Homework 5: K-way Graph Partitioning Using JaBeJa

Group 11:

Boyu Xue (boyux@kth.se), Kristupas Bajarunas(kristupasbajarunas@gmail.com)

Table of Contents

Introduction:

In this assignment we are asked to implement the JA-BE-JA algorithm of F. Rahimian. The main task of the algorithm is to divide Vertices of a graph G into balanced k-way partitions. It achieves this by assigning a color to each node and calculating the energy of each node which is the number of neighbors of a node with a different color, next nodes are allowed to switch colors in order to minimize the total energy of the graph.

Task 1:

For task 1 we had to implement the JA-BE-JA algorithm. Mainly the task was to finish two classes described in the paper. Class *SampleandSwap* was responsible for node selection. In this case hybrid node selection was implemented as first local nodes were considered for swap, and if that did not work a node was drawn from a sample. Class *FindPartner* calculates if it is possible to swap with a partner it also includes a bias term Temperature that helps selecting new states in the beginning.

The results of task one is as follows:

We found that 3elt data converges to around 2600 edge cuts. The add20 data converges to 1800 in two motions. While the Twitter data converges to 40000 edge cuts in also 2 motions. We believe the swift drops happen because of random nodes being swapped.

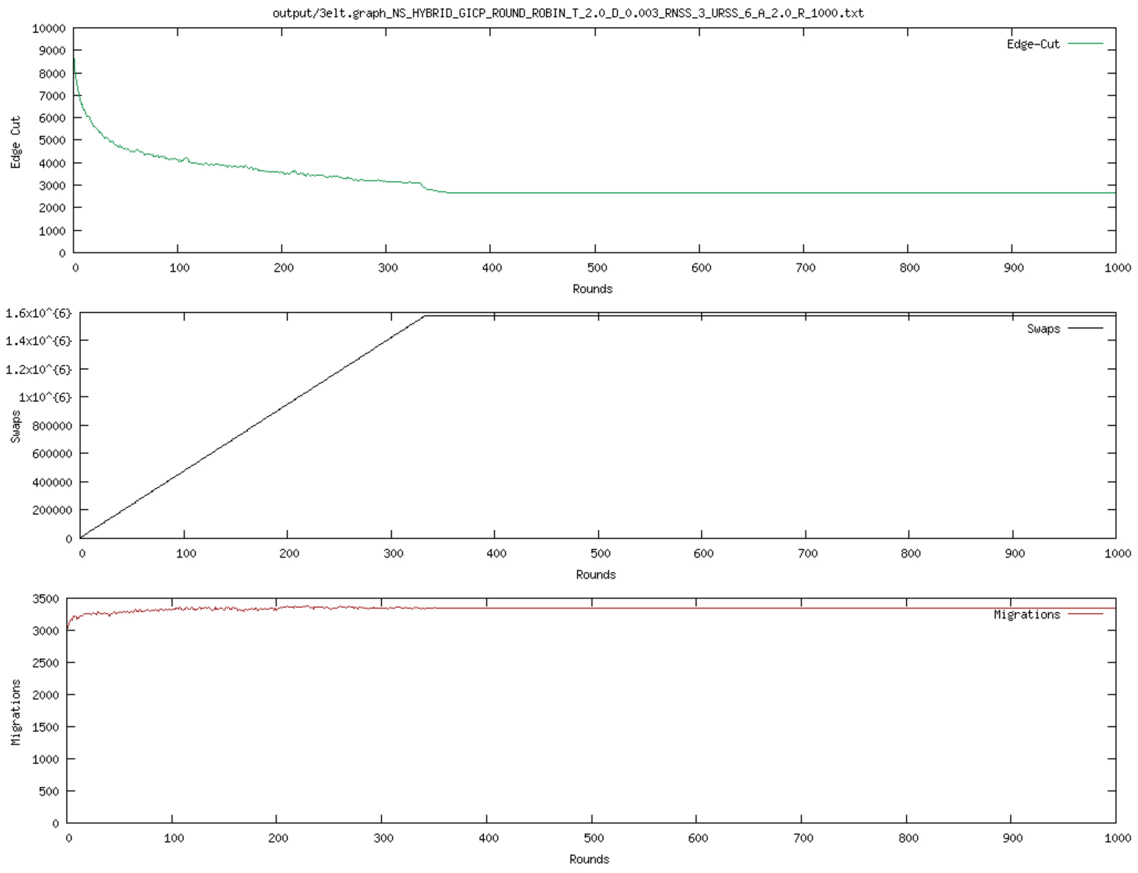

Fig.1. graph-3elt-task1

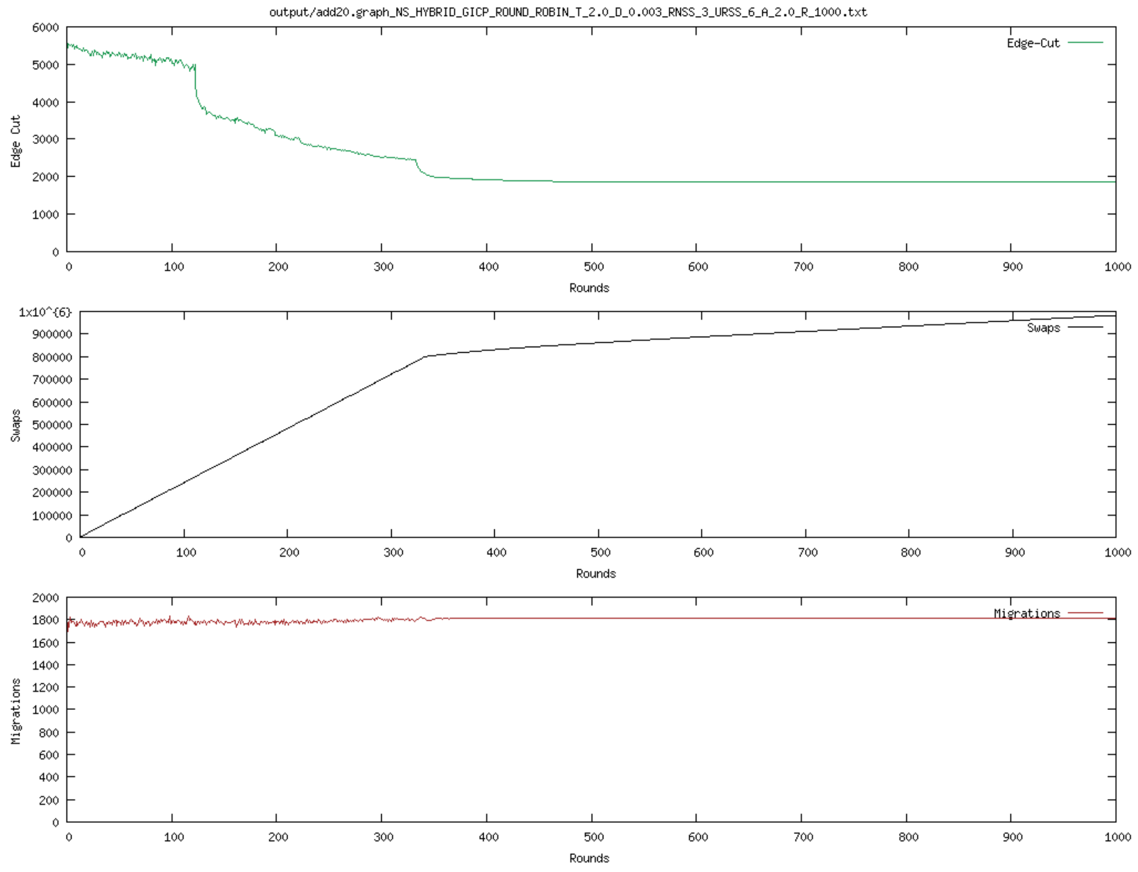

Fig.2. graph-add20-task1

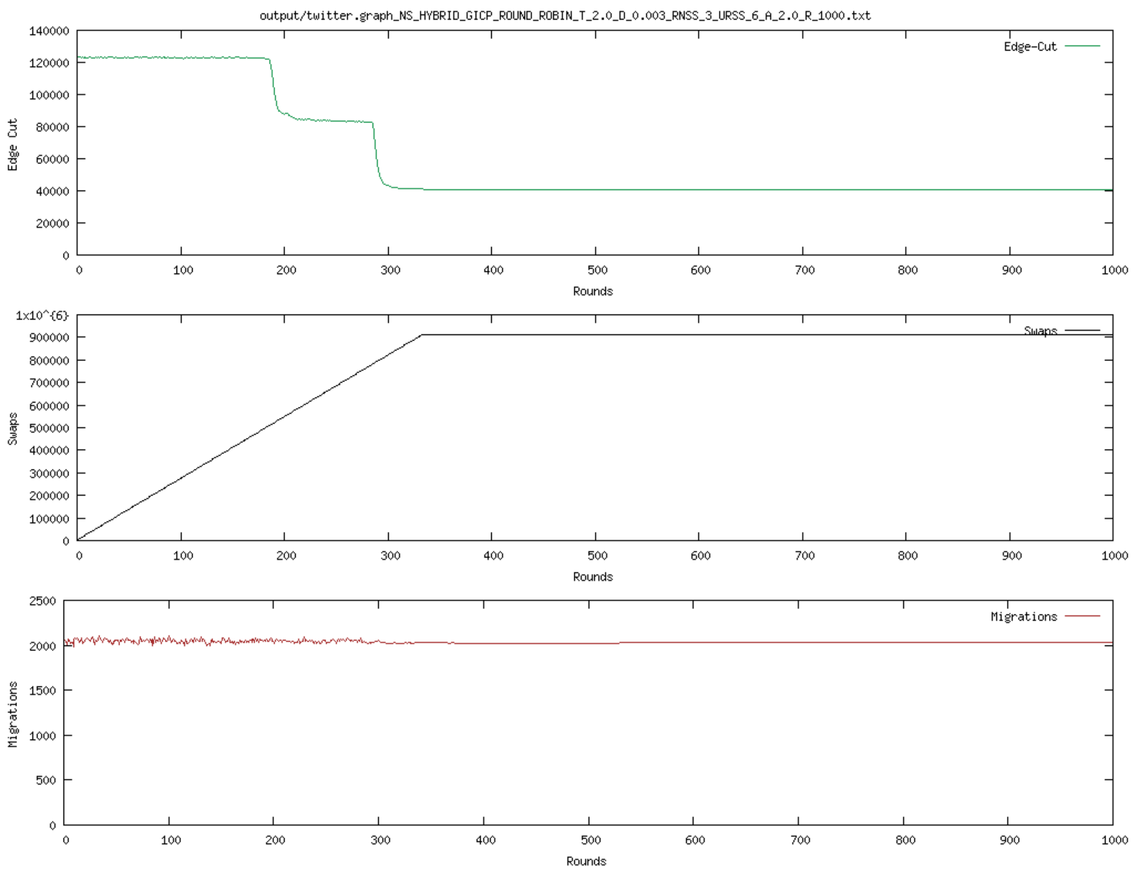

Fig.3. graph-twitter-task1

Task 2, without reset:

For task 2 we were asked to tweak some different parameters of the JA-BE-JA algorithm. First we implemented a different simulated annealing algorithm that uses temperature to make acceptance probabilities. This temperature is now decreasing exponentially instead of linearly. The maximum initial temperature is 1 and it decreases to 0.0001. Acceptance probability is given by $ap = e^{\frac{NewScore-OldScore}{T}}$ where $T = T * \delta$. The delta is the variable that will be tweaked. As it can be seen the acceptance probability is always >1 when the new score has improved, additionally when the temperature is high in the beginning even not improved new score can still be accepted with a certain probability. Once T=Tmin=0.0001 the probability to accept a bad score becomes very low.

The results indicate that for the 3elt data decreasing delta from 0.9 to 0.2 does not really influence the edge cut significantly. Interestingly number of swaps increased from 20k to 24k. Swaps indicate the cost of the algorithm. As delta decreases, the amount of steps needed to decrease from T to Tmin also decreases. Thus the algorithm makes less bad steps, which means it is more likely to get stuck in local sub-optimal solutions.

As for add20 data it seems like delta=0.8 performs the best because the edge cut is lowest at around 2000 and the number of swaps is lowest as well.

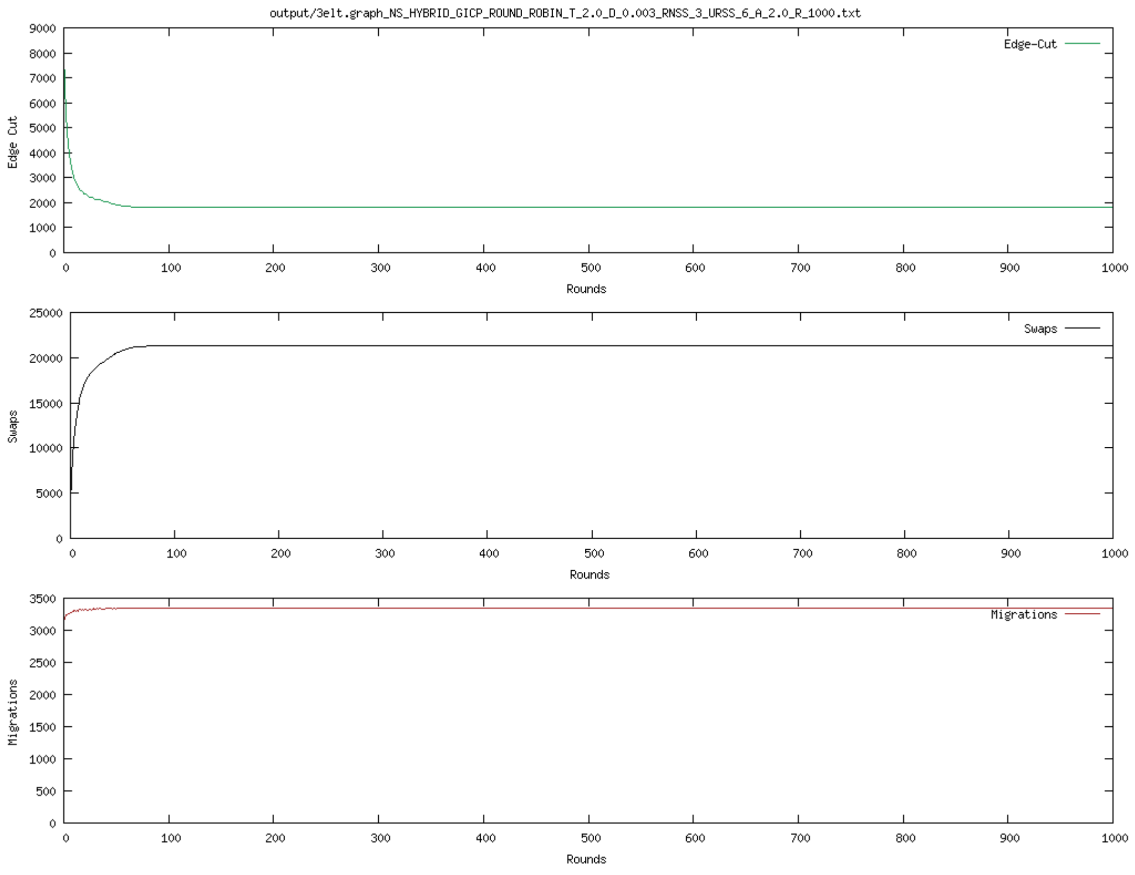

Fig.4. graph-3elt-task2. T = 1; Tmin = 0.00001; alpha = 2; delta = 0.9.

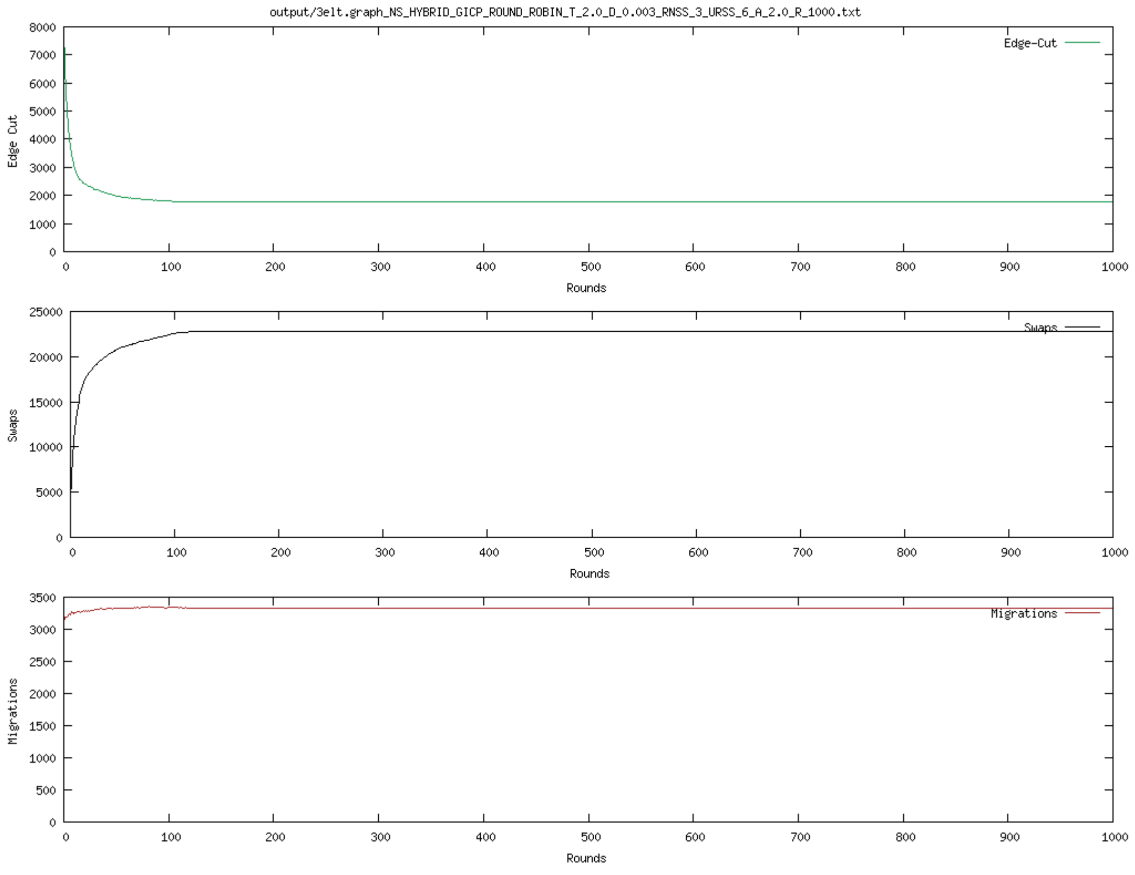

Fig.5. graph-3elt-task2. T = 1; Tmin = 0.00001; alpha = 2; delta = 0.8.

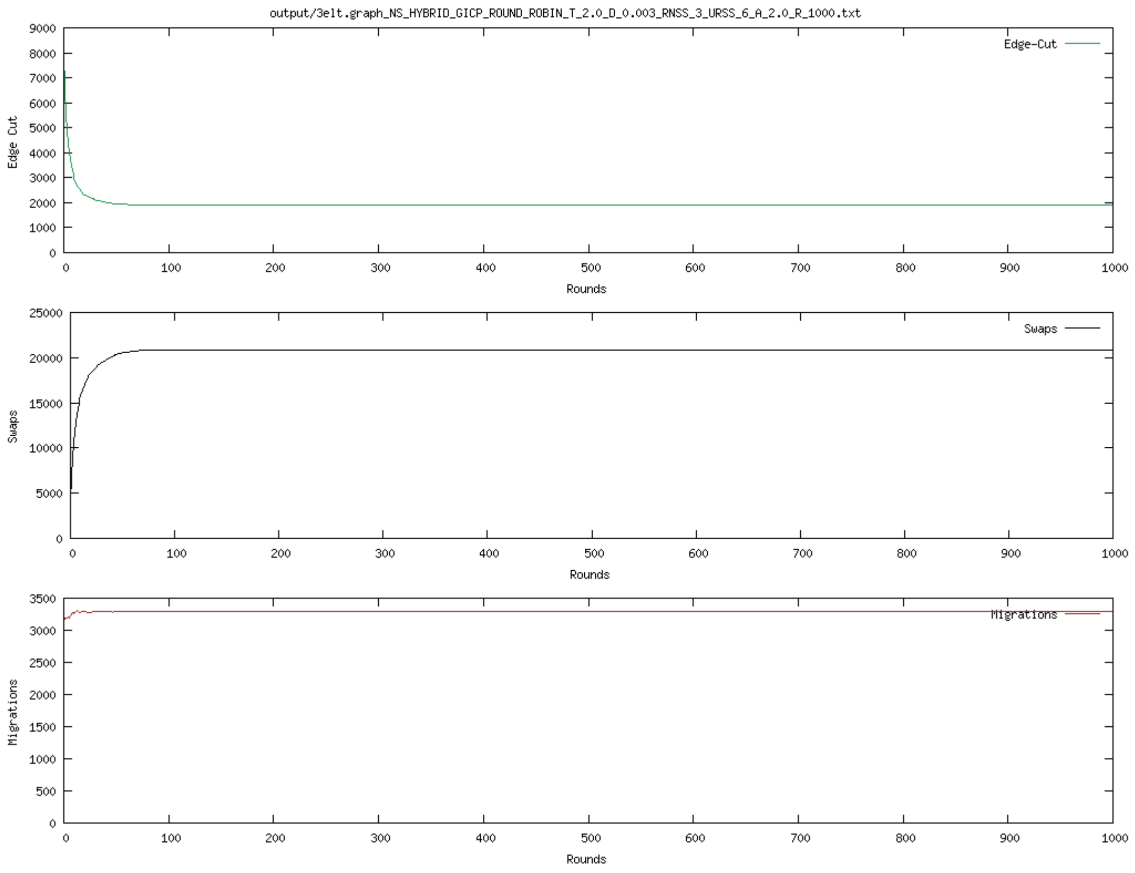

Fig.6. graph-3elt-task2. T = 1; Tmin = 0.00001; alpha = 2; delta = 0.7.

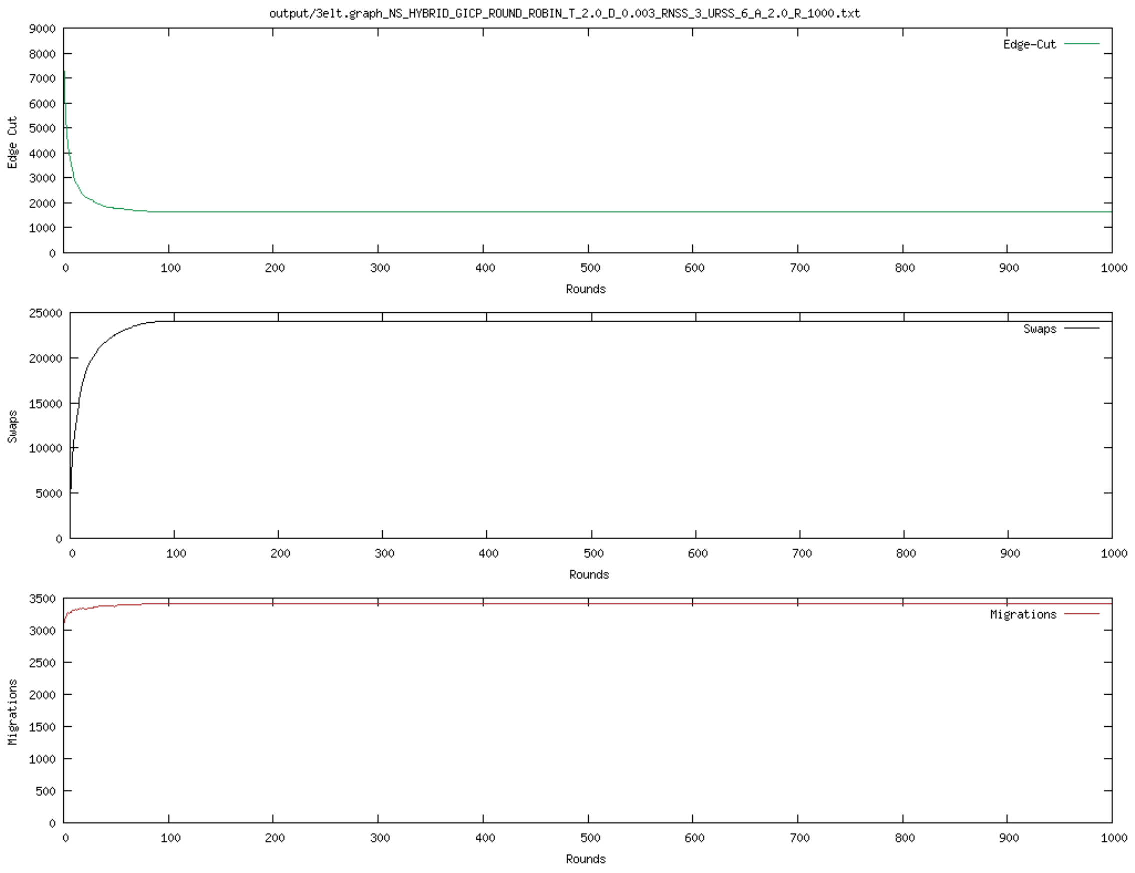

Fig.7. graph-3elt-task2. T = 1; Tmin = 0.00001; alpha = 2; delta = 0.5.

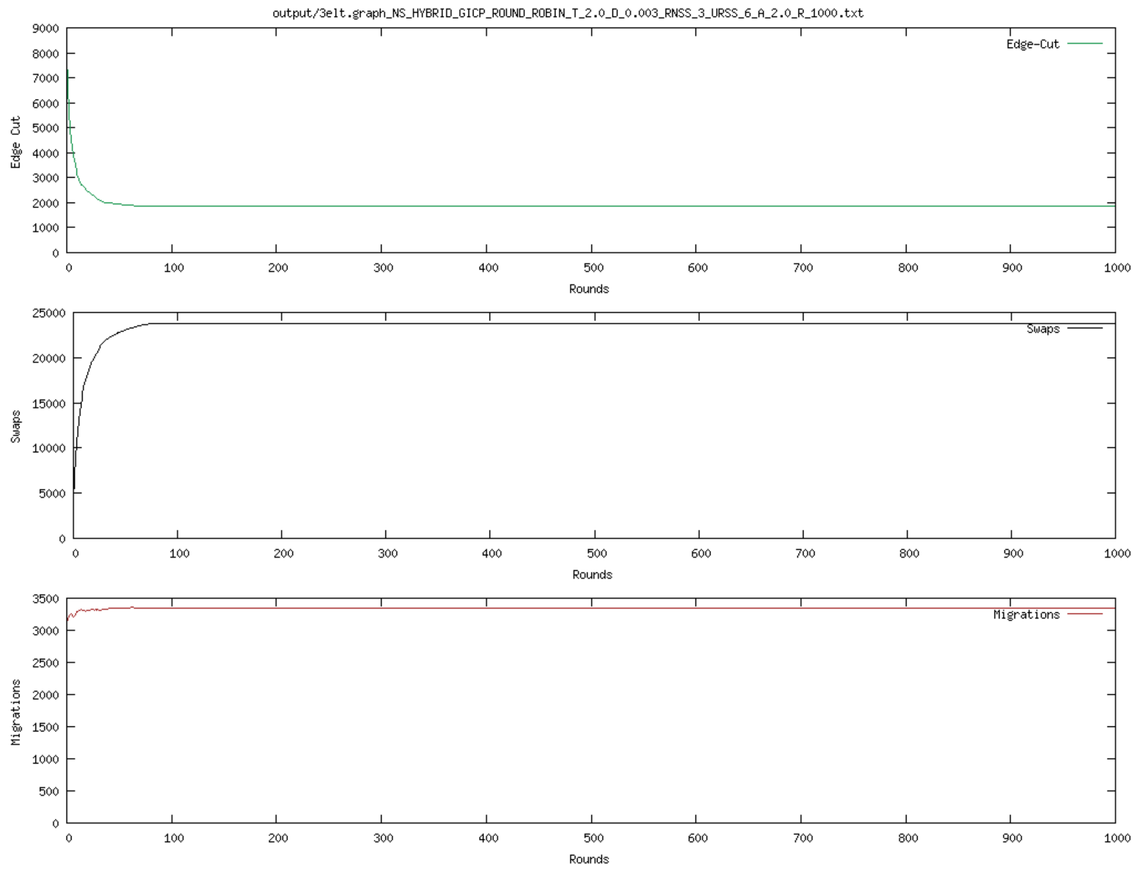

Fig.8. graph-3elt-task2. T = 1; Tmin = 0.00001; alpha = 2; delta = 0.2.

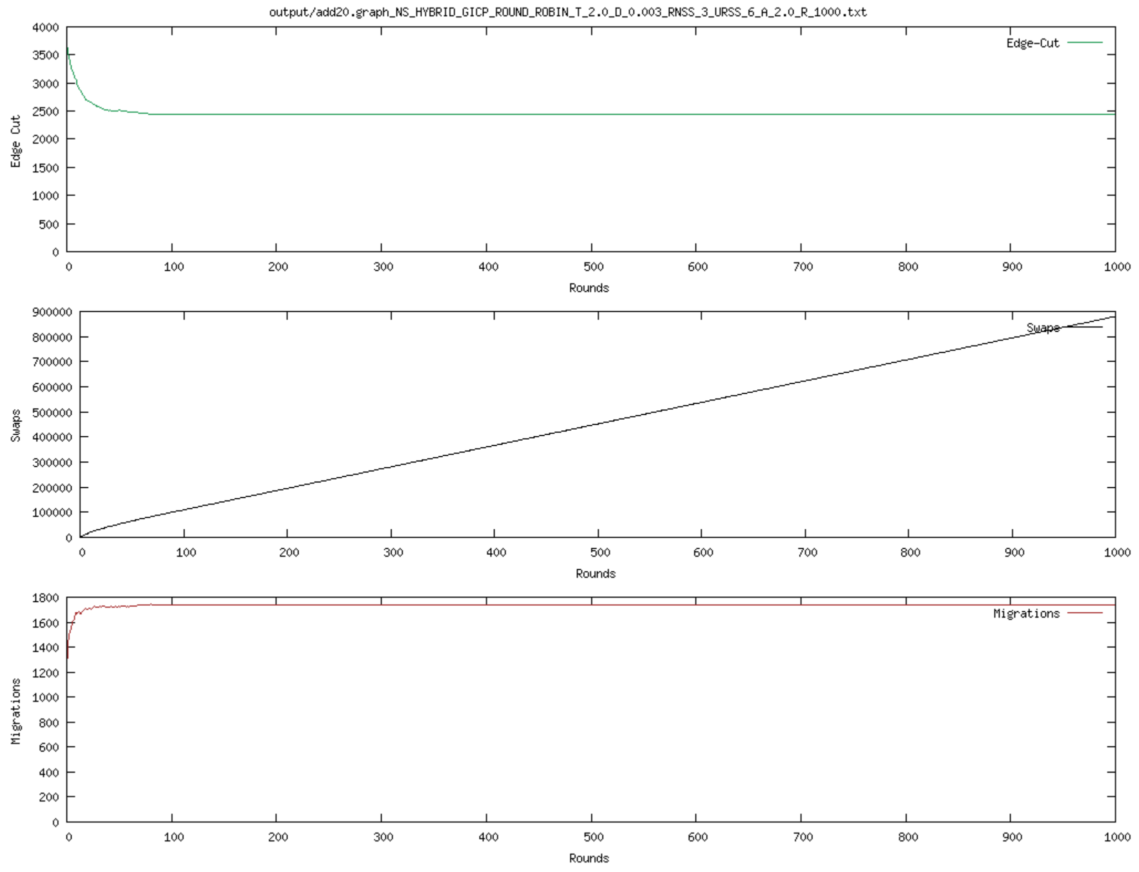

Fig.9. graph-add20-task2. T = 1; Tmin = 0.00001; alpha = 2; delta = 0.9.

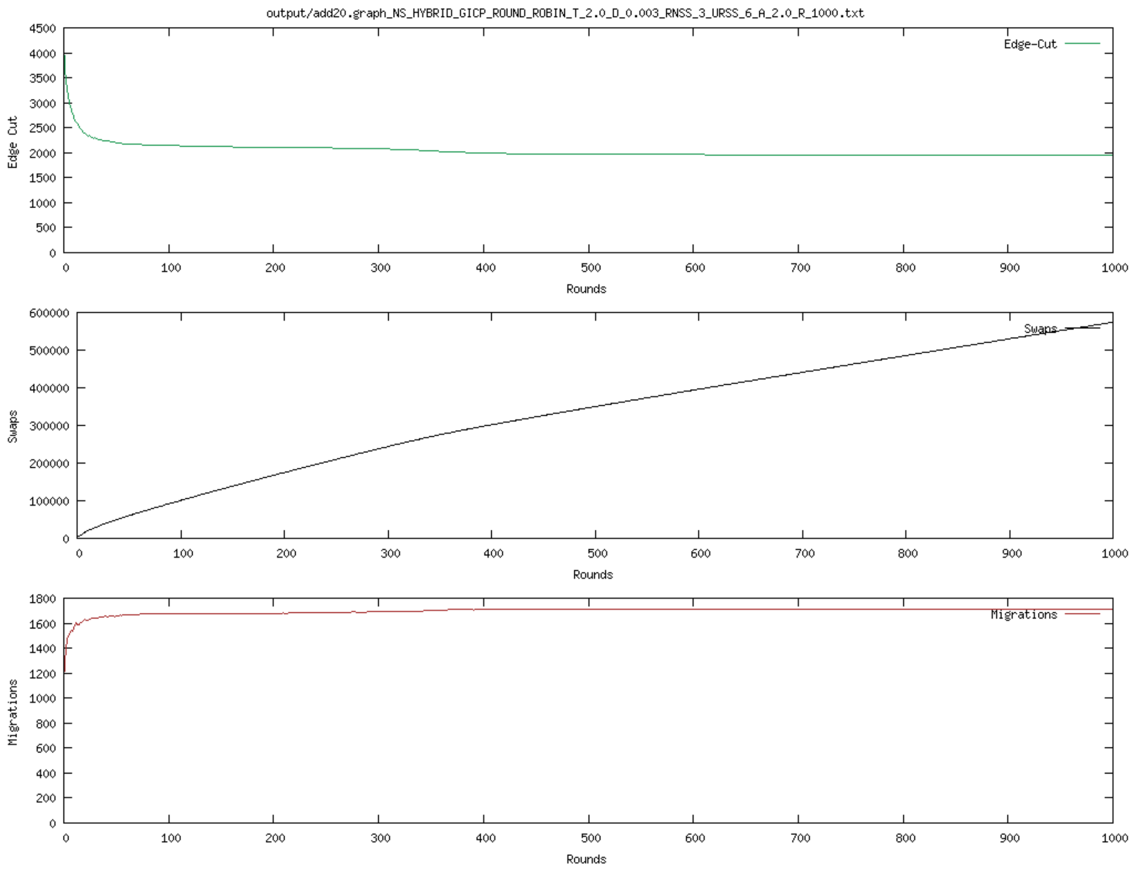

Fig.10. graph-add20-task2. T = 1; Tmin = 0.00001; alpha = 2; delta = 0.8.

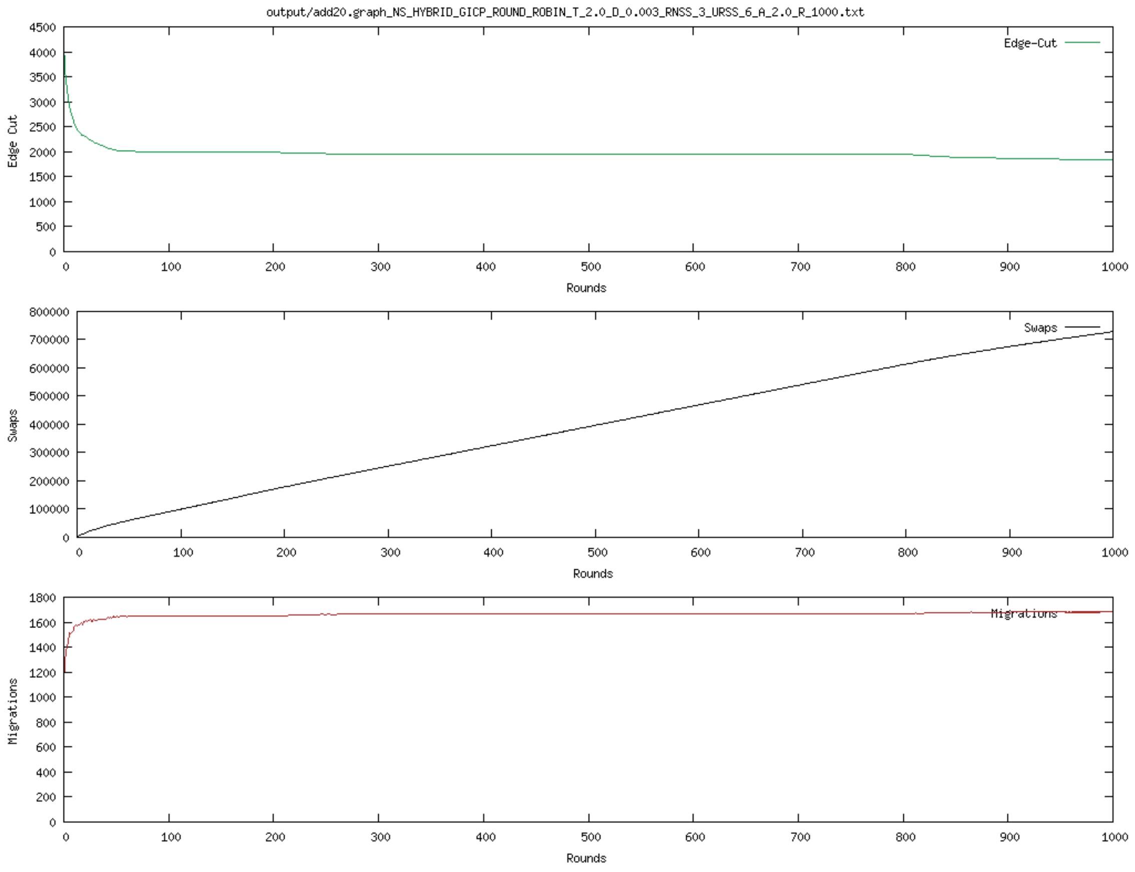

Fig.11. graph-add20-task2. T = 1; Tmin = 0.00001; alpha = ; delta = 0.7.

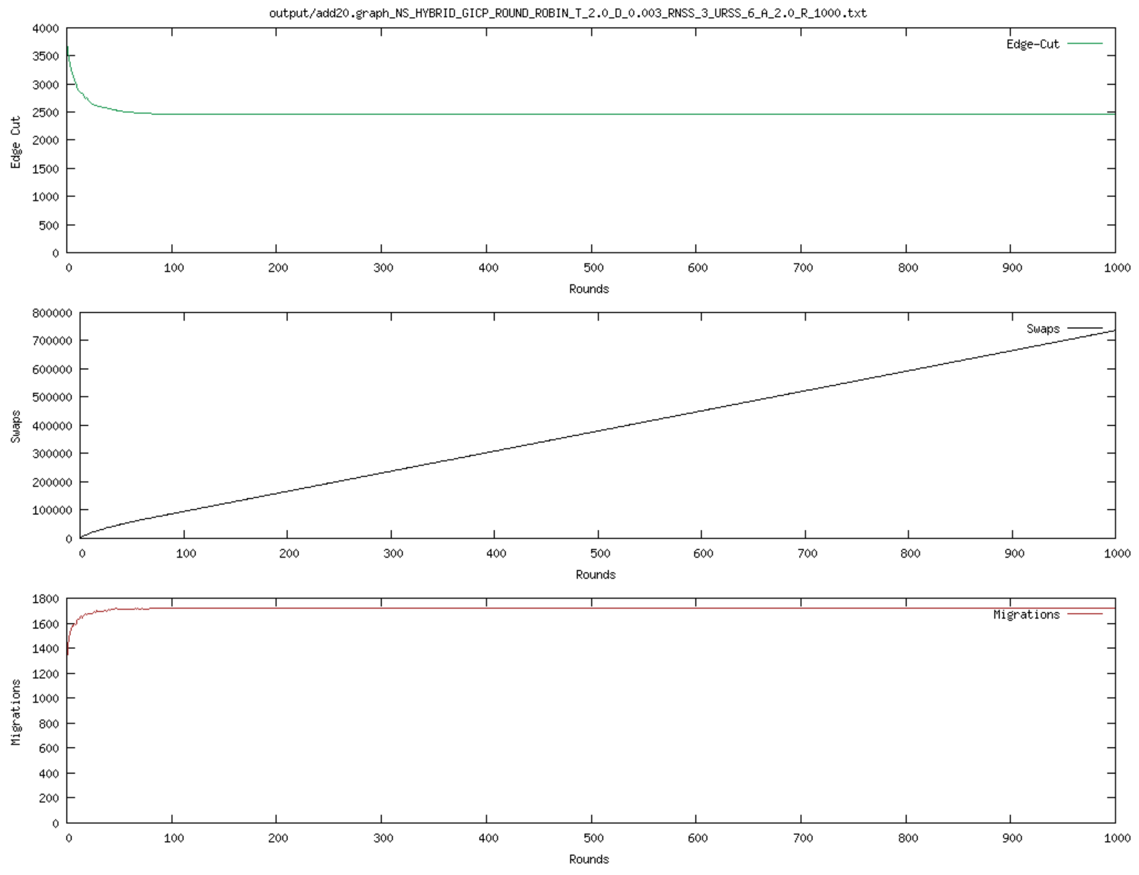

Fig.12. graph-add20-task2. T = 1; Tmin = 0.00001; alpha = 2; delta = 0.1.

Task 2, with reset after 400 round:

Now in this section we did a similar this as previously, except T is reset 400 rounds after it has converged to Tmin.

These results show that for 3elt data this did not affect the edge cut numbers or the number of swaps. Another this that is visible is a slight increase in swaps at around 400 and 800 rounds. This is expected because as T is reset, the probability to make bad swaps increases, thus there should be more swaps observed.

Similarly, for add20 data the number of edge cuts did not change, but the number of swaps increased significantly. Additionally, some jumps in edge cuts can be observed at 400 and 800 rounds that decrease over time to the convergence levels.

Parameters:
T = 1; Tmin = 0.00001; alpha = 2; delta = 0.1; round_converge=1.

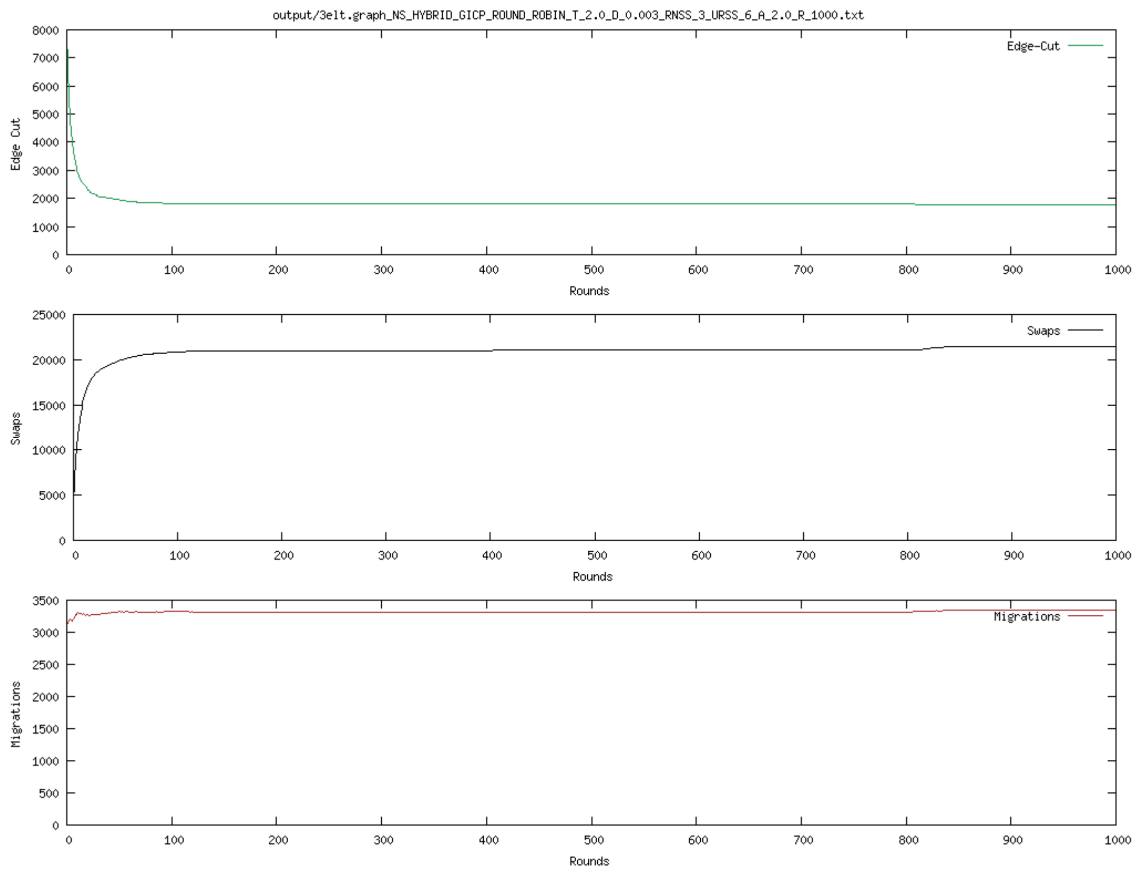

Fig.13. graph-3elt-task2 with reset after 400 round.

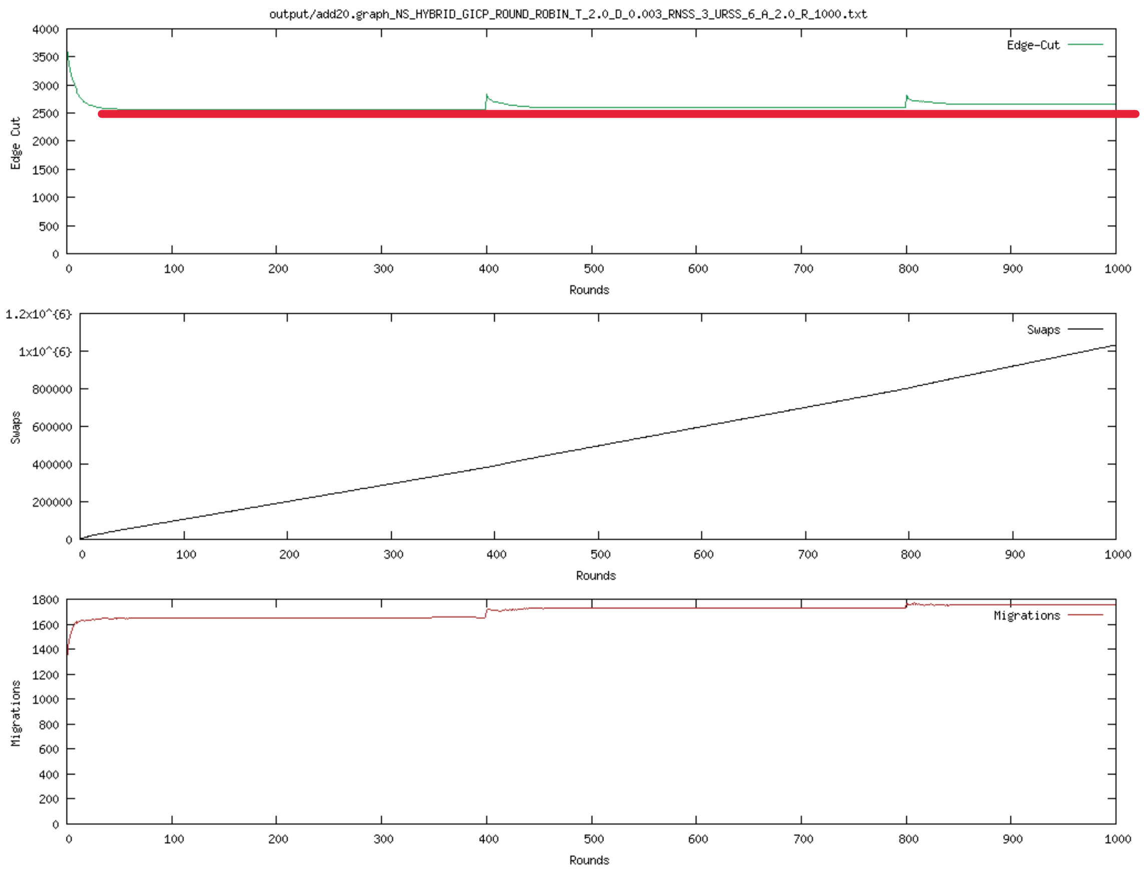

Fig.14. graph-add20-task2 with reset after 400 round.

Extra point:

For the extra point we have implemented our own variation of the acceptance probability. We have defined $ap = \frac{1+T}{1+e^{-(NewScore-OldScore)}}$.
Thus when
T=1 and New>Old score then ap>1;
T=1 and Old>new score then 0<ap<1;
T=0 and New>old score then ap≈1;
T=0 and Old>new score then ap≈0.

Results show that the AP managed to get a better edge cut for 3elt data. The edge cut reached around 1100 which was better than the previous best score of around 1800. The number does not seem to have converged because even at T=0 there is still some chance to make bad decisions. The number of swaps is thus higher as well by about double. For add20 data, the new ap did not seem to get better results. The lack of convergence just increased the complexity of the model by increasing the number of edge swaps.

Parameters:
T = 1; **Tmin** = 0.00001; **alpha** = 2; **round_converge**=1.

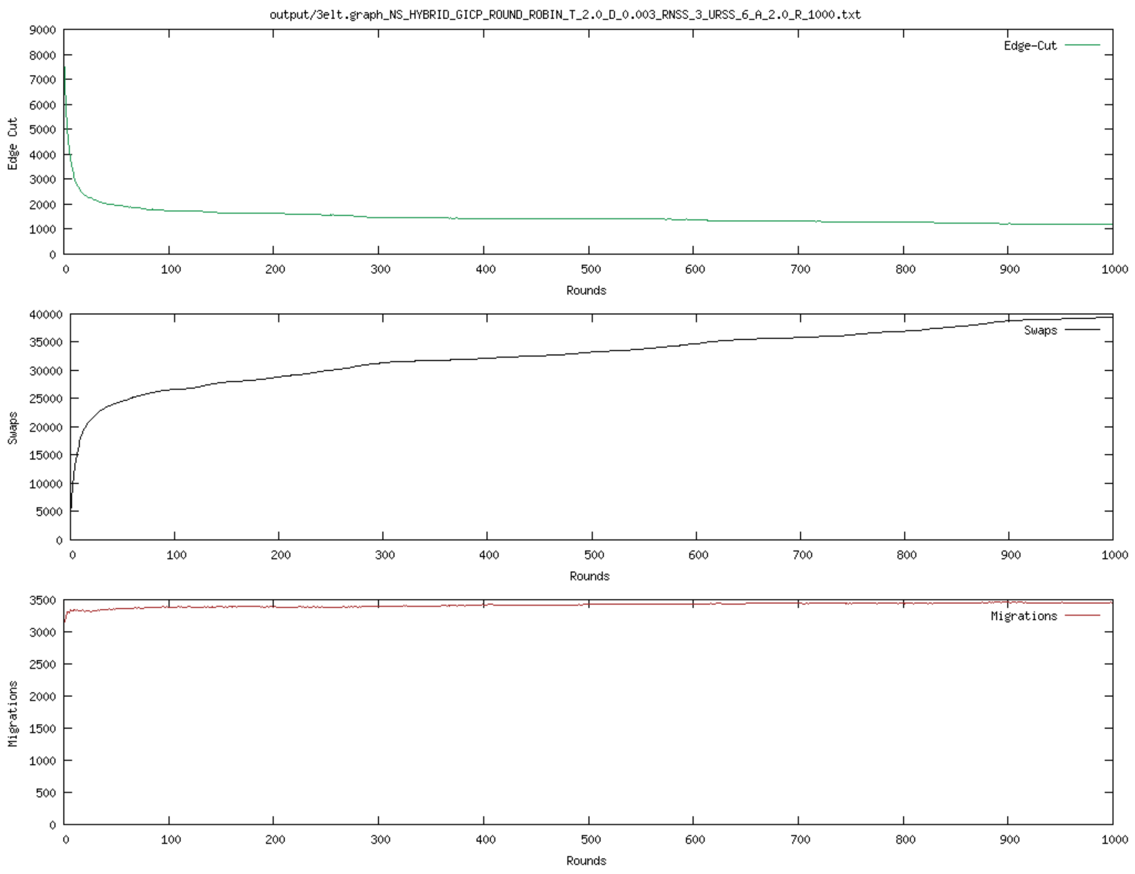

Fig.15. graph 3elt delta =0.9.

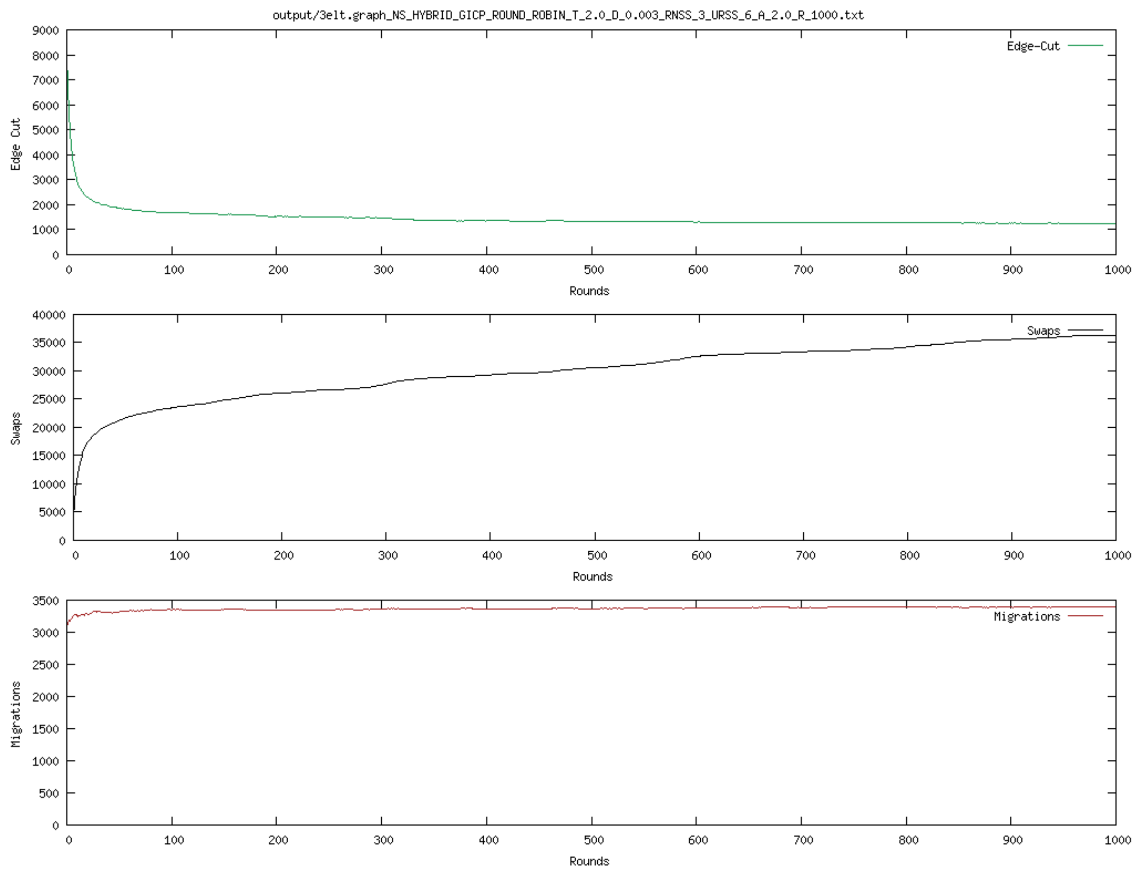

Fig.16. graph-3elt, delta =0.1.

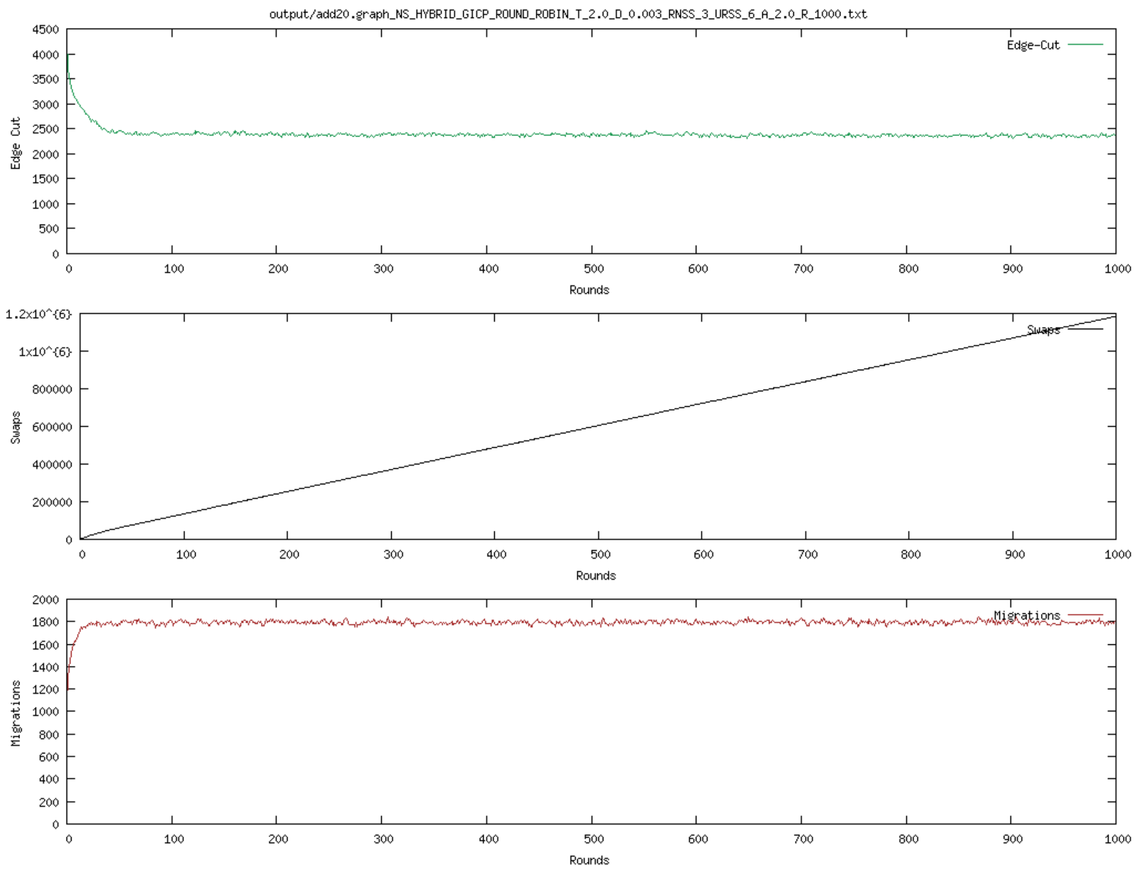

Fig.17. graph-add20, delta =0.9.

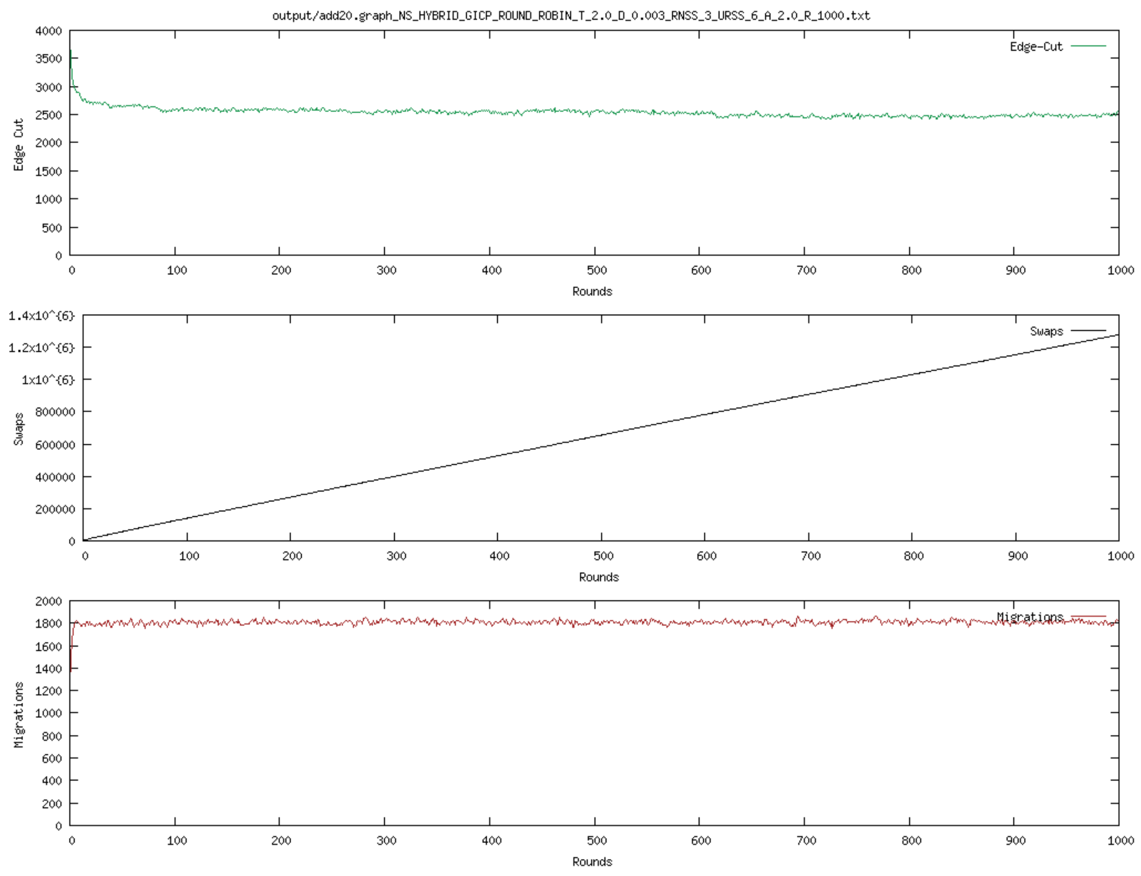

Fig.18. graph-add20, delta =0.1.