Homework 5: K-way Graph Partitioning Using JaBeJa

By Group 16

The goal of this project is to understand distributed graph partitioning using gossip-based peer-to-peer techniques, such as, JaBeJa described in Rahimian, et al. (2013)[1]. This project consists of two main tasks and one bonus task. In the last section it is described how to run the program with the corresponding requirements.

Task 1

In the first task, we implement the Ja-Be-Ja algorithm by modifying the JaBeJa.java class in the provided scaffolding source code for Ja-Be-Ja simulation for one-host-one-node model[2]. In particular the methods sampleAndSwap(...) and findPartner(...) with T being the temperature:

Algorithm 1 JA-BE-JA Algorithm.

Require: Any node p in the graph has the following methods:

- $getNeighbors()$: returns p's neighbors.
- $getSample()$: returns a uniform sample of all the nodes.
- $getDegree(c)$: returns the number of p's neighbors that have color c.

```
//Sample and Swap algorithm at node p
procedure SAMPLEANDSWAP
    partner ← FindPartner(p.getNeighbors(), T_r)
    if partner = null then
        partner ← FindPartner(p.getSample(), T_r)
    end if
    if partner ≠ null then
        color exchange handshake between p and partner
    end if
    T_r ← T_r − δ
    if T_r < 1 then
        T_r ← 1
    end if
end procedure
```

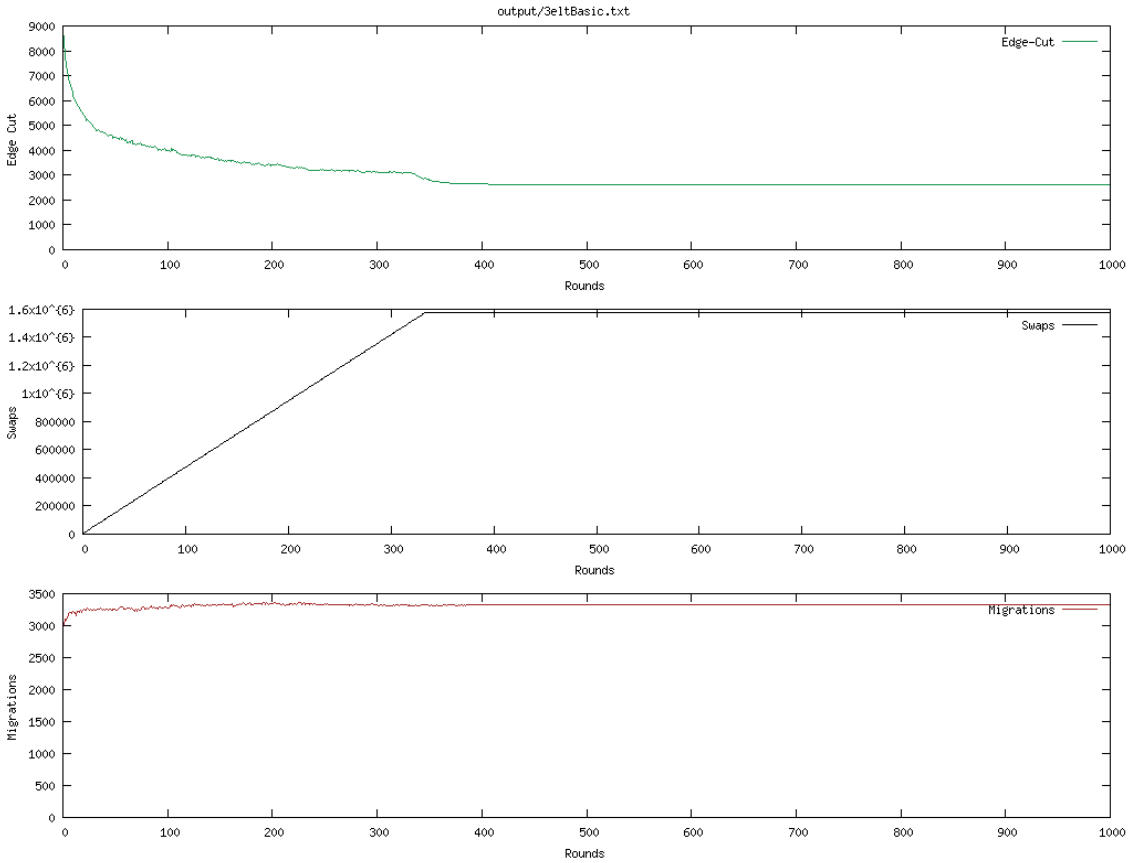
```
//Find the best node as swap partner for node p
function FINDPARTNER(Node[] nodes, float T_r)
    highest ← 0
    bestPartner ← null
    for q ∈ nodes do
        d_pp ← p.getDegree(p.color)
        d_qq ← q.getDegree(q.color)
        old ← d_pp^α + d_qq^α
        d_pq ← p.getDegree(q.color)
        d_qp ← q.getDegree(p.color)
        new ← d_pq^α + d_qp^α
        if (new × T_r > old) ∧ (new > higest) then
            bestPartnere ← q
            highest ← new
        end if
    end for
    return bestPartner
end function
```

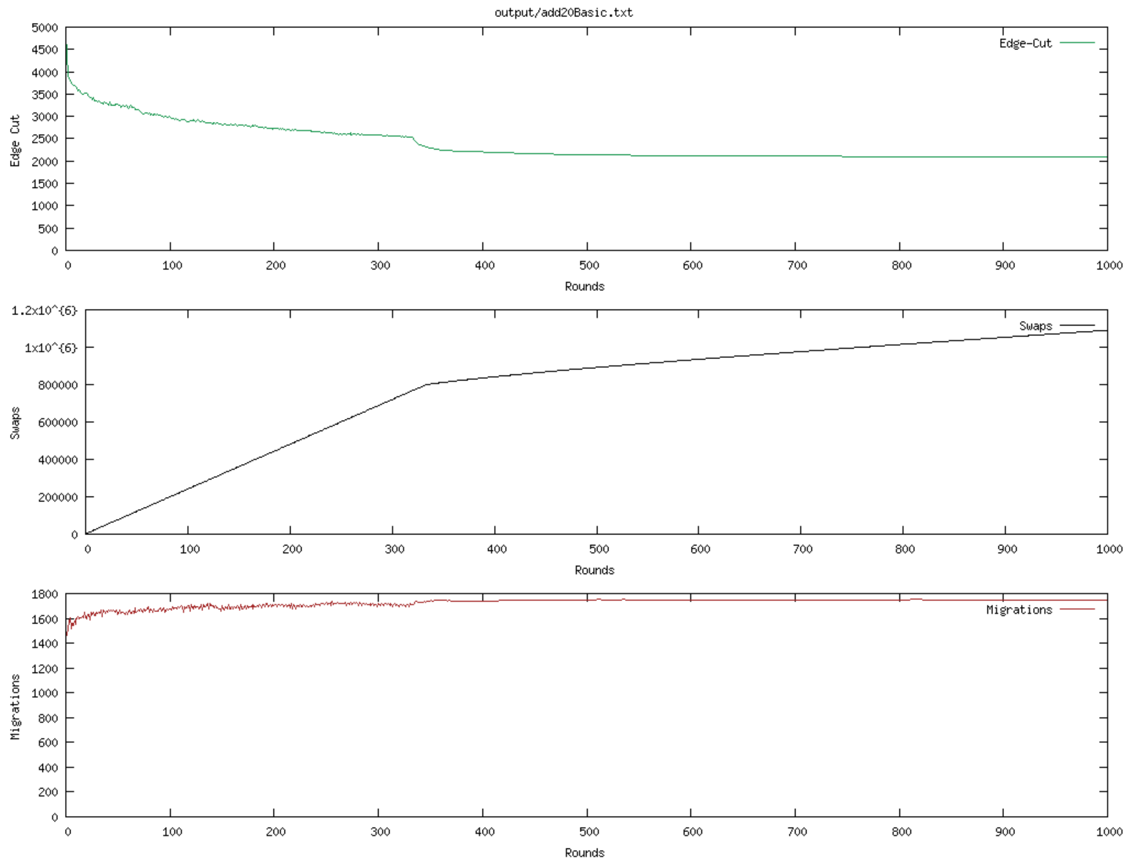
We will focus on the 3elt, add20, Twitter, and Facebook graphs and analyse how changes affect the performance of the algorithm in terms of edge cut observed, number of swaps, and time to converge. The results are depicted below. For all four graphs, the first figure depicts the edge cut, the second figure shows the number of swaps, and the last figure shows the migrations. The edge cut is defined as the number of inter-partition edges. The number of swaps are the swaps that take place between different hosts during run-time, i.e. swaps between graph nodes stored in the same host are not counted. Lastly, data migration is the number of nodes that need to be migrated from their initial partition to their final partition.

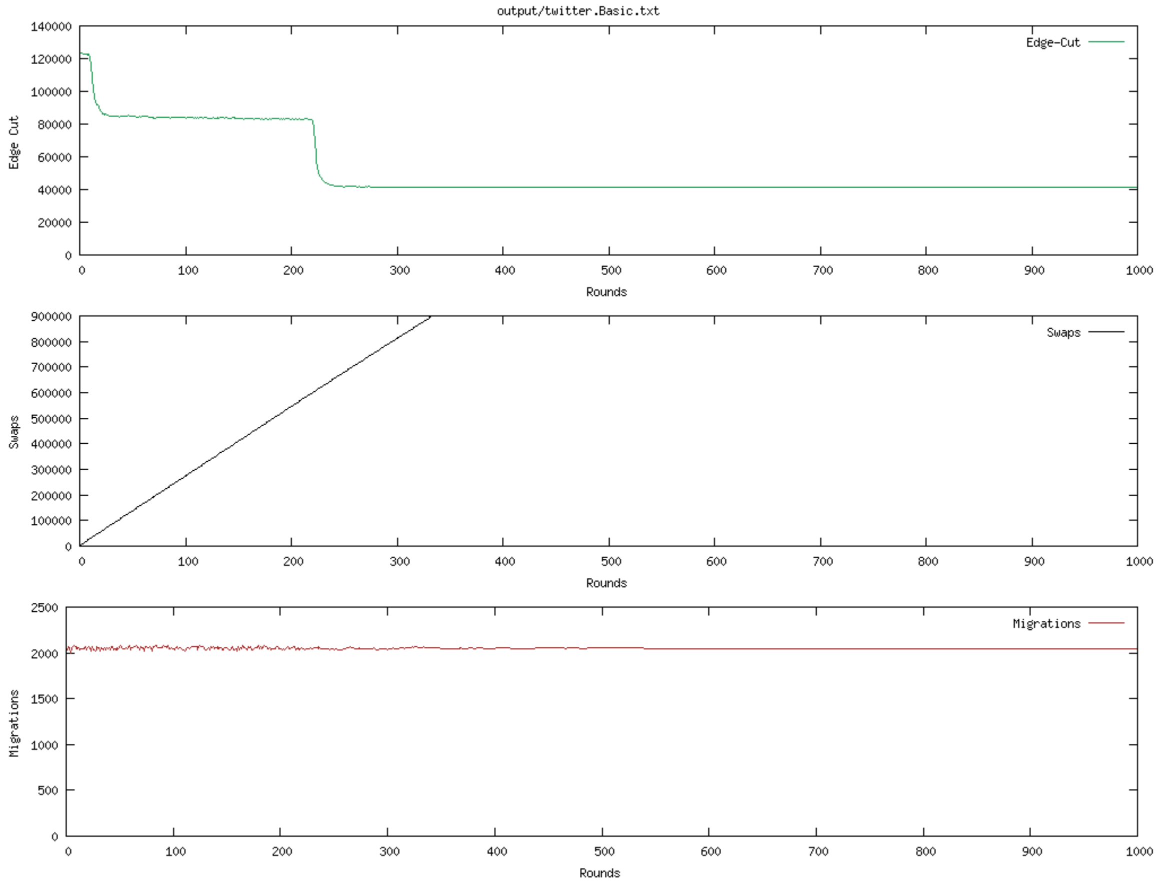
---

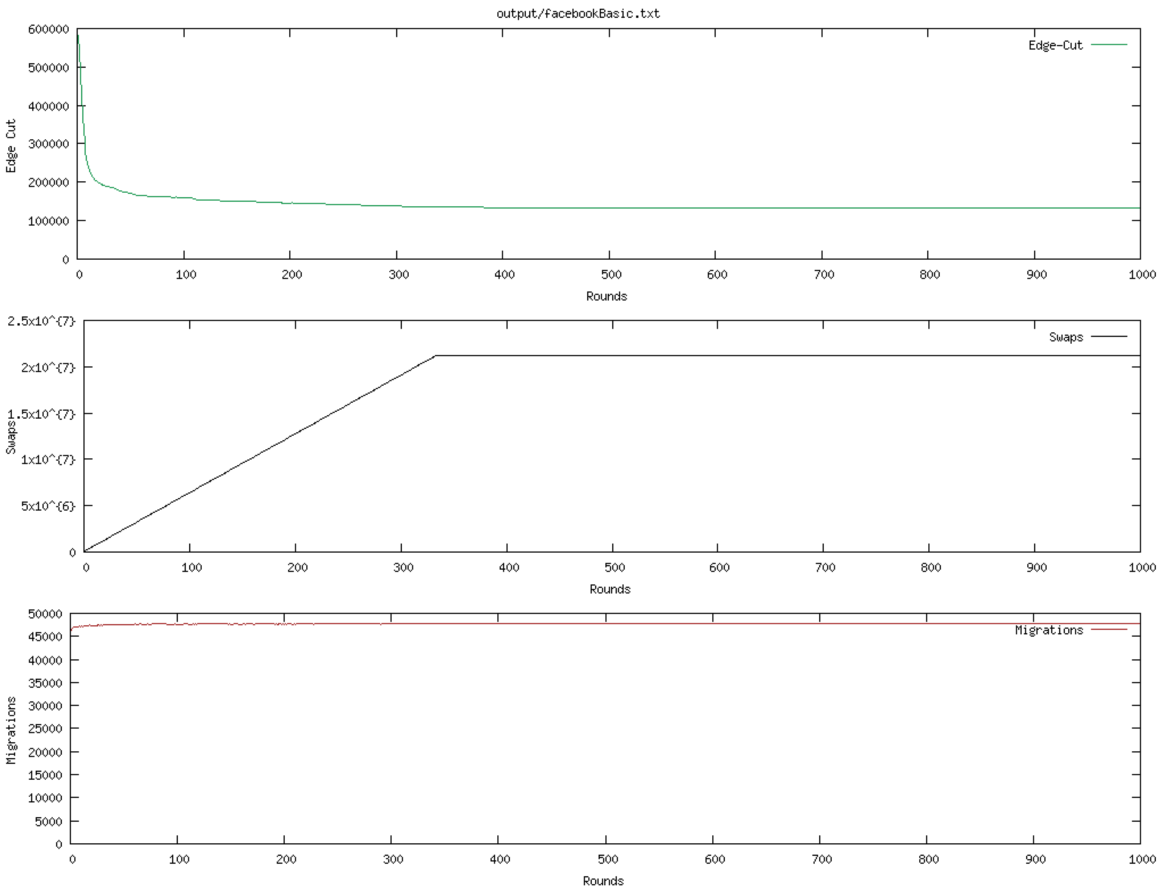
[1] Rahimian, F., Payberah, A. H., Girdzijauskas, S., Jelasity, M., & Haridi, S. (2013, September). Ja-be-ja: A distributed algorithm for balanced graph partitioning. In 2013 IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems (pp. 51-60). IEEE.

[2] https://github.com/smkniazi/id2222


output/3eltBasic.txt
Edge Cut
Edge-Cut
Rounds
Swaps
Migrations


output/add20Basic.txt
Edge Cut
Edge-Cut
Rounds
Swaps
Migrations


output/twitter.Basic.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds


output/facebookBasic.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

Regarding the edge cut, it is noticed that the more rounds, the smaller the edge cut. The number of swaps, however, increases linearly with the number of rounds. The migrations seem to be consistent over the number of rounds. This can be explained by the fact that Ja-Be-Ja uses a linear function to decrease the temperature. The values are depicted in Table 1 for a more clear comparison and where we will discuss it into more detail.

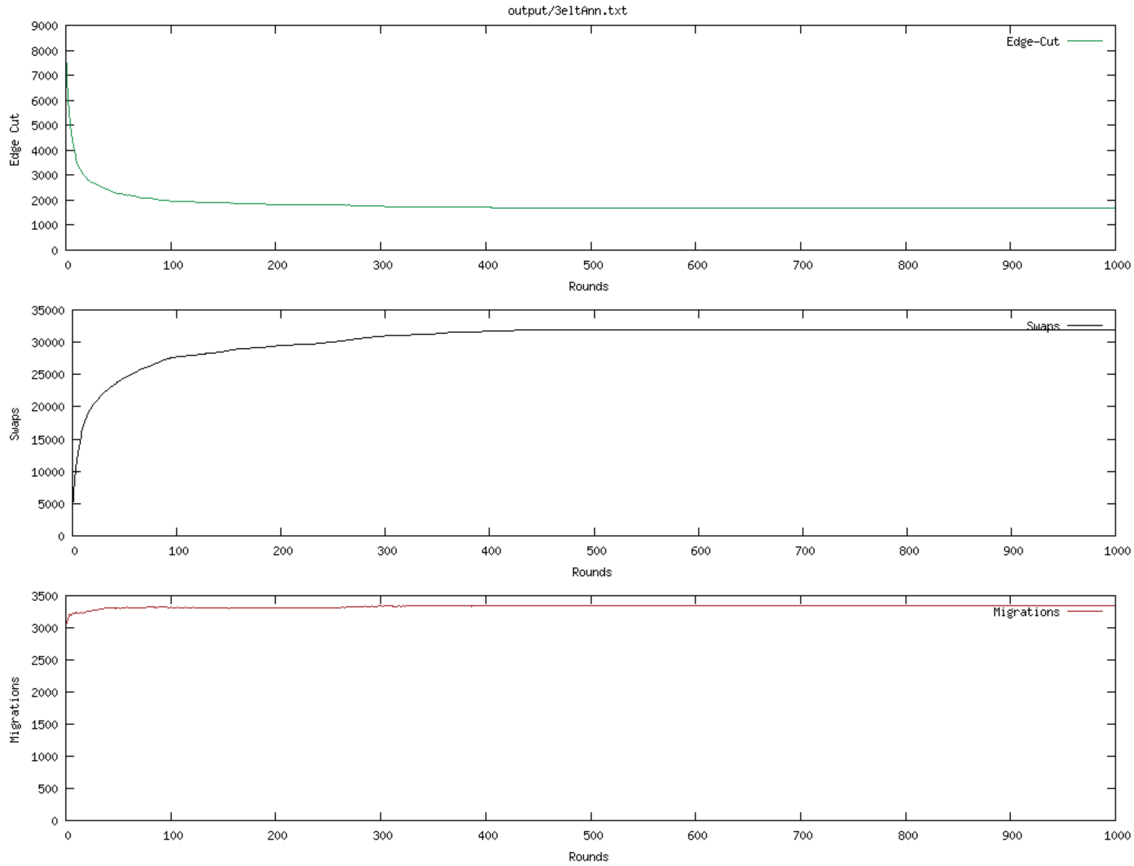
Task 2

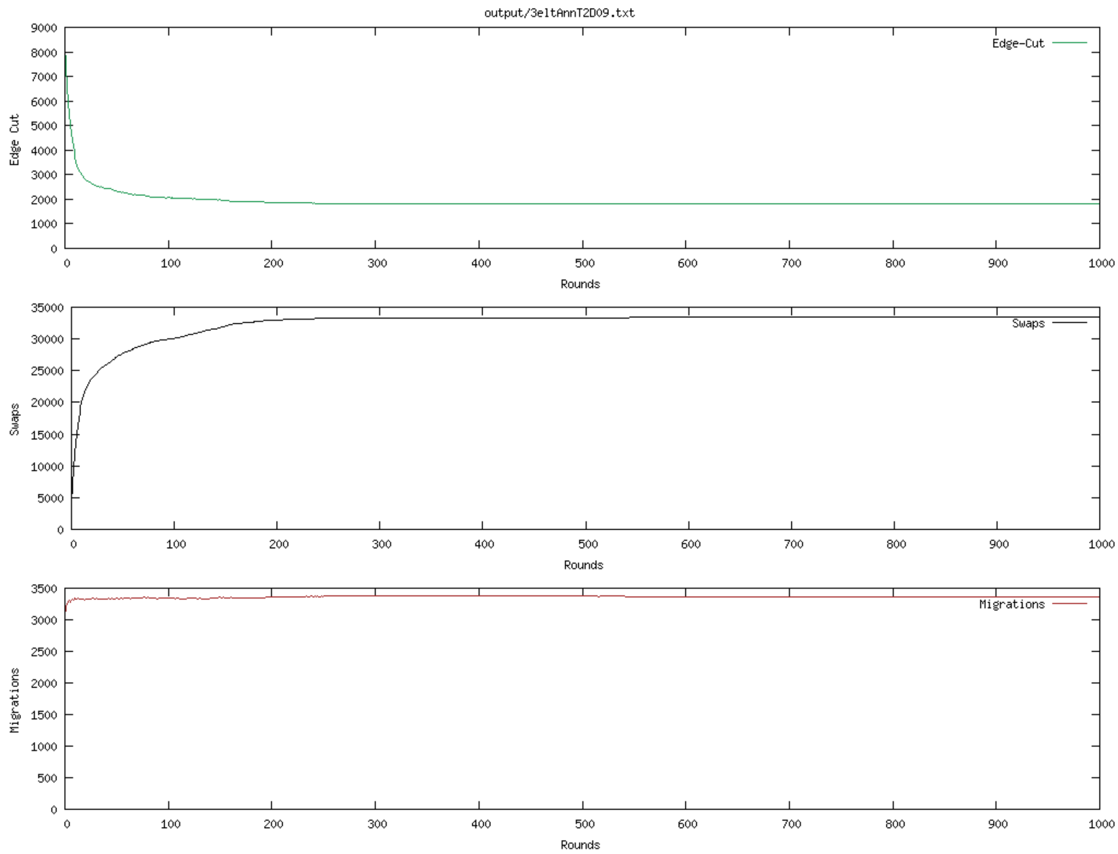
In the second task we tweaked different JaBeJa configurations in order to find the smallest edge cuts for the given graphs. We analyzed how the performance of the algorithm is affected when different parameters are changed, specially the effect of simulated annealing and the acceptance probability function. A different simulated mechanism is implemented based on Geltman[3], namely:

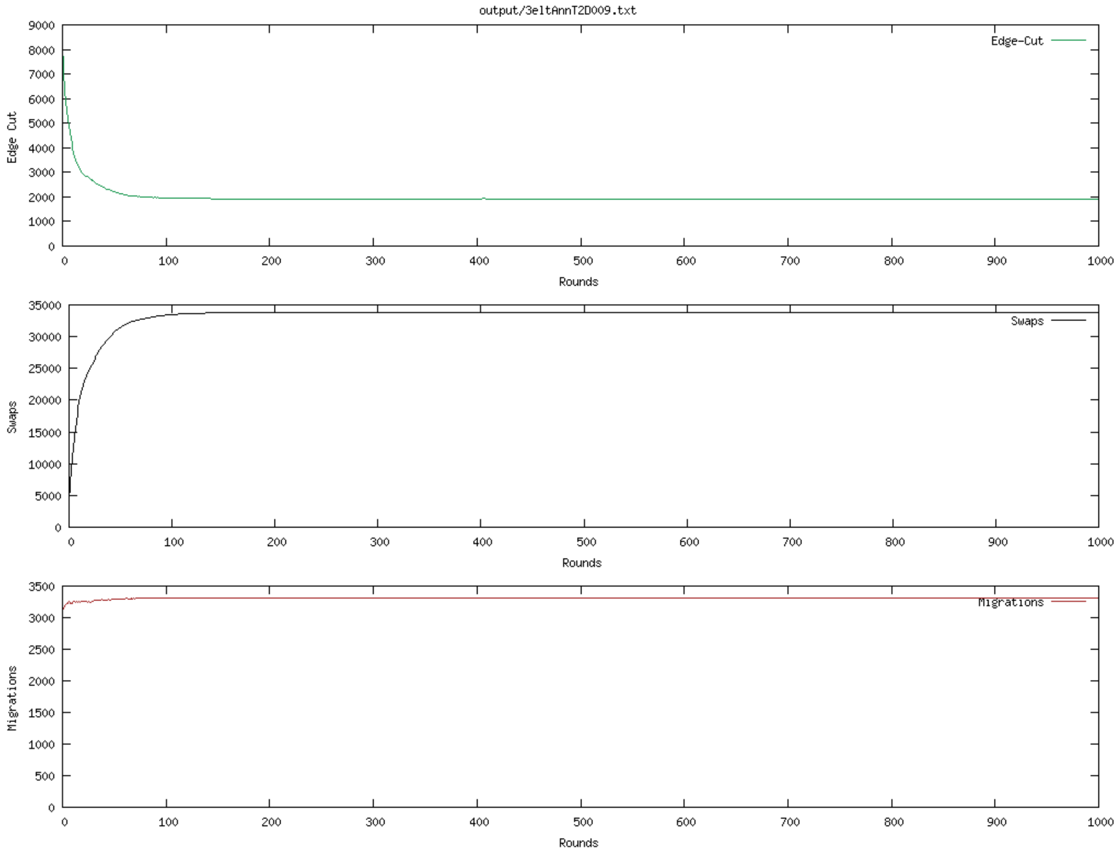
$$a = e^{\frac{c_{new} - c_{old}}{T}}$$

where a is the acceptance probability, $c_{new} - c_{old}$ is the difference between the new cost and the old cost, T is the temperature, and e is 2.71828…From this, we can see that the smaller a as the new cost is lower than the old score. Also when the temperature T increases, “bad jumps” are penalised i.e. a higher acceptance probability. It is noticed that once the parameter T reaches its final value (that is, no more bad swaps are allowed) then Ja-Be-Ja converges to an edge cut rapidly and the edge cut does not change over time. We therefore restart simulated-annealing again after 400 rounds. Observed is how this change affects the rate of convergence. The values from this experiment are also depicted in Table 1.

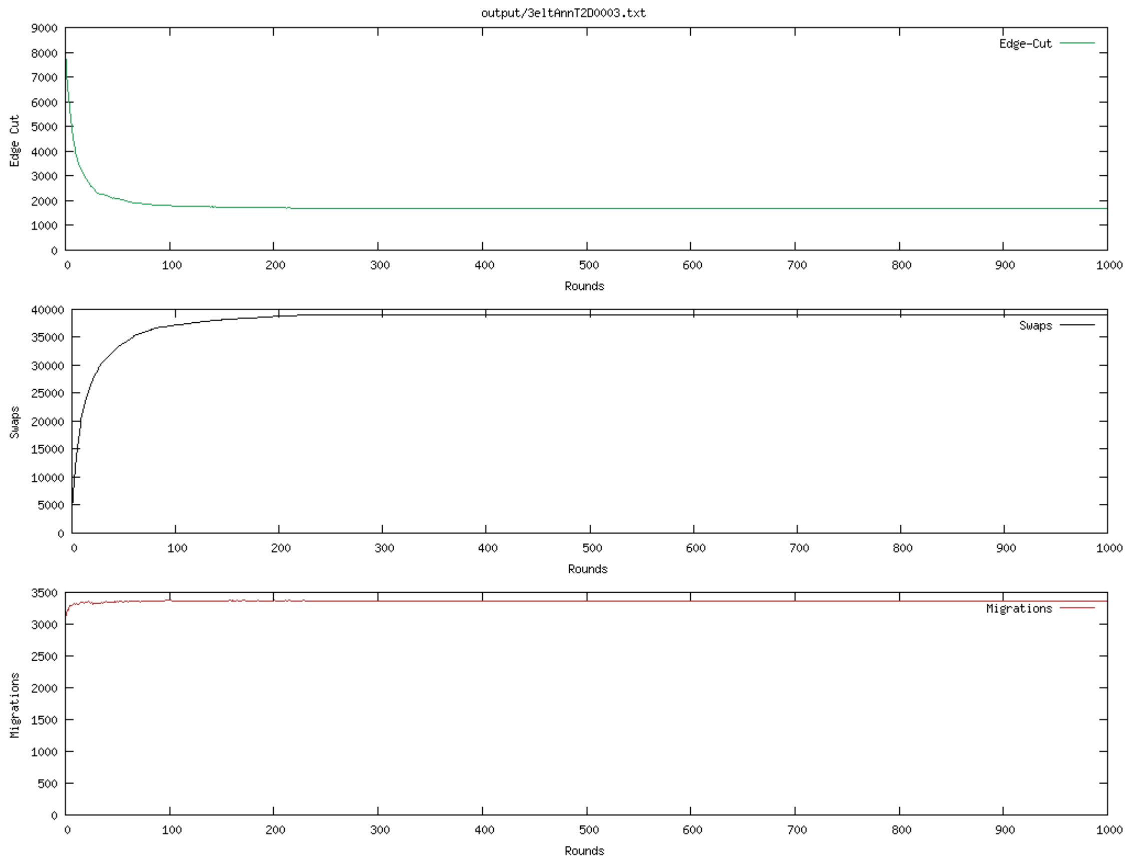
[3] http://katrinaeg.com/simulated-annealing.html


output/3eltAnn.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

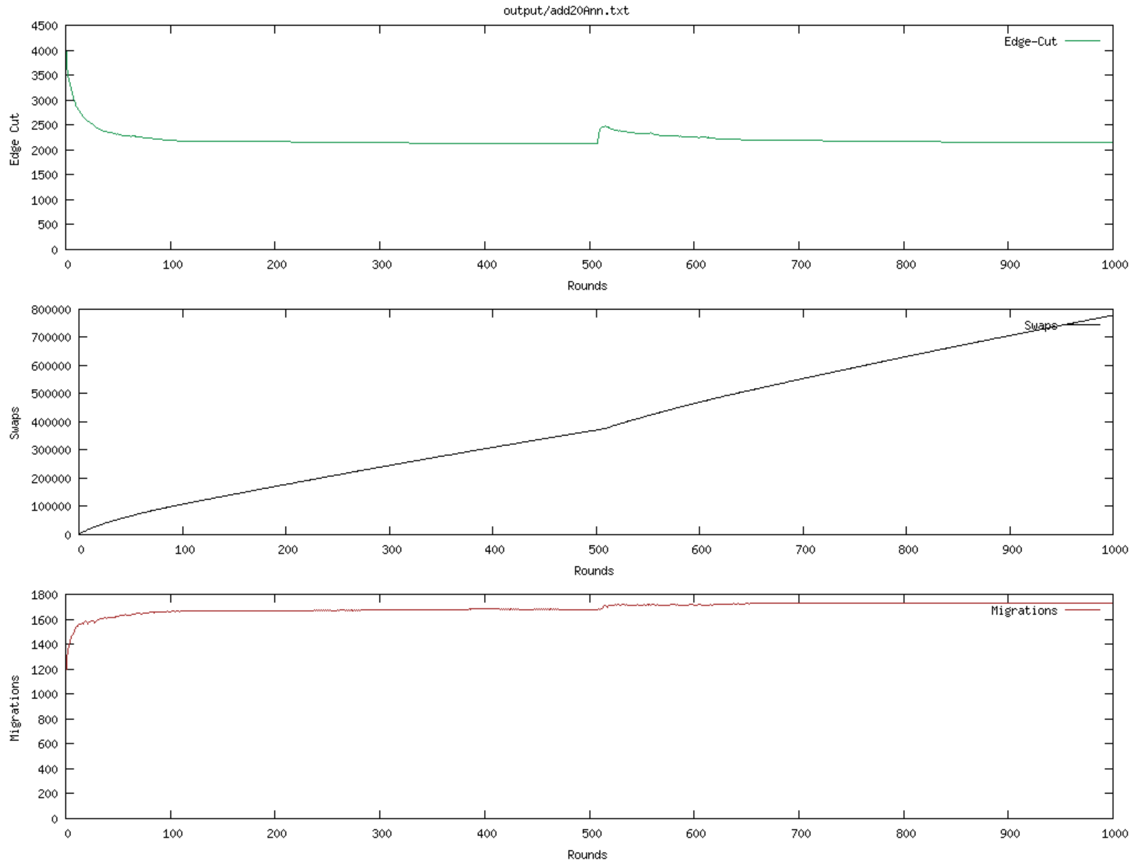

output/3eltAnnT2D09.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

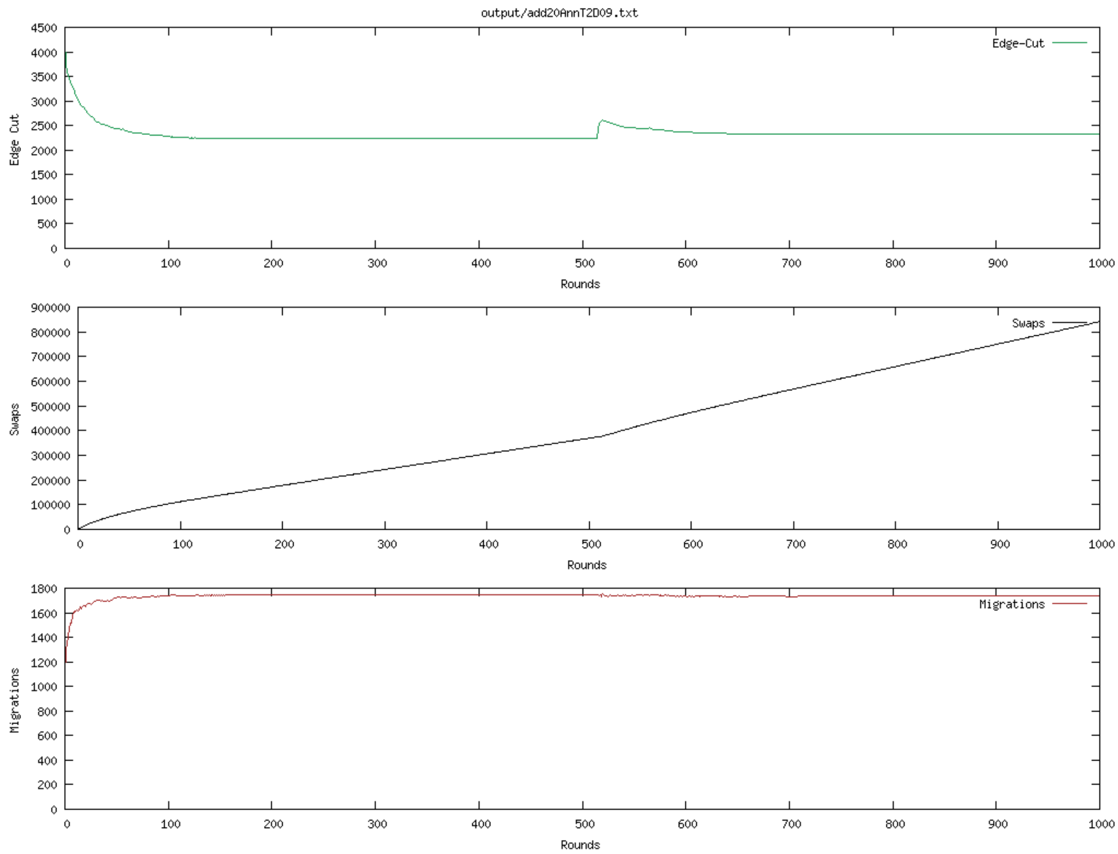


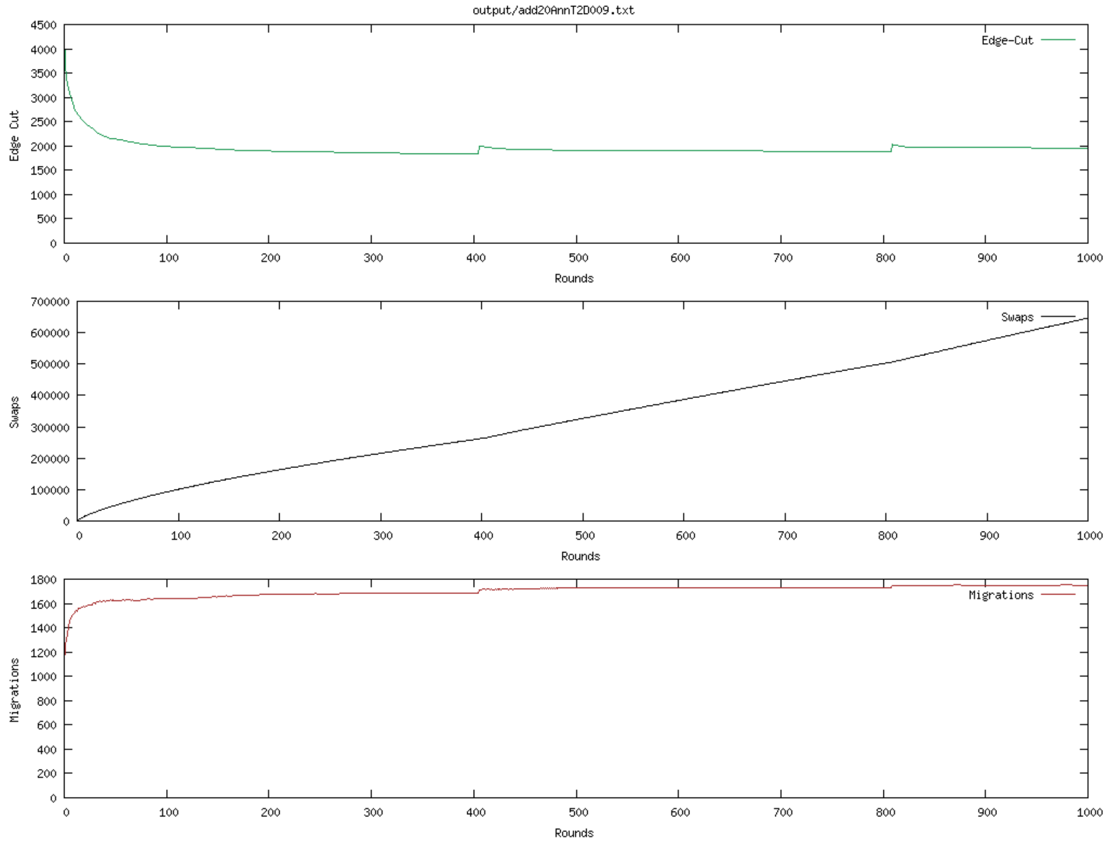
+

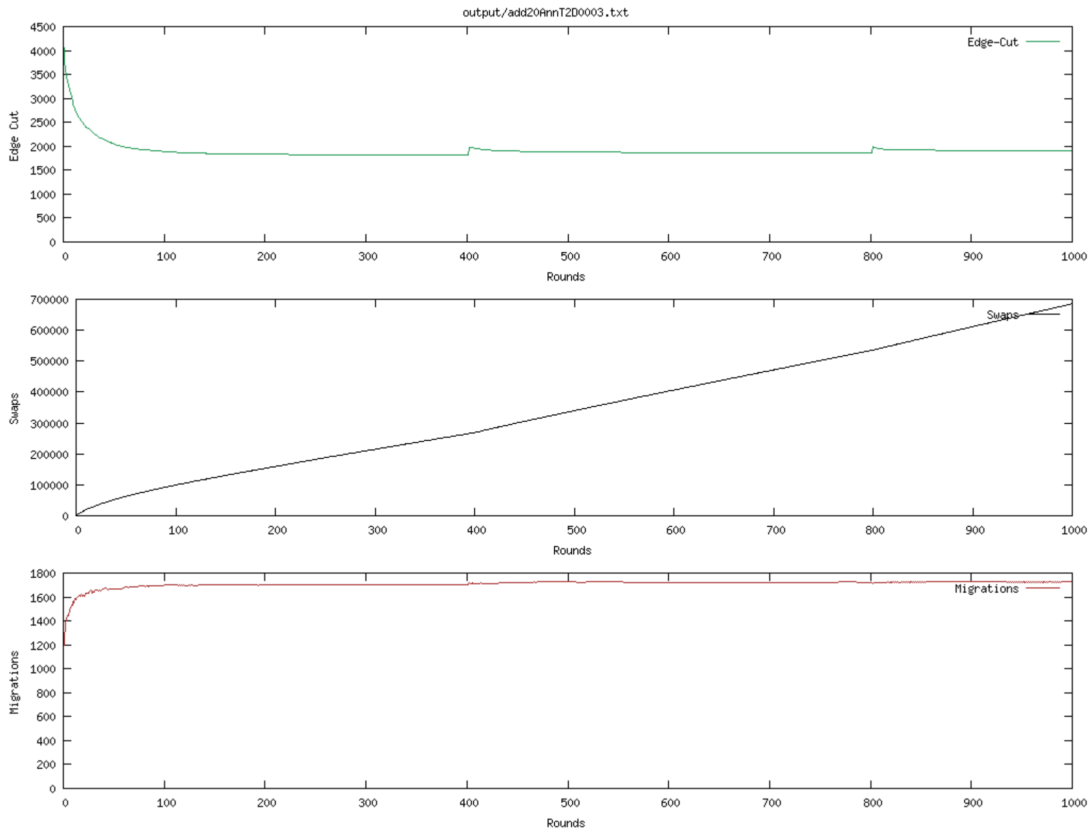


3elt with annealing and (1) T=1, delta=0.9 (2) T=2, delta = 0.9 (3) T=2, delta = 0.09 (4) T=2, delta = 0.003

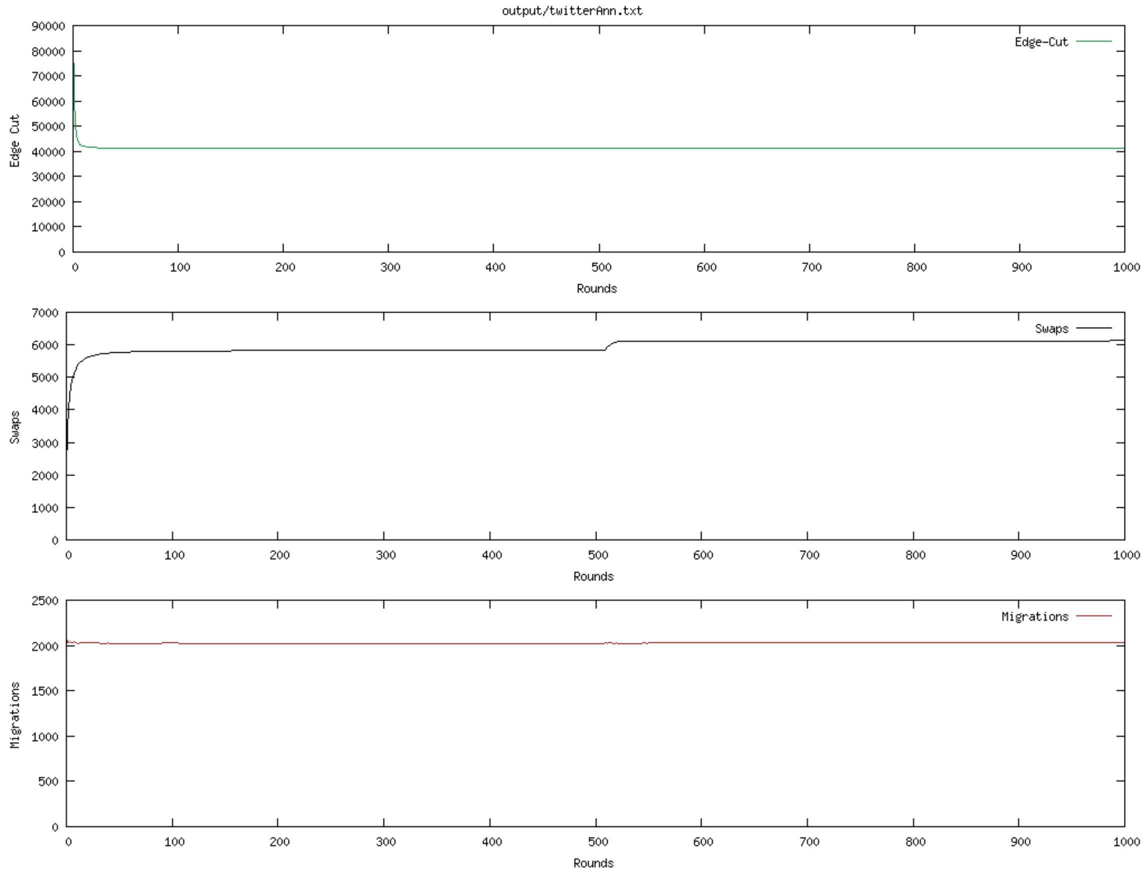

output/add20Ann.txt
Edge Cut
Edge-Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

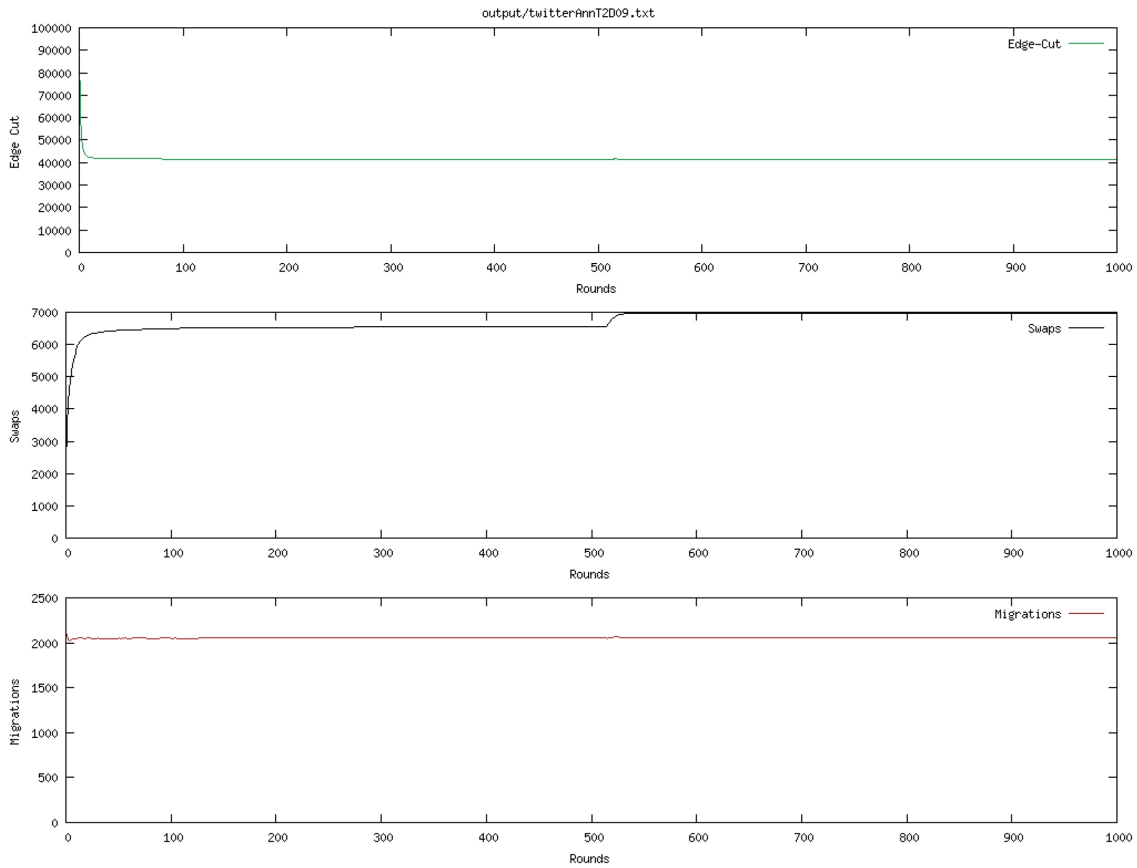

output/add20AnnT2D09.txt
Edge Cut
Edge-Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

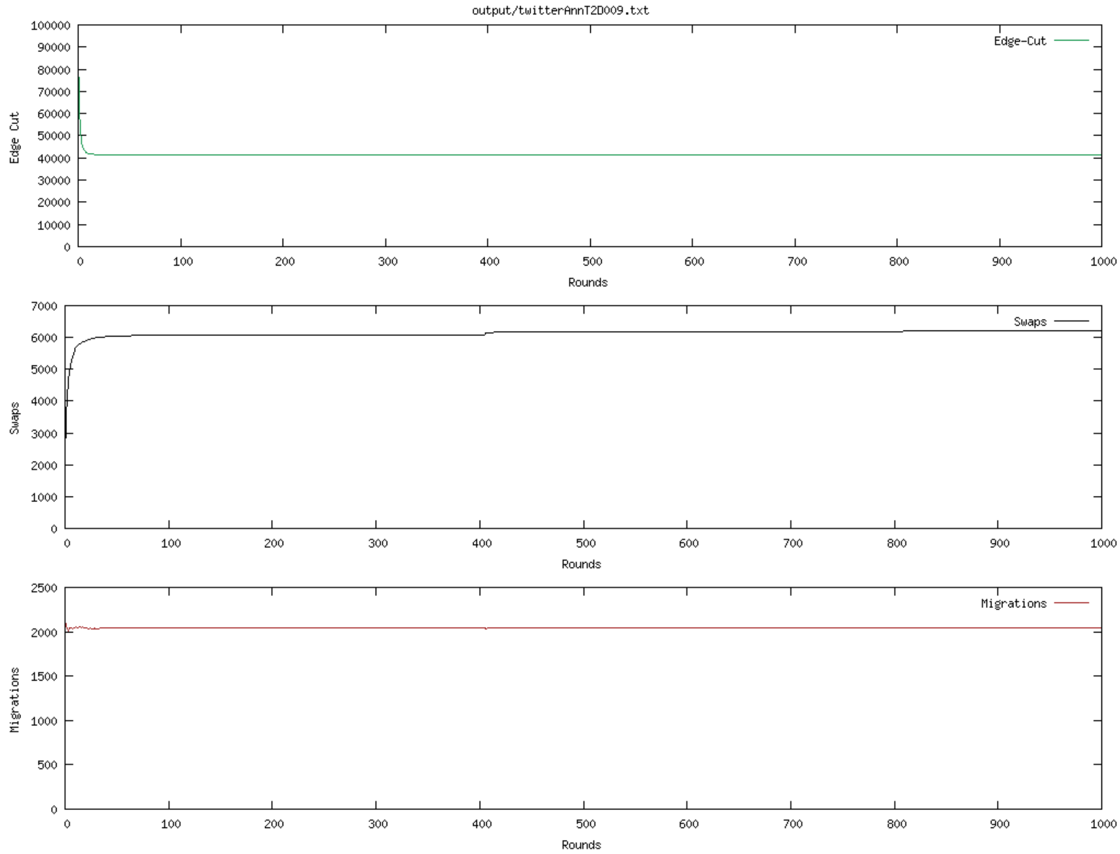


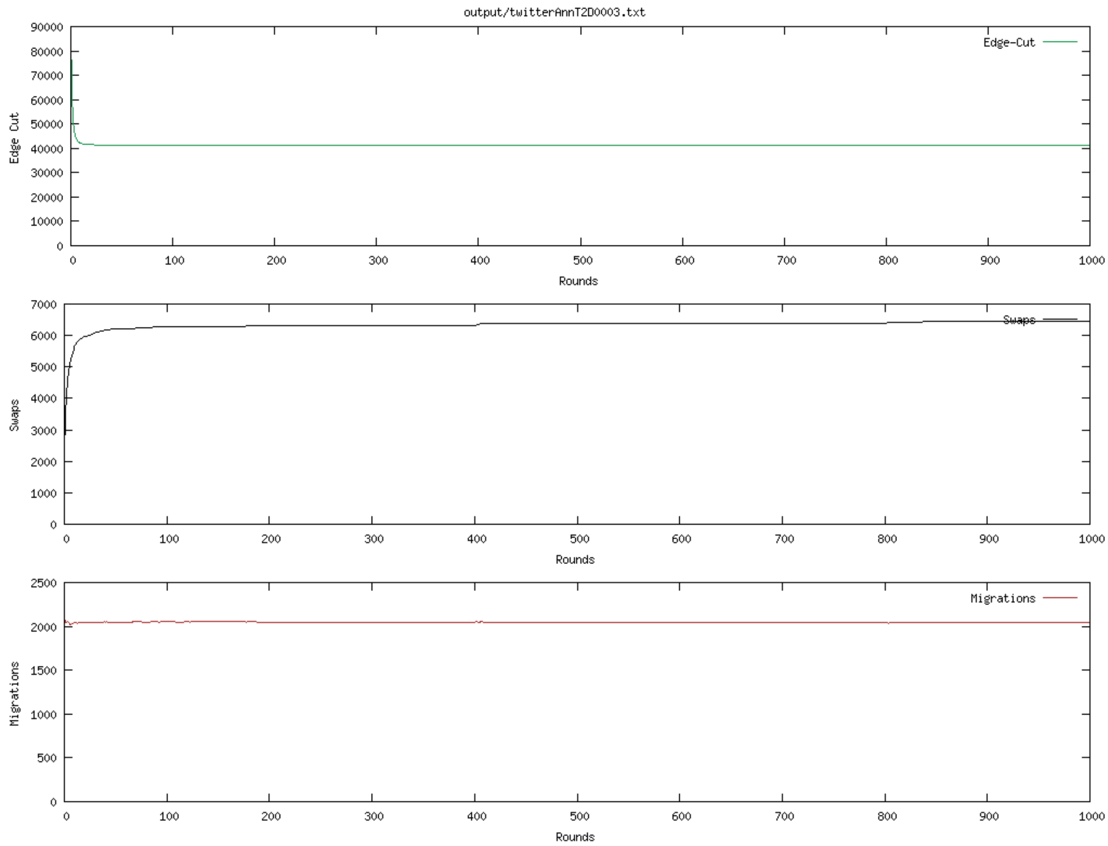


add20 with annealing and (1) T=1, delta=0.9 (2) T=2, delta = 0.9 (3) T=2, delta = 0.09 (4) T=2, delta = 0.003

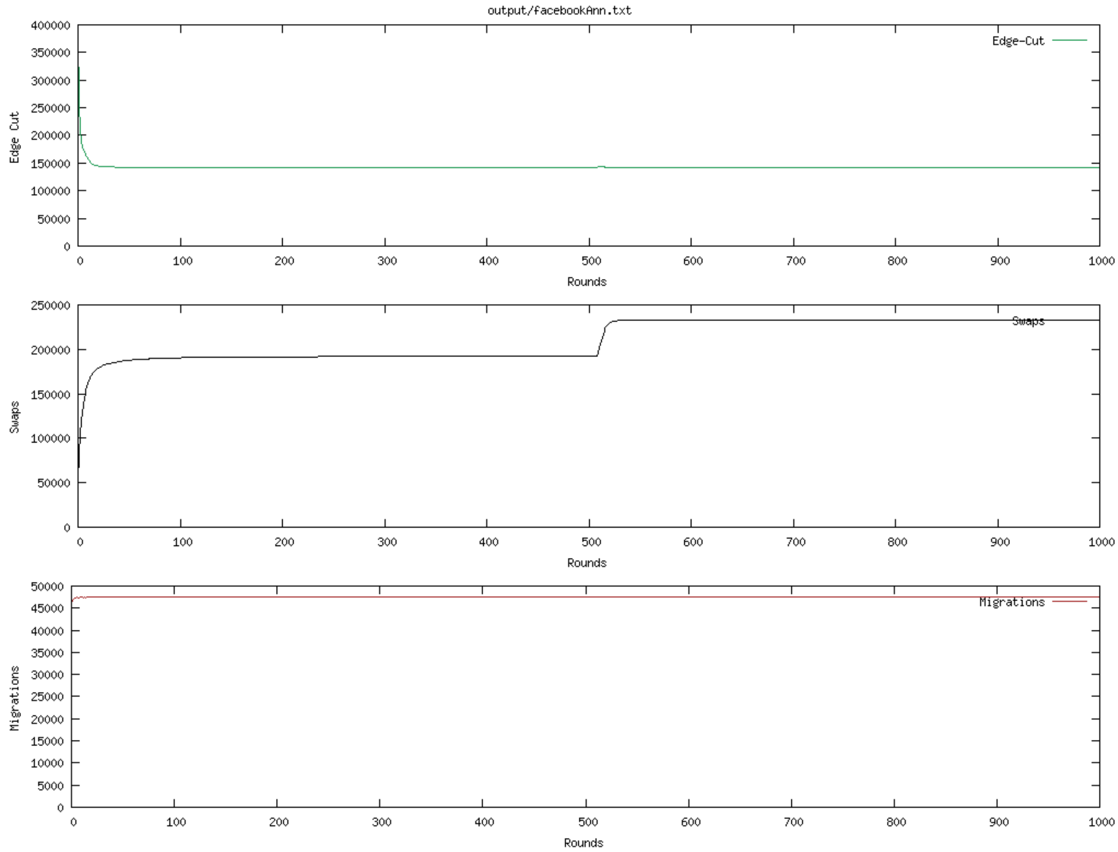

output/twitterAnn.txt
Edge Cut
Edge-Cut
Rounds
Swaps
Rounds
Migrations
Rounds

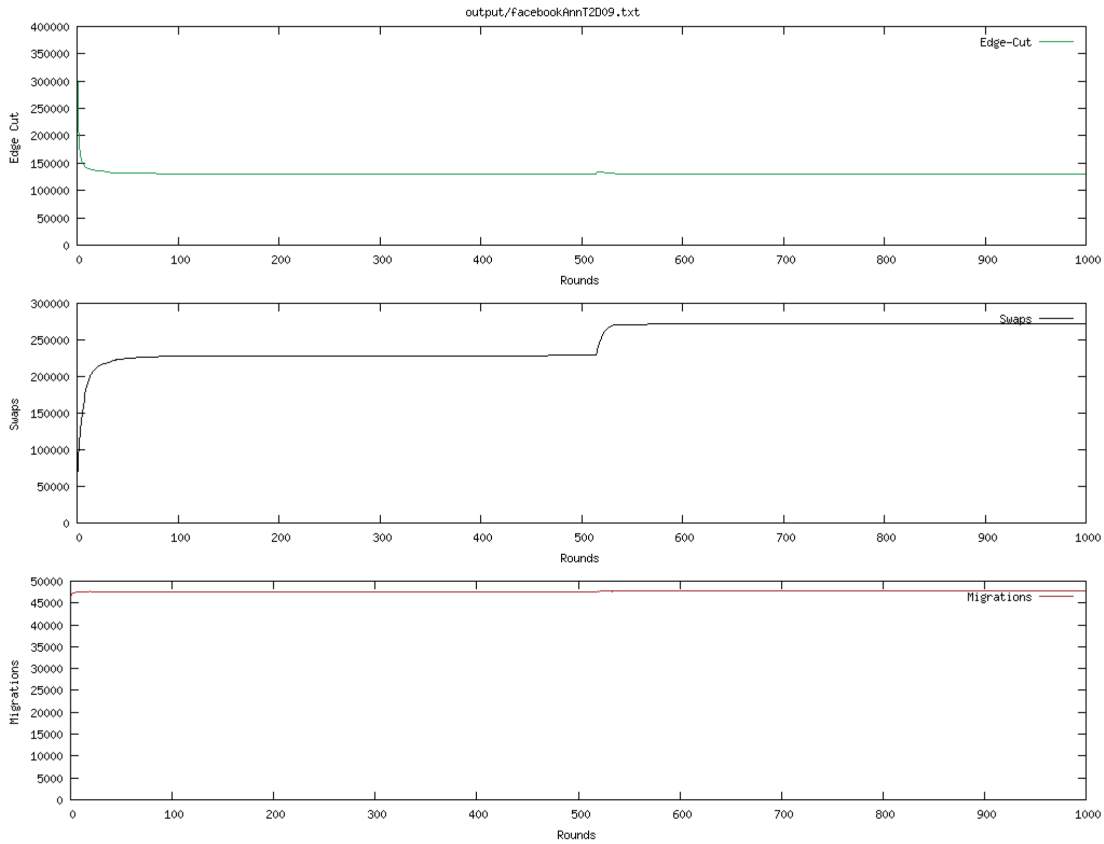

output/twitterAnnT2D09.txt
Edge Cut
Edge-Cut
Rounds
Swaps
Rounds
Migrations
Rounds

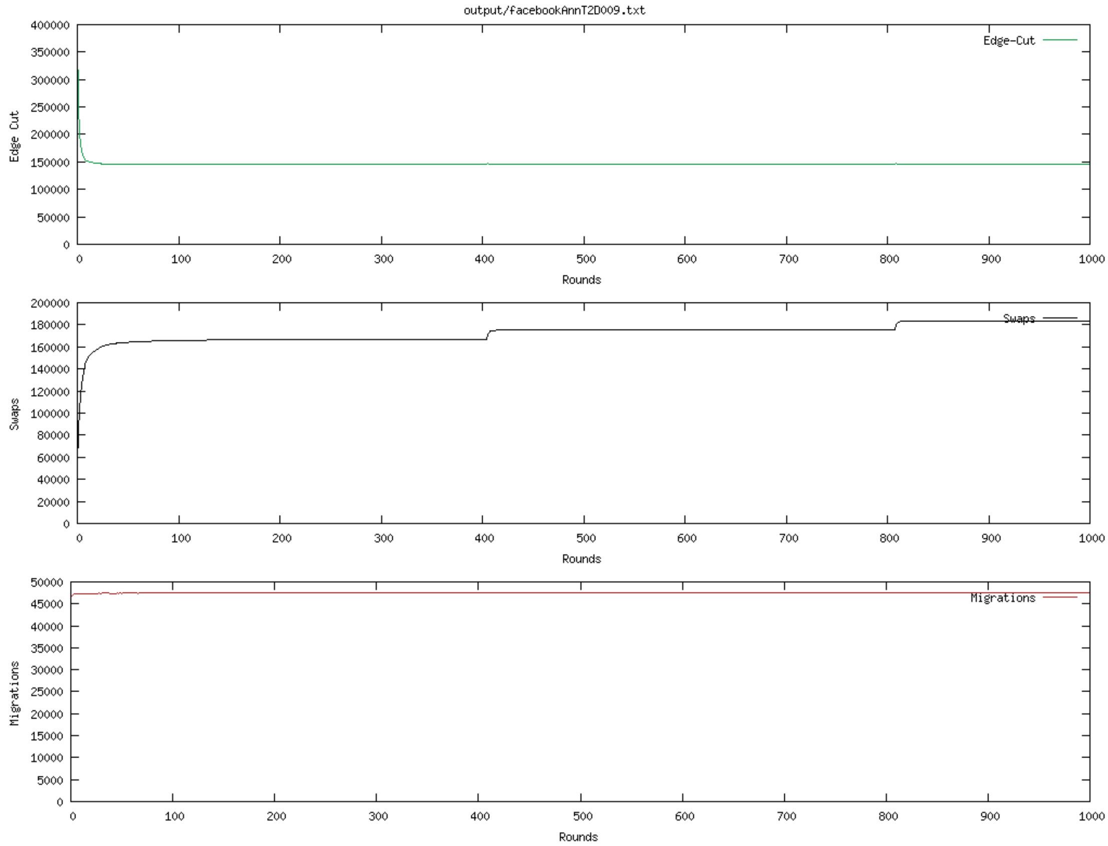


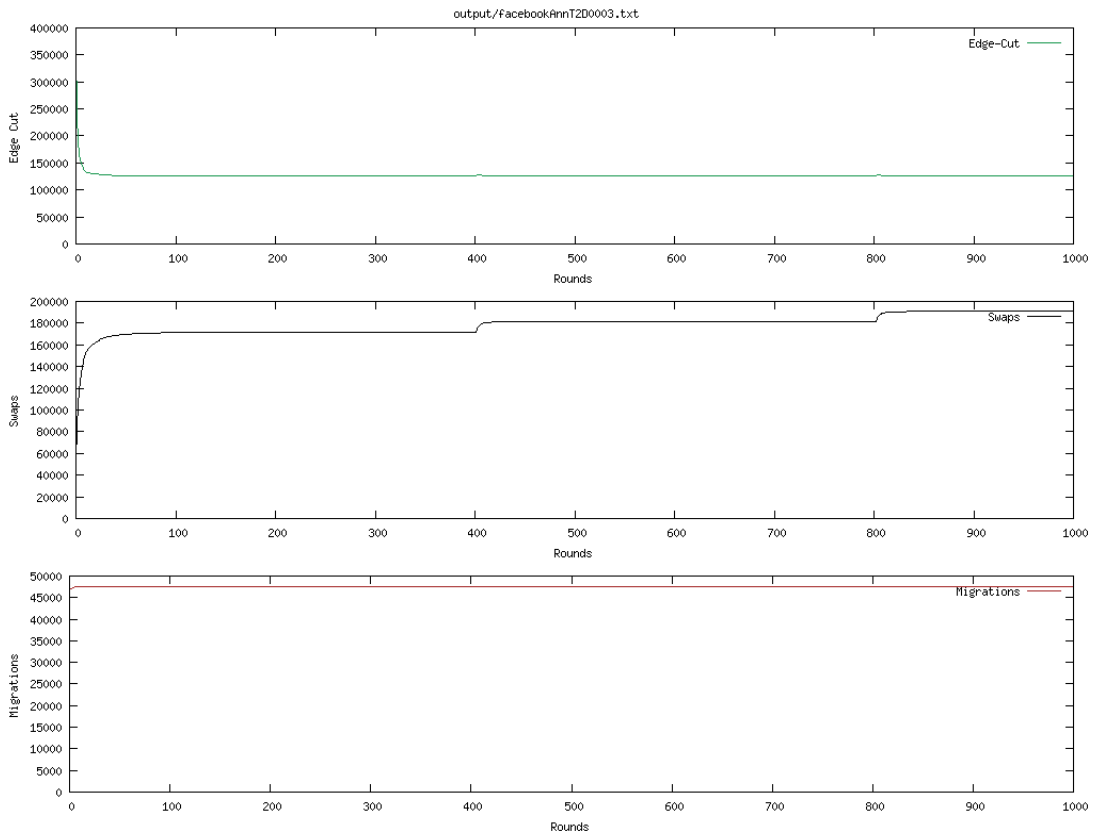


twitter with annealing and (1) T=1, delta=0.9 (2) T=2, delta = 0.9 (3) T=2, delta = 0.09 (4) T=2, delta = 0.003


output/facebookAnn.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds


output/facebookAnnT2D09.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds





facebook with annealing and (1) T=1, delta=0.9 (2) T=2, delta = 0.9 (3) T=2, delta = 0.09 (4) T=2, delta = 0.003

Bonus Task

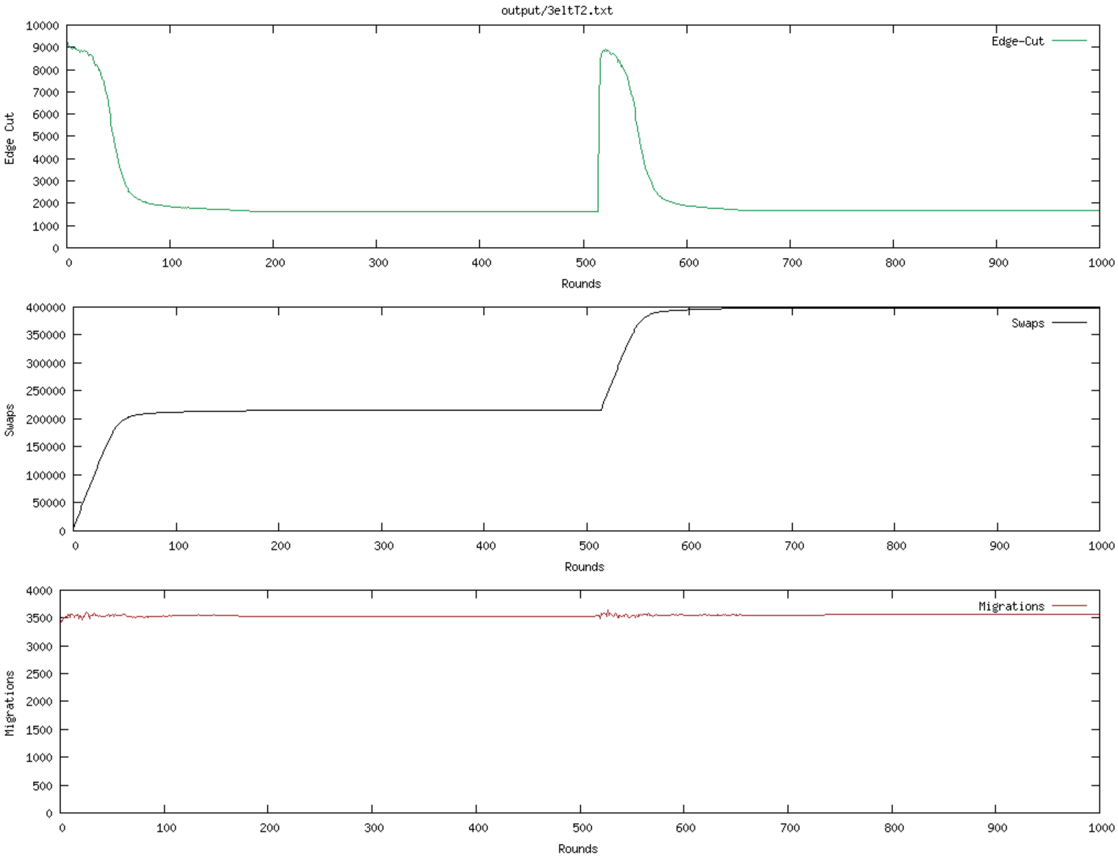
We have defined our own acceptance probability function in order to improve its performance and evaluated how these changes affect the performance of graph partitioning. The acceptance probability based on benefit corresponding to the simulated annealing from before was equal to

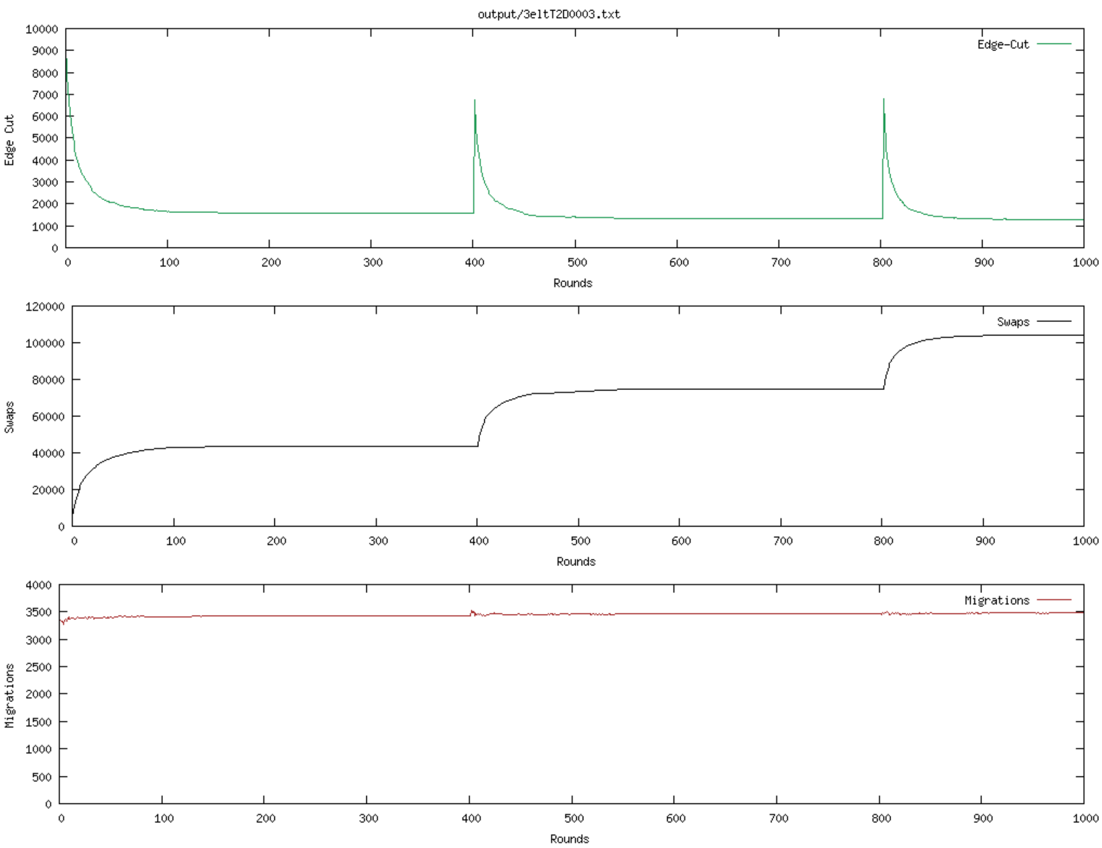
$$a = e^{\frac{c_{new} - c_{old}}{T}}$$

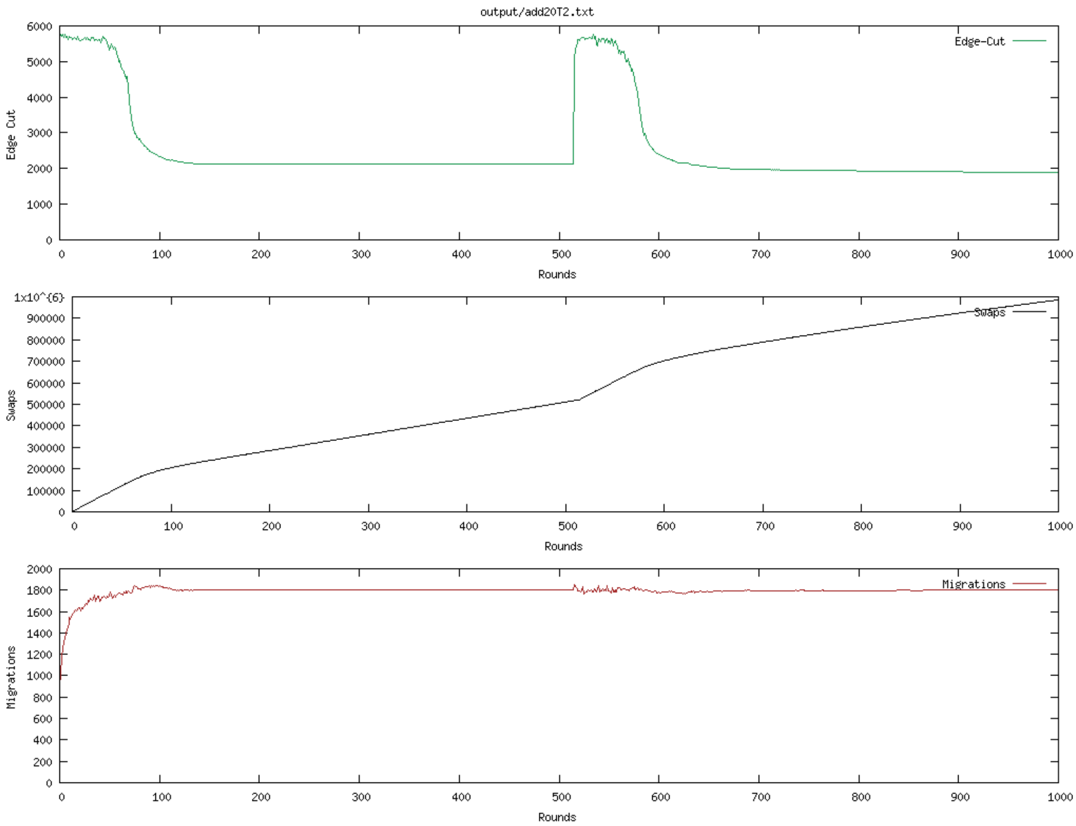
We modified this to

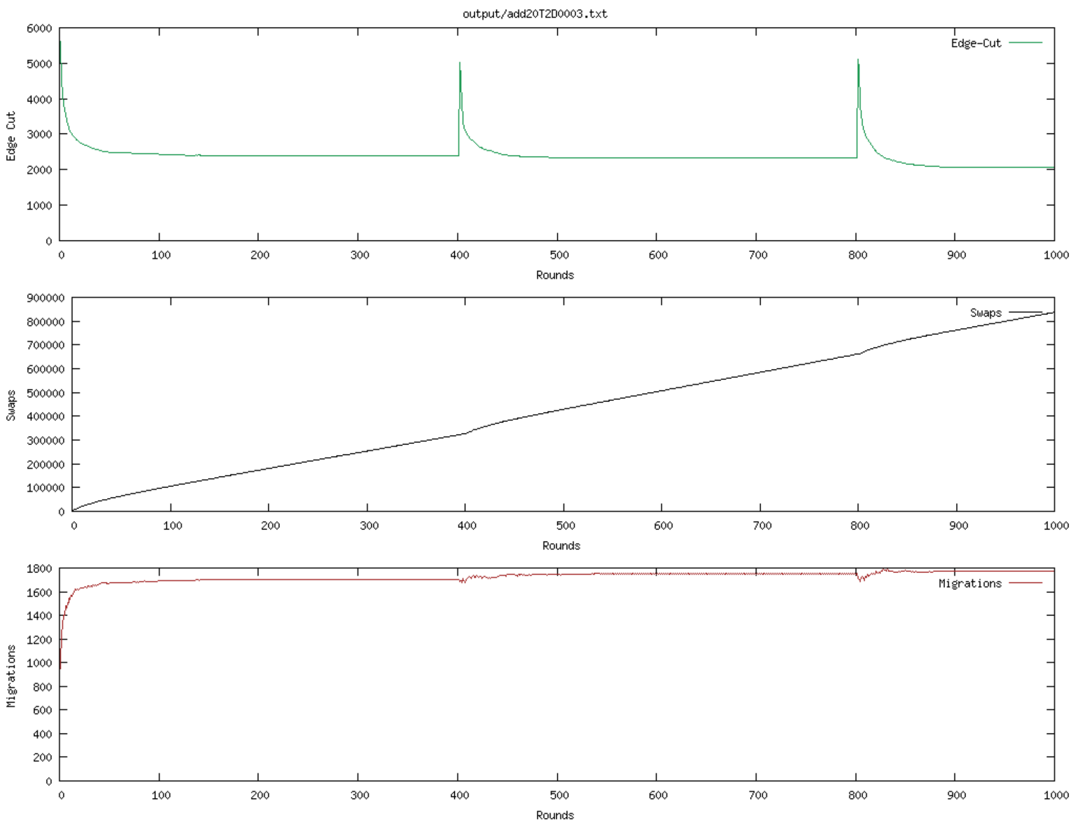
$$a = e^{\frac{\frac{1}{c_{old}} - \frac{1}{c_{new}}}{T}}$$

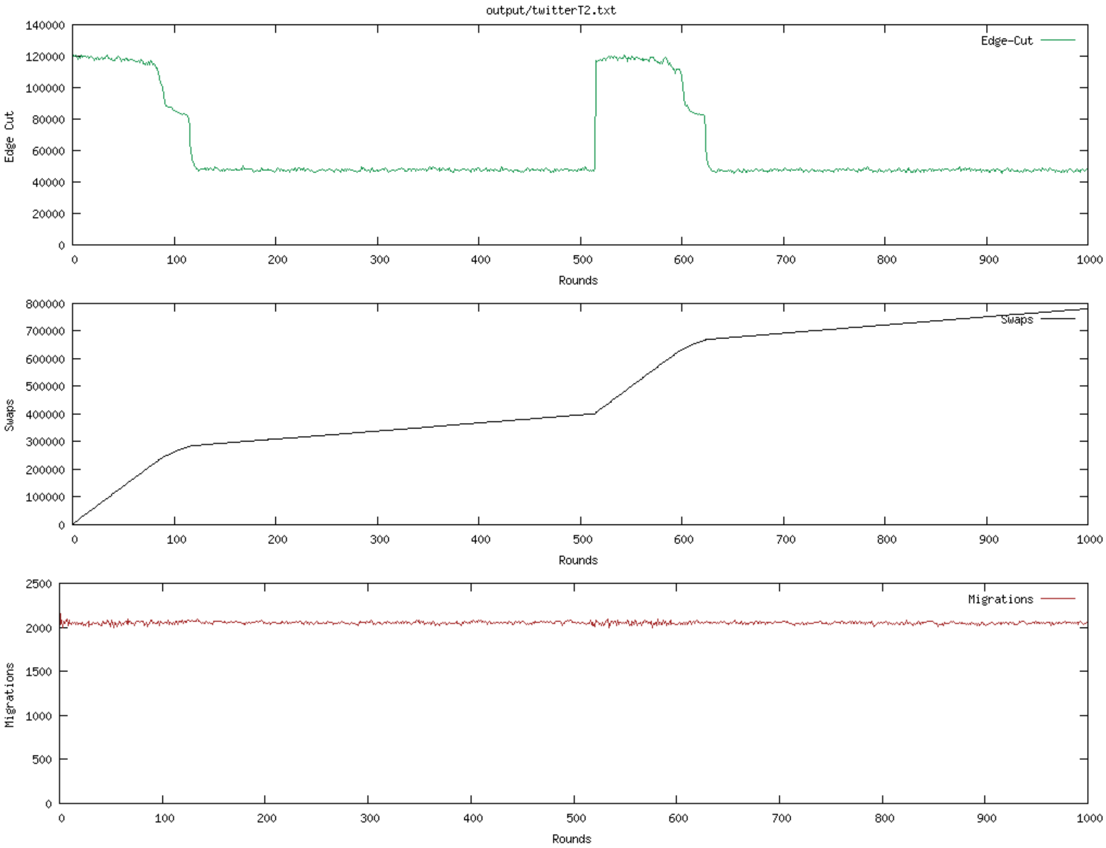
The reasoning behind this change is that for greater differences between c_{new} and c_{old}, we think that it diverges faster compared to the previous one. The results are shown below. Also here we restart simulated-annealing again after 400 rounds, which explains the repetition behaviour.

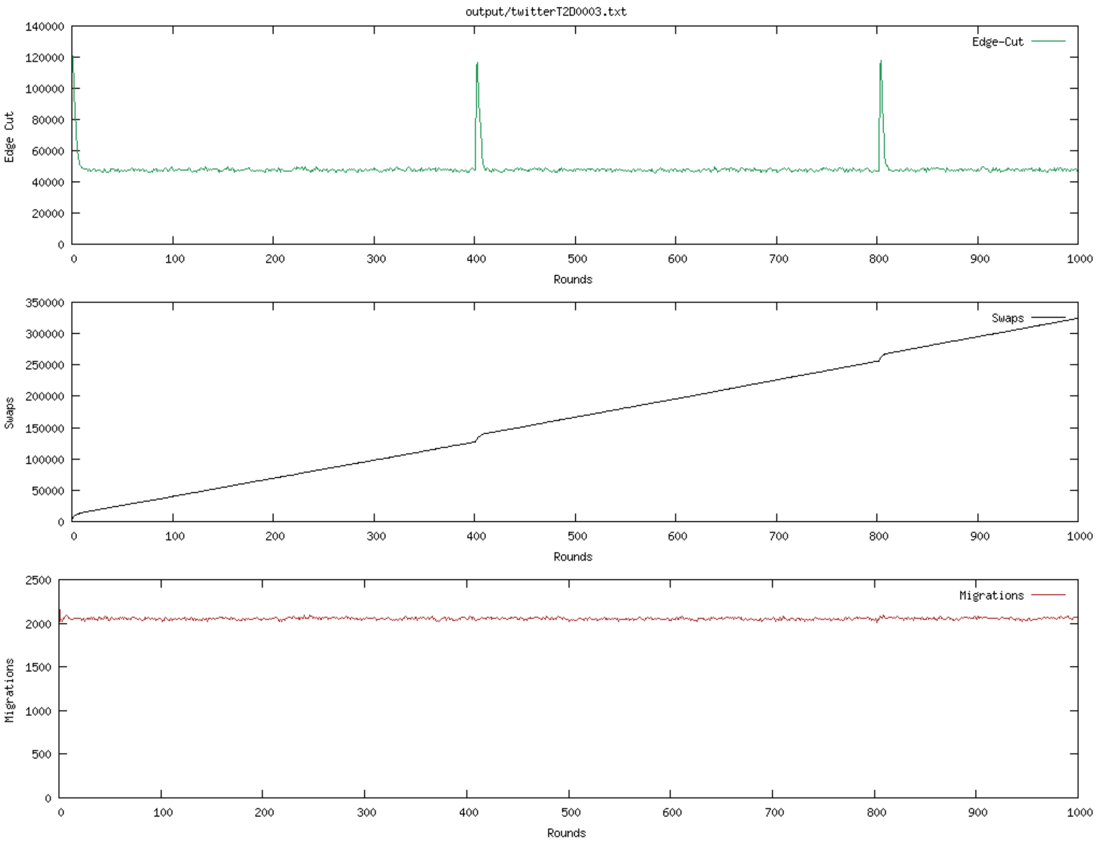

output/3eltT2.txt
Edge Cut
Edge-Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

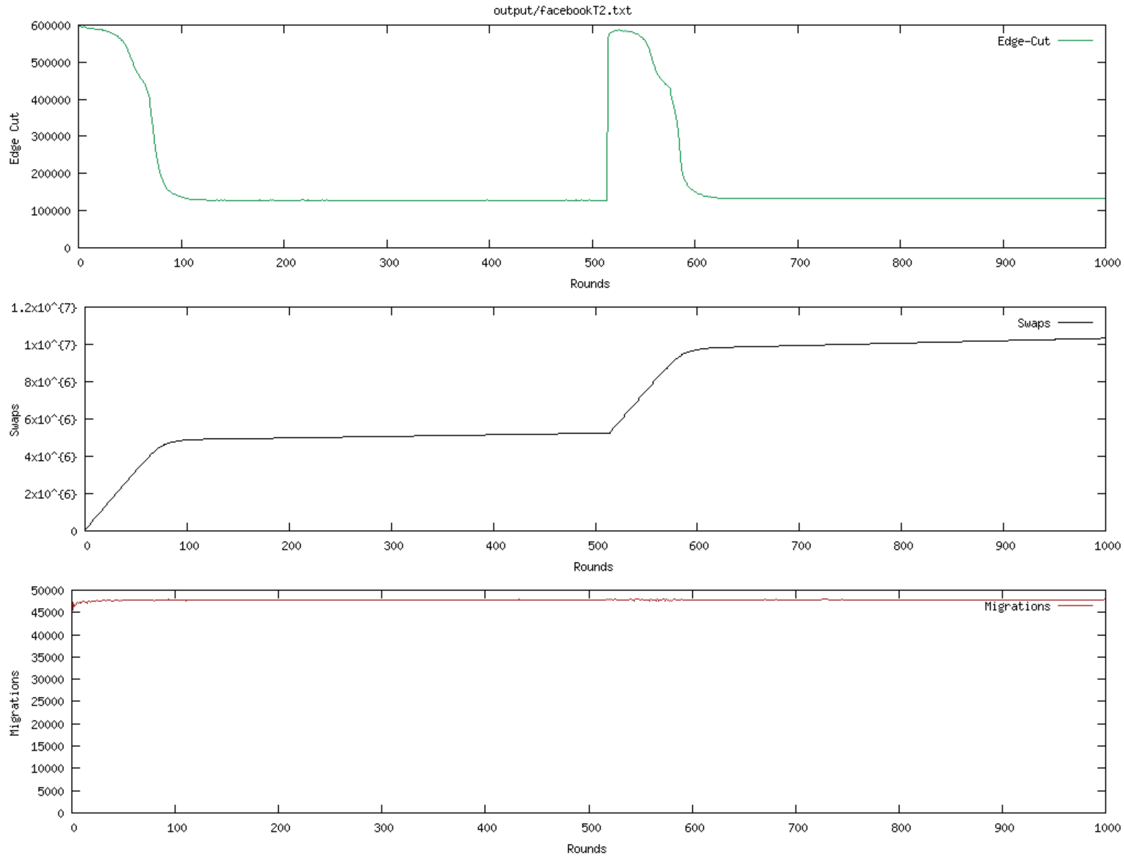

output/3eltT2D0003.txt
Edge Cut
Edge-Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

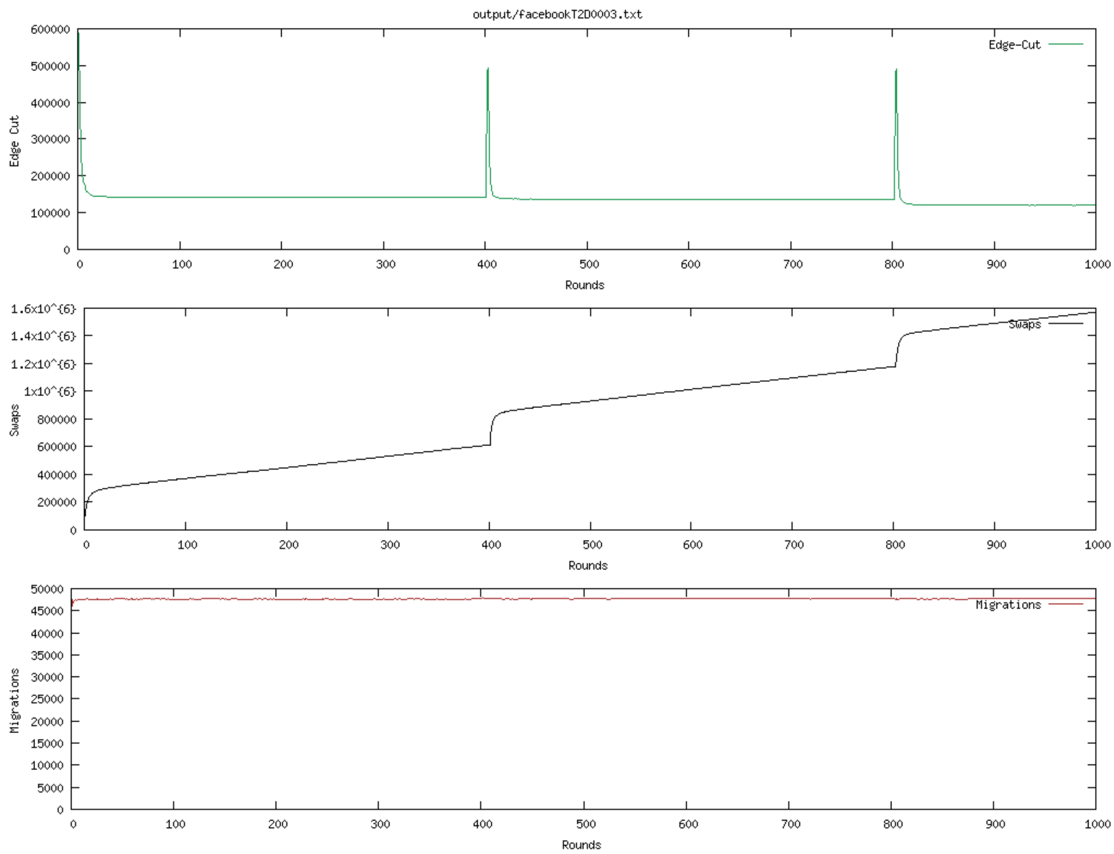

output/add20T2.txt
Edge Cut
Edge-Cut
Rounds
Swaps
1x10^{6}
Rounds
Migrations
Rounds


output/add20T2D0003.txt
Edge Cut
Edge-Cut
Rounds
Swaps
Rounds
Migrations
Rounds


output/twitterT2.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds


output/twitterT2D0003.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds


output/facebookT2.txt
Edge Cut
Edge-Cut
Rounds
Swaps
Rounds
Migrations
Rounds


output/facebookT2D0003.txt
Edge Cut
Edge-Cut
Rounds
Swaps
Rounds
Migrations
Rounds

Dataset	edge-cut	swaps	migrations
3elt	2604	1580209	3328
3elt with ann.	1698	31894	3344
3elt with ann.,T=2, delta = 0.9	1826	33404	3370
3elt with ann.,T=2, delta = 0.09	1924	33778	3305
3elt with ann., T=2, delta=0.003	1695	38995	3370
3elt with annealing and own accept. prob, T=2, delta=0.9	1660	397813	3561
3elt with annealing and own accept. prob, T=2, delta=0.003	1276	104239	3482
add20	2095	1090263	1751
add20 with ann.	2141	777709	1727
add20 with ann.,T=2, delta = 0.9	2317	840628	1735
add20 with ann.,T=2, delta = 0.09	1956	645723	1751
add20 with ann.,T=2, delta = 0.003	1904	684679	1726
add20 with annealing and own accept. prob, T=2, delta=0.9	1889	985220	1802
add20 with annealing and own accept. prob, T=2, delta=0.003	2075	836600	1773
twitter	41156	899515	2049
twitter with ann.	41283	6125	2026
twitter with ann.,T=2, delta = 0.9	41596	6979	2057
twitter with ann.,T=2, delta = 0.09	41285	6225	2047
twitter with ann.,T=2, delta = 0.003	41268	6436	2045
twitter with annealing and own accept. prob, T=2, delta=0.9	48351	780559	2055
twitter with annealing and own accept. prob, T=2, delta=0.003	46821	324001	2064
facebook	134246	212003	47763
facebook with ann.	141397	233200	47588

facebook with ann.,T=2, delta = 0.9	130670	271594	47679
facebook with ann.,T=2, delta = 0.09	145240	183729	47495
facebook with ann.,T=2, delta = 0.003	125850	191568	47537
facebook with annealing and own accept. prob, T=2, delta=0.9	132053	10328809	47912
facebook with annealing and own accept. prob, T=2, delta=0.003	119882	1571153	47723

Table 1: Number of edge-cut, swaps, and migrations of each of the proposed methods for the datasets 3elt, add20, twitter, and facebook.

The goal was to minimize the energy of the system i.e. the number of edges between nodes with different colors (=edge-cut). So the edge cut acts as a quality metric for our partitioning, whereas the number of swaps, i.e. swaps taking place between different hosts during run-time, defines the cost of the algorithm. So we want to see what method gives us the minimum edge-cut, the minimum number of swaps, and what effect this has on the number of migrations. These values between the different methods are highlighted for each dataset in Table 1. The number of migrations only changed by a bit. We noticed that in annealing when we decrease delta then the edge-cuts decrease in their majority and swaps are increased. This can be explained by, when delta is decreased that means the update is done in smaller steps (so slower), which results in more swaps but also may result in better performance i.e. lower edge cut.

How to run

The implementation is written in Java and needs to be run on Linux. Before running the program, it is required to install gnuplot[4] and maven[5]. When this is completed, open the terminal in the directory and execute the following:

```
./compile.sh
```

All graphs are stored in the ./graphs directory. Below is for the graph 3elt:

```
./run.sh -graph ./graphs/3elt.graph
```

Modify the text file name which is generated and saved in the output folder to e.g. "result". Then execute the following command to plot the result:

```
./plot.sh output/result.txt
```

To see all possible command line parameters:

```
./run.sh -help
```

[4] http://gnuplot.sourceforge.net/
[5] https://maven.apache.org/install.html