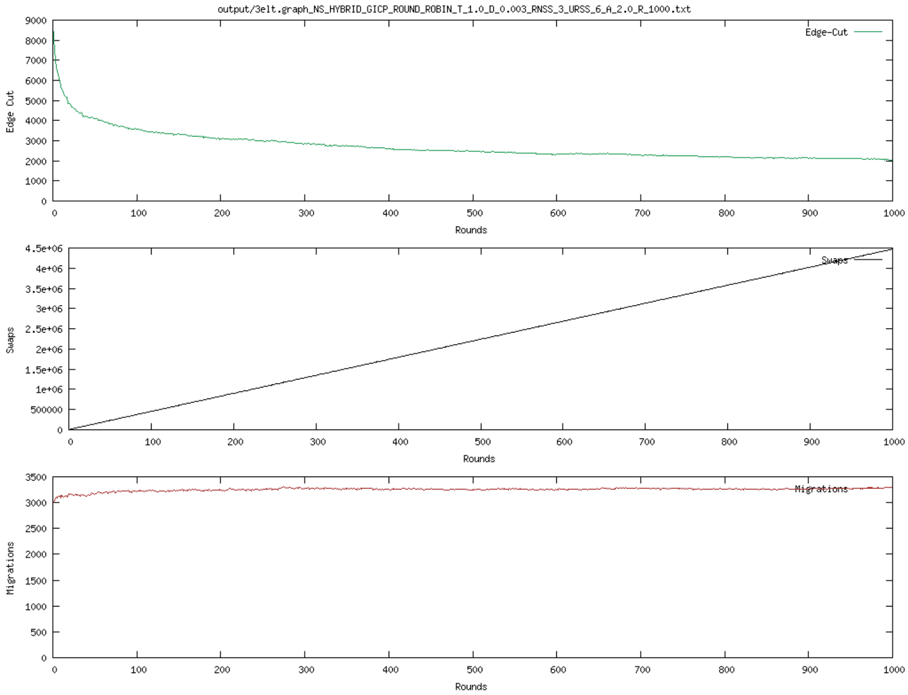

Figure 7: 3elt

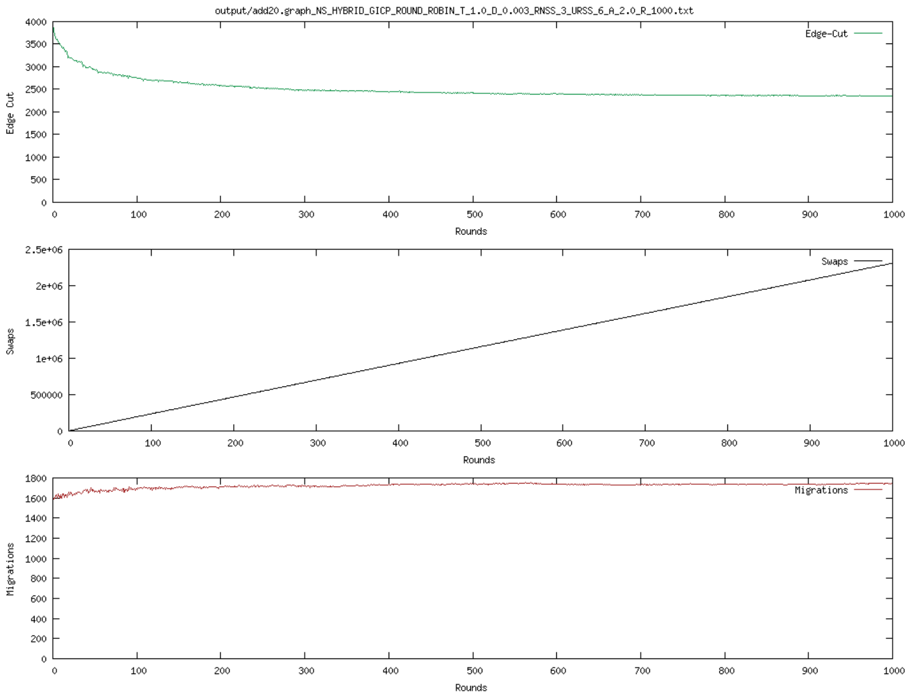

Figure 8: add20

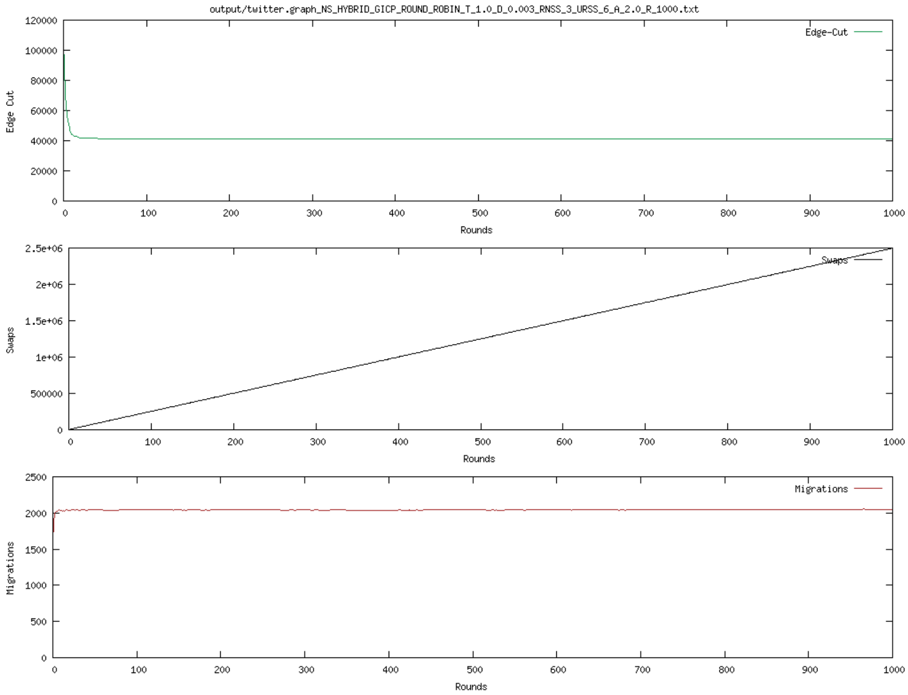

Figure 9: twitter

The results demonstrate that adding restarts did not help much without tuning the rest of the parameters.

2.4 Task 2.3 Exponential simulated annealing with restarts, randomness and acceptance probability

graph	delta	T	edge-cut	rounds	swaps	migrations	partitions	converge	alpha	policy	restart
3elt	0.003	1	2504	1000	4713193	3328	4	no	2	hybrid	1
add20	0.003	1	2471	1000	2392641	1679	4	no	2	hybrid	1
twitter	0.003	1	41327	1000	2636468	2045	4	yes	2	hybrid	1

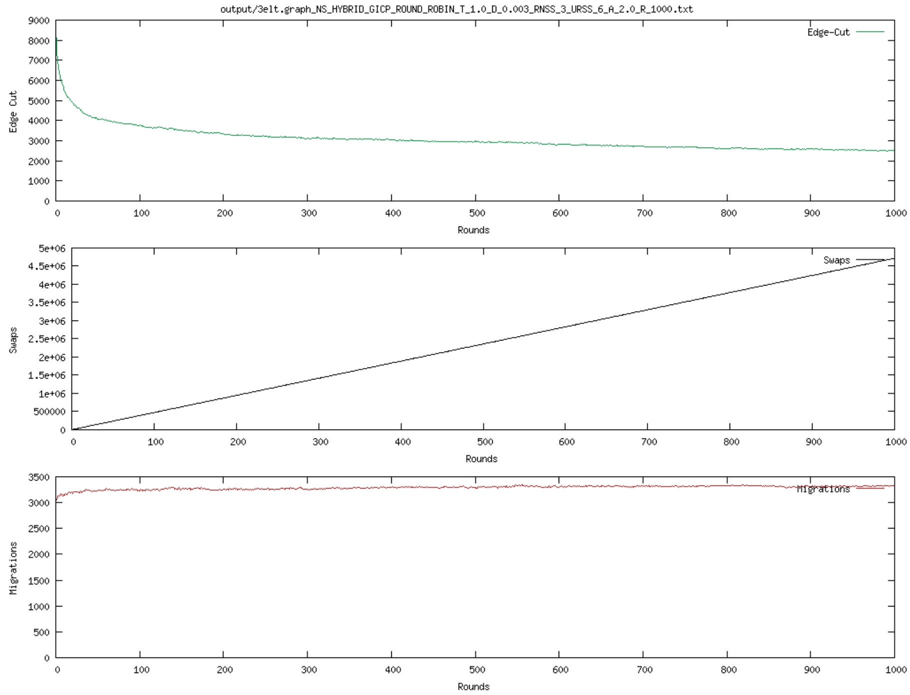

Figure 10: 3elt

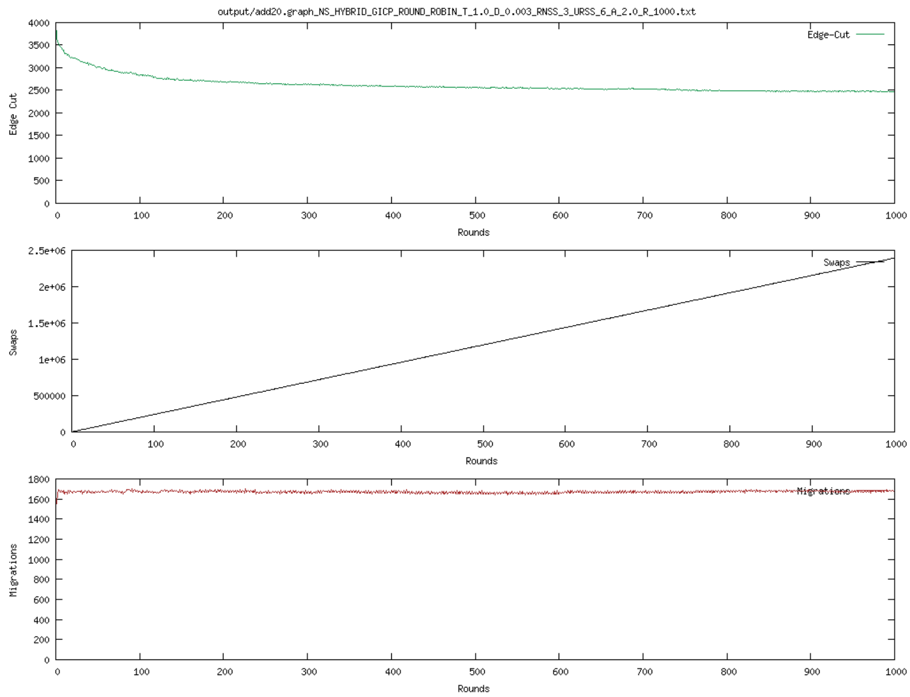

Figure 11: add20

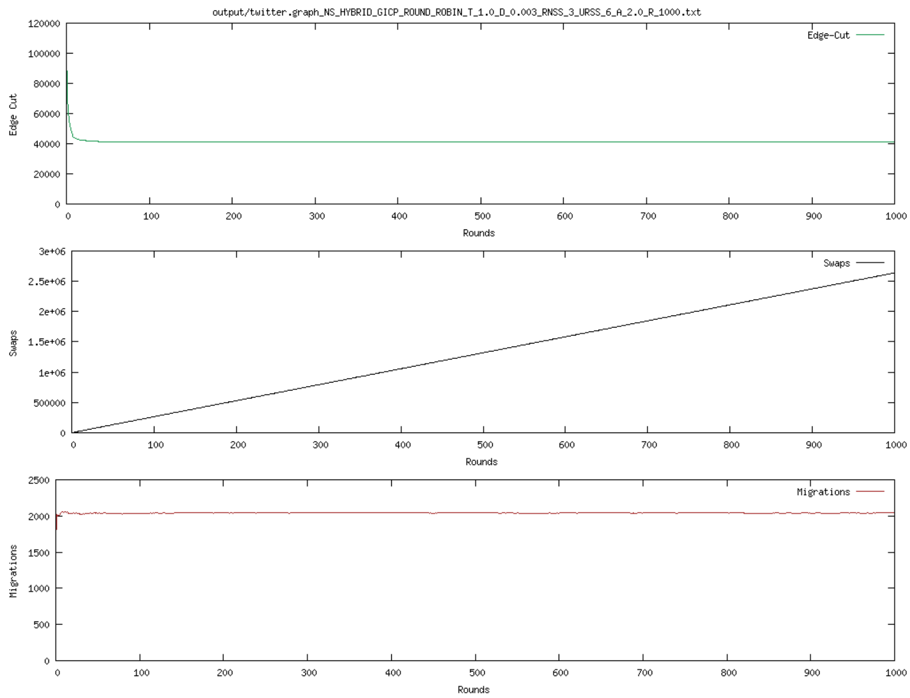

Figure 12: twitter

The results with exponential simulated annealing without parameter tuning gave consistently worse results.

2.5 Task 2.4 Linea simulated annealing with restarts, randomness and acceptance probability - Parameter tuning

graph	delta	T	edge-cut	rounds	swaps	migrations	partitions	converge	alpha	policy	restart
3elt	0.00001	1	1021	10000	42010302	3441	4	no	2	hybrid	1
3elt	0.00001	1	1208	10000	42541398	3425	4	no	1	hybrid	1
3elt	0.003	1	1011	10000	44596460	3422	4	no	2	hybrid	1
3elt	0.003	1	**731**	50000	222928227	3435	4	yes	2	hybrid	1
add20	0.00001	1	2196	10000	22126510	1753	4	no	2	hybrid	1
add20	0.00001	1	1792	10000	21773341	1757	4	yes	1	hybrid	1
add20	0.003	1	**1780**	10000	22704110	1734	4	yes	1	hybrid	1
twitter	0.00001	1	41258	2000	4739316	2046	4	yes	2	hybrid	1
twitter	0.003	1	**40841**	2000	5000891	2043	4	yes	1	hybrid	1

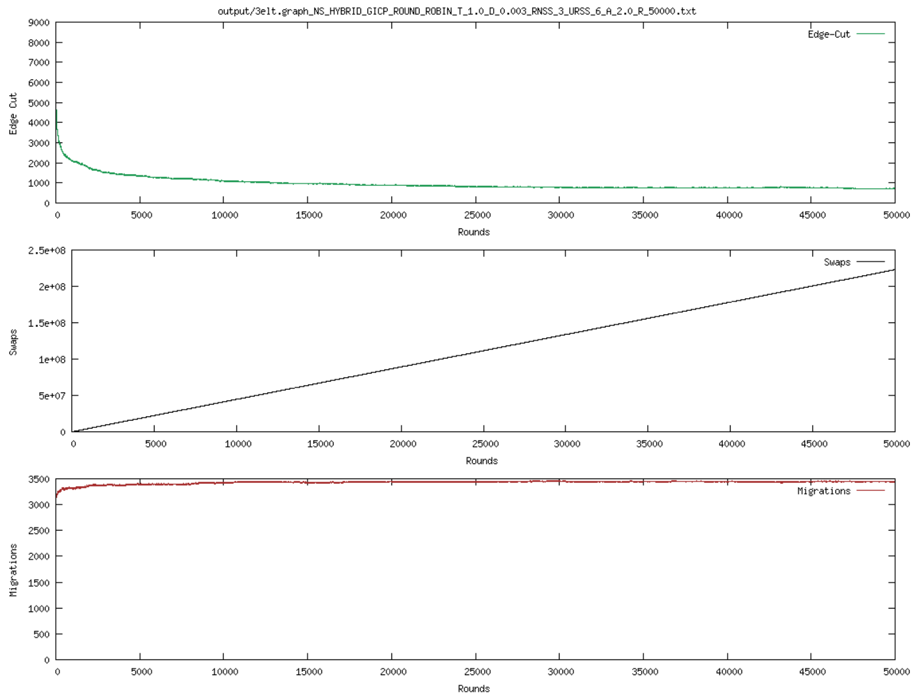

Figure 13: 3elt

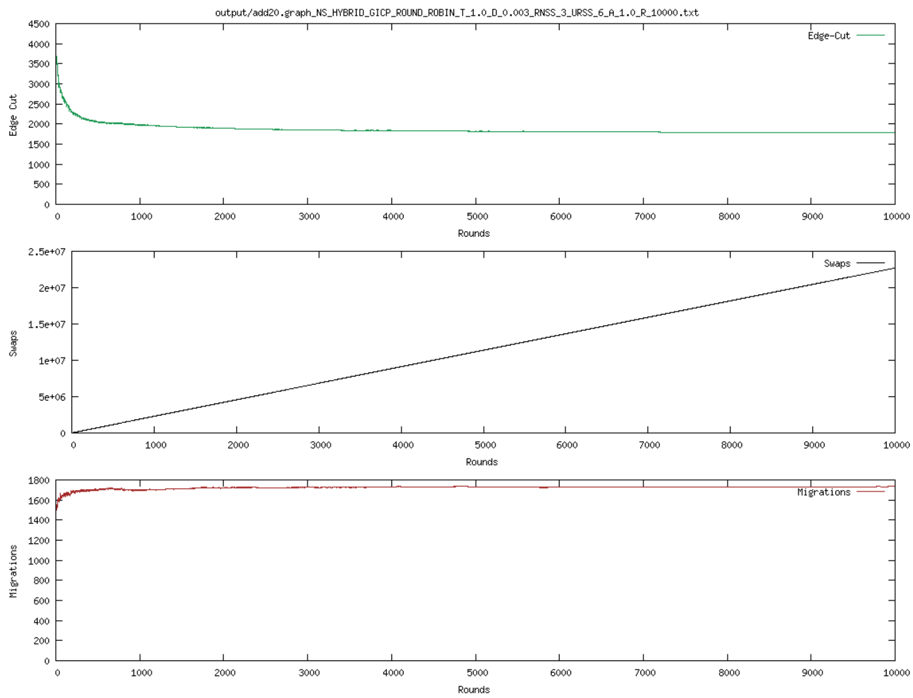

Figure 14: add20

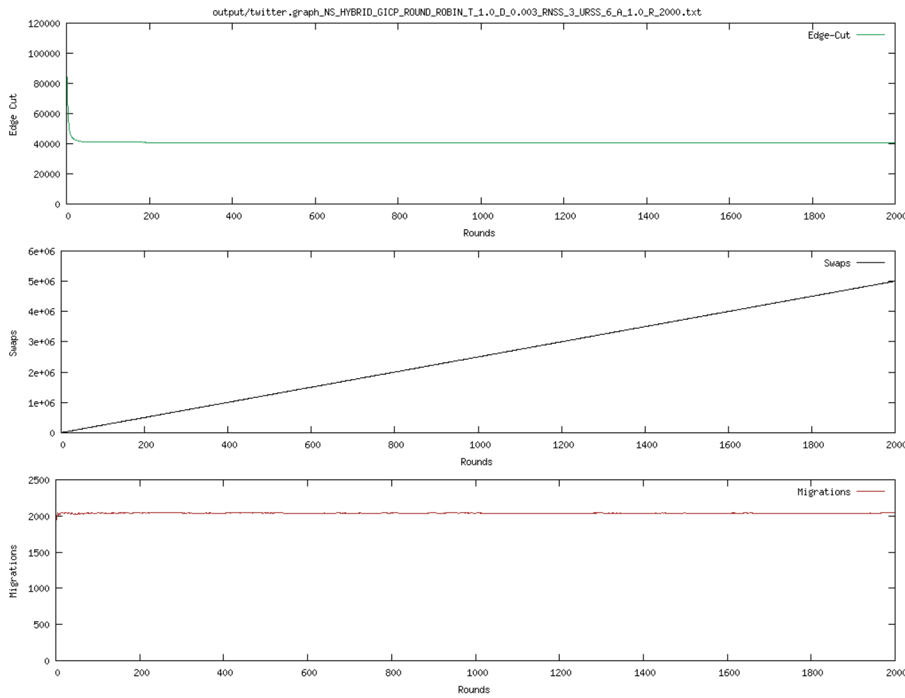

Figure 15: twitter

From these results we can see that by tuning some parameters, primarily alpha, and the number of rounds, we can greatly improve the partitions on all graphs. Both twitter and add20 gave better results with $\alpha = 1$.

2.6 Bonus Task: Momentum + Simulated Annealing

graph	delta	T	edge-cut	rounds	swaps	migrations	partitions	converge	alpha	policy	momentum
3elt	0.003	1	1256	1000	4280889	3420	4	no	2	hybrid	0.001
3elt	0.003	1	5139	1000	4685823	3535	4	no	2	hybrid	10
3elt	0.003	1	1344	1000	4281498	3397	4	no	2	hybrid	0.0001
3elt	0.003	1	697	10000	42315849	3457	4	no	2	hybrid	0.001
3elt	0.003	1	**518**	50000	210569840	3463	4	yes	2	hybrid	0.001
add20	0.003	1	2095	1000	2294945	1815	4	no	2	hybrid	0.001
add20	0.003	1	**1997**	10000	22776283	1785	4	yes	1	hybrid	0.00001
twitter	0.003	1	41137	1000	2485027	2034	4	yes	2	hybrid	0.001
twitter	0.003	1	40878	1000	2498748	2035	4	yes	1	hybrid	0.001
twitter	0.003	1	**40833**	1000	2488748	2041	4	yes	1	hybrid	0.0001
twitter	0.003	1	41436	1000	2490911	2068	4	yes	1	hybrid	0.00001

With the momentum technique we achieved the best results on 3elt! And about the same results on add20 and twitter. Primarily the momentum improved the convergence time as we can see that only after 1000 iterations we got pretty good results on 3elt.

3 Conclusion

We got pretty close to the results presented in the paper but still a bit off, this is likely because we did not tune all parameters, for example we did very little tuning of $\delta, T, \alpha, restart, momentum$. For proper evaluation we could have applied grid search or random search to find the optimal parameters.

Finally, momentum looks like an promising techniques to improve convergence rate on some graphs.

4 How to run

Clone this repository and navigate to *jabeja* project. Then use:

```
/run.sh -graph ./graphs/3elt.graph -rounds 5000 -numPartitions 4 -temp 1
-delta 0.00001 -restart 0.000001 -alpha 1 -nodeSelectionPolicy HYBRID -momentum 0.001
```