ID2222 – Data Mining

Homework 5: K-way Graph Partitioning Using JaBeJa

Group 10

Xin Ren

Nikhil Dattatreya Nadig

2018-12-05

Introduction

In this homework, the objective was to understand distributed graph partitioning using gossip-based peer-to-peer techniques, such as JaBeJa. The assignment consisted of two sub-tasks followed by a bonus task.

The first task was to complete JaBeJa algorithm over the boilerplate code provided. The task essentially was to implement the sampleAndSwap and findPartner functions.

The second task required us to tweak different JaBeJa configurations in order to find the smallest edge cuts for the given graphs.

The subtasks were:

- To implement a mechanism such as Simulated Annealing Algorithm and observe how this change affects the rate of convergence.
- To investigate how the Ja-Be-Ja algorithm behaves when the simulated annealing is restarted after Ja-Be-Ja has converged.

We also defined our own acceptance probability function for the bonus task.

Implementation

Following classes were modified for implementation.

Jabeja.java:

- sampleAndSwap: This function takes a node id as input, and if the NodeSelectionPolicy is LOCAL, it finds the best partner to swap color with the node from its neighbors got with function getNeighbors/1 provided; if the NodeSelectionPolicy is RANDOM, it finds the best partner from random sample nodes got with function getSample/1; if the NodeSelectionPolicy is HYBRID, it first tries to find the best partner from the neighbors, and if this fails, it tries to find it from random sample nodes. If the best partner is found, the node swaps color with it if they have different colors. Counter numberOfSwaps increases when swap happens.
- findPartner: This function implements the FINDPARTNER function in the paper. But it has different acceptance probability functions for different tasks. For task 1 and task 2.2, it is "newDegree * T > oldDegree" which is provided by the paper; for task 2.1, we implemented function calcAcceptanceProbability/3 and for bonus task, we implemented function calcAcceptanceProbabilityB/3.
- calcAcceptanceProbability: This is the acceptance probability function for task 2.1, which is implemented according to the Simulated Annealing Algorithm provided, where the probability is defined as $a = e^{(c_{new}-c_{old})/T}$. The function takes newDegree, oldDegree and T as input and simply returns Math.exp((newDegree - oldDegree) / T). Then the returned value is compared with a random number between 0 and 1 in function findPartner to decide if it is accepted or not.

- calcAcceptanceProbabilityB: This is our self defined acceptance probability function for bonus task. The probability is defined as 1-(C_old-C_new)/(max(C_old,C_new)*T). The function takes newDegree, oldDegree and T (starting from 1 and reducing until 0.00001 by multiplying a number smaller than but close to 1 in every round) as input and simply returns 1-((oldDegree - newDegree) / (Math.max(oldDegree, newDegree)*T)). Then the returned value is compared with a random number between 0 and 1 in function findPartner to decide if it is accepted or not.
- saCoolDown: This function is updated to support task 2.1 and bonus task, where temperature is cooling down by multiplying a number smaller than but close to 1 in every round.
- startJabeja: This function is updated to support task 2.2. The temperature is reset to initial configured value every certain rounds which is also configurable. The ideal configuration should make sure that it is reset after the algorithm converges everytime.
- Jabeja: Initial temperature set to 1 for task 2.1 and bonus task.
- saveToFile: Parameter outputFilePath is updated to support different tasks.
- T_min: A new float number added to define the lower boundary of the temperature in task 2.1.

rand/RandNoGenerator.java: A new function nextDouble/0 is added to generate a random number between 0 and 1 to be used in function findPartner.

io/CLI.java: Following CLI parameters were added.

- -task: Task ID in the assignment. Supported values are 1, 2.1, 2.2, 3(Bonus). Default is 1.
- -alpha2: The number used to multiply with temperature to cool it down in task 2.1. Default value is 0.9.
- -restart: The number defines how often the temperature is reset. Default is 500 (rounds).

config/Config.java: set and get functions are added for new CLI parameters.

How to Run

>> ./compile.sh
>> ./run.sh -graph ./graphs/XXXX.graph -task Y (for usage, run "./run.sh -help")
>>./plot .sh output/result

Graph Analysis (Results)

We performed following tests for each graph.

For 3elt.graph, the lowest edge cuts we found is 600 in task 2.2 with delta 0.003, round number 10000 and resetting temperature every 500 rounds. The Simulated Annealing Algorithm in task 2.1 provided better performance than the one in the paper (task 1). Resetting temperature brought better performance as well. In task 2.1, increasing alpha brought worse performance, same as increasing delta in task 2.2. Our self defined acceptance probability function provided

similar performance as the one in task 2.1, which is better than the performance of the one in the paper regarding edge cuts and swaps, and it did not converge within 1000 rounds which means we could get even better performance by increasing round number.

Task ID	Configurations	Edge Cuts	Swaps	Migrations
1	T_2.0_D_0.003_A_2.0_R_1000	1417	29810	3361
2.1	T_1.0_A_2.0_A2_0.9_R_1000	1259	17357	3402
	T_1.0_A_2.0_A2_0.9_R_10000	806	26630	3450
	T_1.0_A_2.0_A2_0.99_R_1000	1295	18535	3400
	T_1.0_A_2.0_A2_0.99_R_10000	824	27180	3456
2.2	T_2.0_D_0.003_A_2.0_R_1000_RA_500	1196	36726	3414
	T_2.0_D_0.003_A_2.0_R_10000_RA_500	600	76172	3497
	T_2.0_D_0.005_A_2.0_R_9000_RA_300	688	64681	3435
	T_2.0_D_0.01_A_2.0_R_10000_RA_200	661	55587	3470
Bonus	T_1.0_A_2.0_A2_0.9_R_1000	1258	23144	3369

Table 1: 3elt.graph results

For add20.graph, the lowest edge cuts we found is 1366 in task 2.2 with delta 0.003, round number 10000 and resetting temperature every 500 rounds. The Simulated Annealing Algorithm in task 2.1 provided worse performance than the one in the paper (task 1). However resetting temperature brought better performance. In task 2.1, increasing alpha does not impact performance much, same as increasing delta in task 2.2. Our self defined acceptance probability function provided similar performance as the one in the paper regarding edge cuts and convergence time, but lower swaps.

Task ID	Configurations	Edge Cuts	Swaps	Migrations
1	T_2.0_D_0.003_A_2.0_R_1000	1411	14421	1787
2.1	T_1.0_A_2.0_A2_0.9_R_1000	1559	6796	1791
	T_1.0_A_2.0_A2_0.9_R_10000	1550	10340	1782
	T_1.0_A_2.0_A2_0.99_R_1000	1554	11073	1778
	T_1.0_A_2.0_A2_0.99_R_10000	1554	14403	1779
2.2	T_2.0_D_0.003_A_2.0_R_1000_RA_500	1405	22971	1806
	T_2.0_D_0.003_A_2.0_R_10000_RA_500	1366	157626	1799

	T_2.0_D_0.005_A_2.0_R_9000_RA_300	1373	141923	1795
	T_2.0_D_0.01_A_2.0_R_10000_RA_200	1366	118532	1781
Bonus	T_1.0_A_2.0_A2_0.9_R_1000	1414	9600	1785

Table 2: add20.graph results

For facebook.graph, the lowest edge cuts we found is 117994 in task 2.2 with delta 0.003, round number 1000 and resetting temperature every 500 rounds. The Simulated Annealing Algorithm in task 2.1 provided worse performance than the one in the paper (task 1). However resetting temperature brought better performance. In task 2.1, increasing alpha brought worse performance, same as increasing delta in task 2.2. Our self defined acceptance probability function did not provide good performance for this graph.

Task ID	Configurations	Edge Cuts	Swaps	Migrations
1	T_2.0_D_0.003_A_2.0_R_1000	136172	918074	47849
2.1	T_1.0_A_2.0_A2_0.9_R_1000	145848	243623	47551
	T_1.0_A_2.0_A2_0.99_R_1000	145899	263350	47557
2.2	T_2.0_D_0.003_A_2.0_R_1000_RA_500	117994	1551108	47816
	T_2.0_D_0.005_A_2.0_R_900_RA_300	131149	1414935	47555
	T_2.0_D_0.01_A_2.0_R_1000_RA_200	124540	1280938	47679
Bonus	T_1.0_A_2.0_A2_0.9_R_1000	180536	460945	47639

Table 3: facebook.graph results

For twitter.graph, the lowest edge cuts we found is 41121 in task 2.2 with delta 0.005, round number 900 and resetting temperature every 300 rounds. All the different configurations somehow provided similar performance. Our self defined acceptance probability function provided lower swaps and shorter convergence time than the one in the paper.

Task ID	Configurations	Edge Cuts	Swaps	Migrations
1	T_2.0_D_0.003_A_2.0_R_1000	41231	169674	2061
2.1	T_1.0_A_2.0_A2_0.9_R_1000	41215	6456	2038
	T_1.0_A_2.0_A2_0.99_R_1000	41222	6888	2044
2.2	T_2.0_D_0.003_A_2.0_R_1000_RA_500	41267	201338	2068
	T_2.0_D_0.005_A_2.0_R_900_RA_300	41121	148477	2014
	T_2.0_D_0.01_A_2.0_R_1000_RA_200	41306	104005	2061

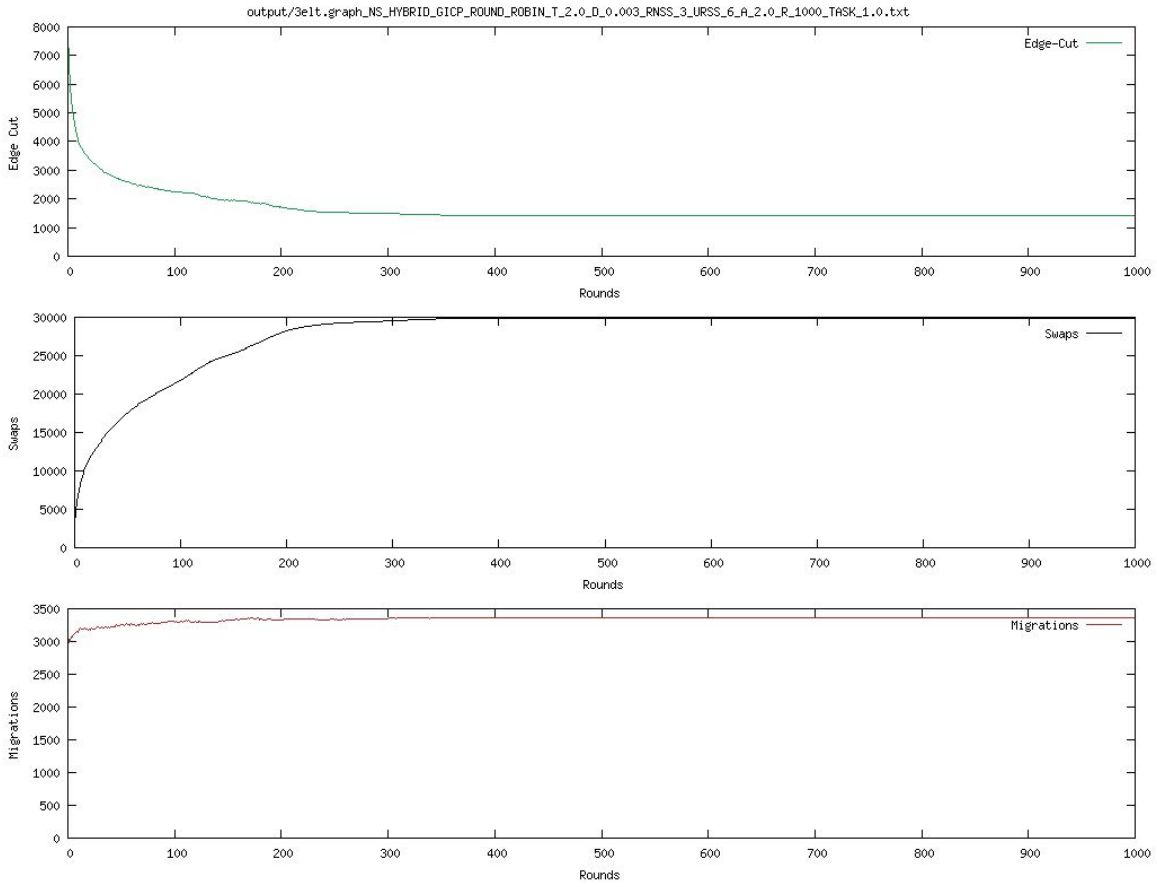
Bonus	T_1.0_A_2.0_A2_0.9_R_1000	41295	16059	2052

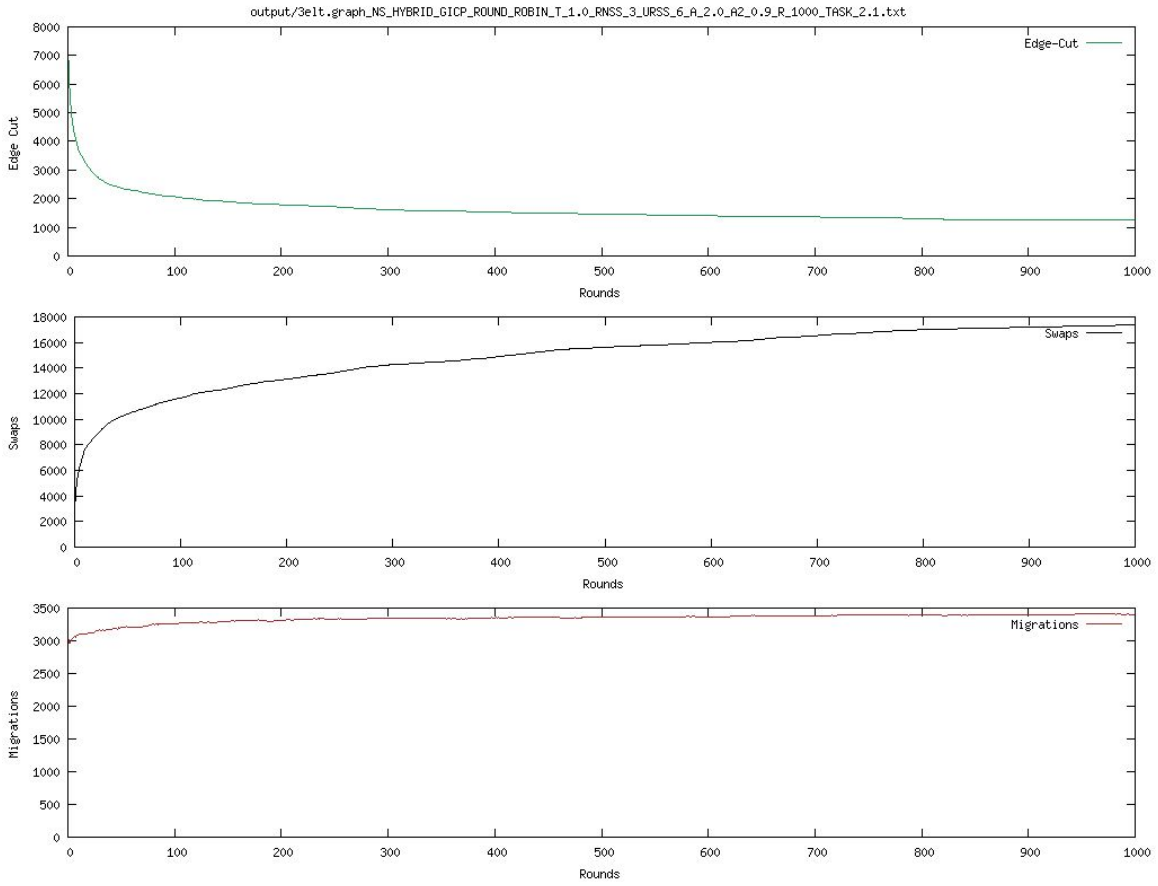
Table 4: twitter.graph results

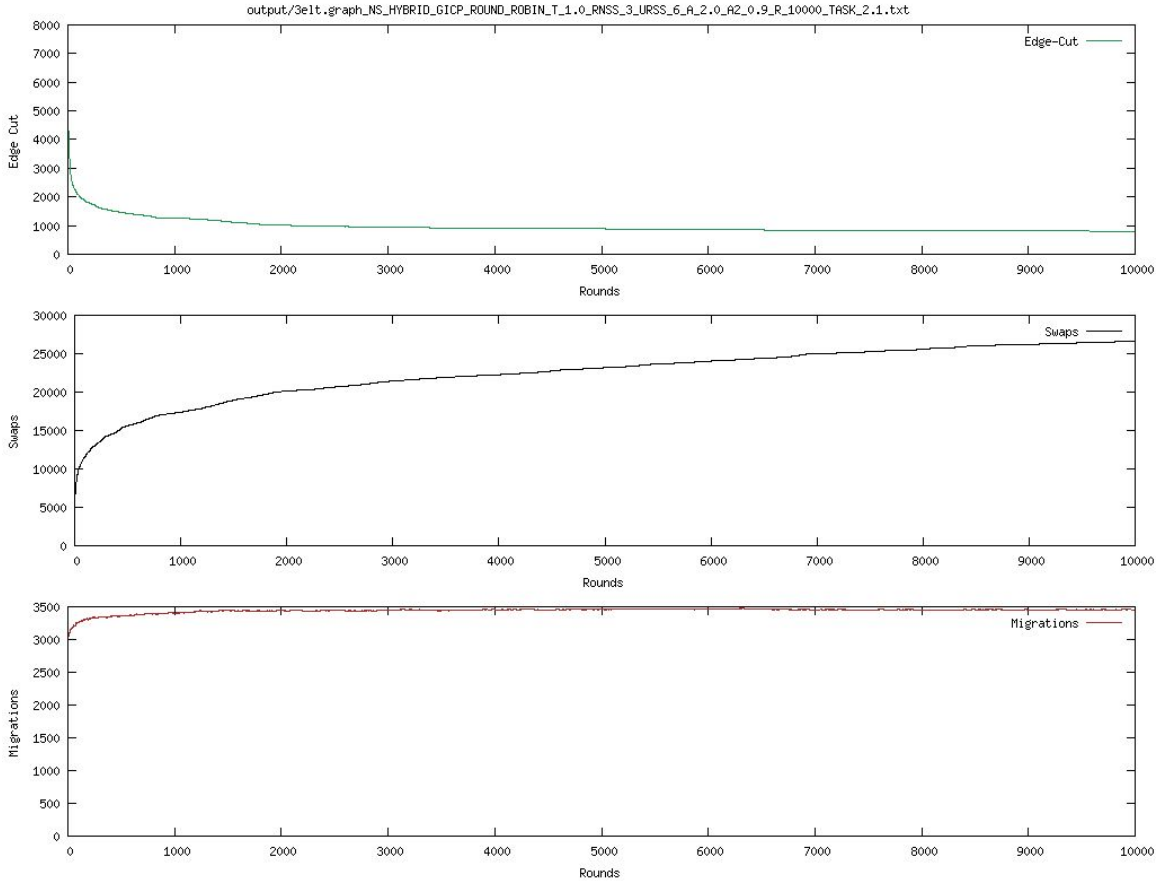
Appendix (Figures)

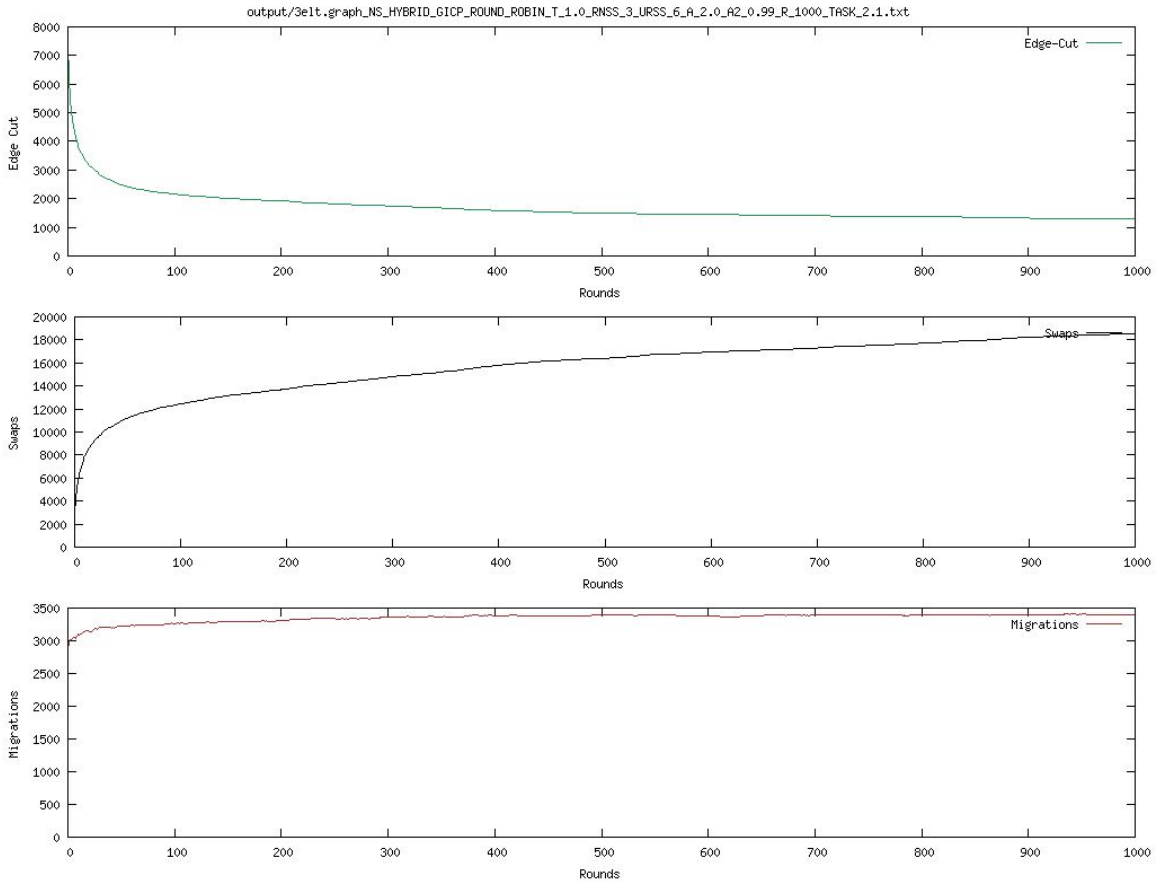
Here we provide all the figures from our tests.

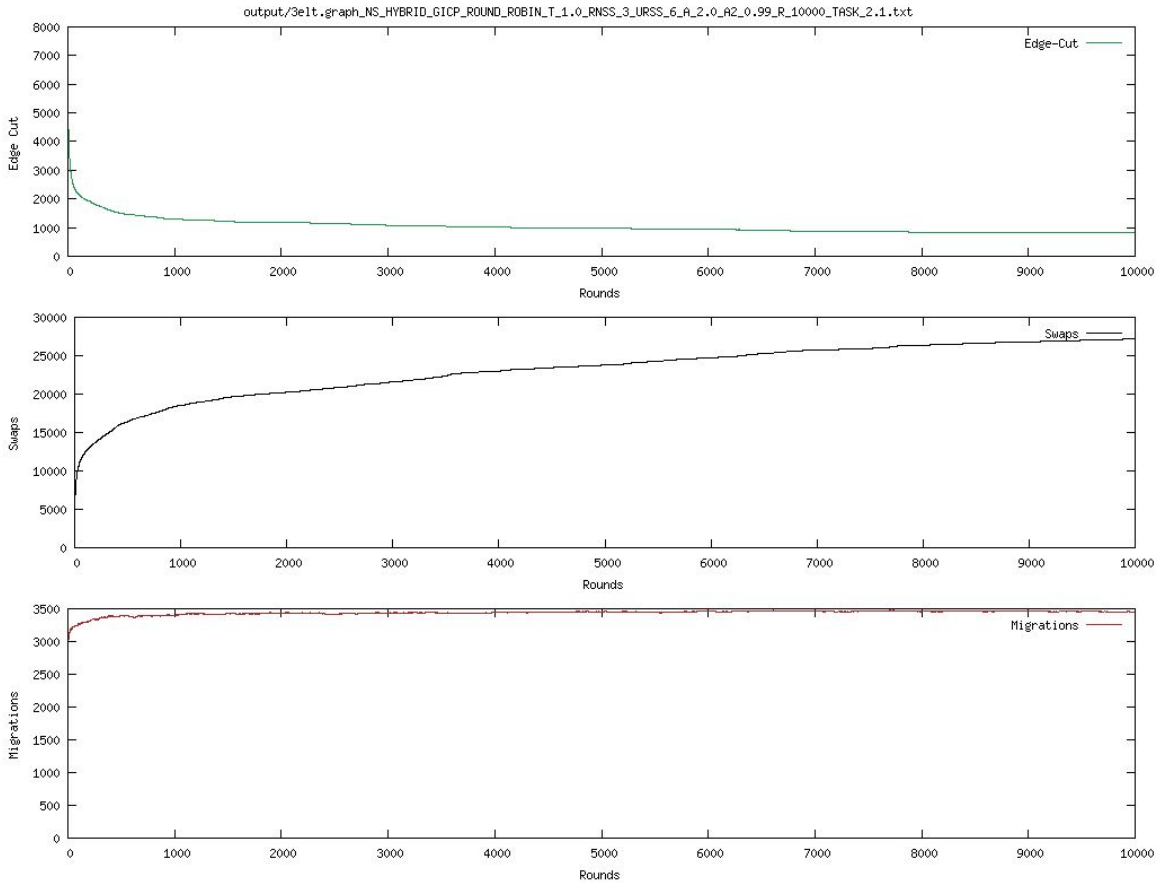
3elt.graph

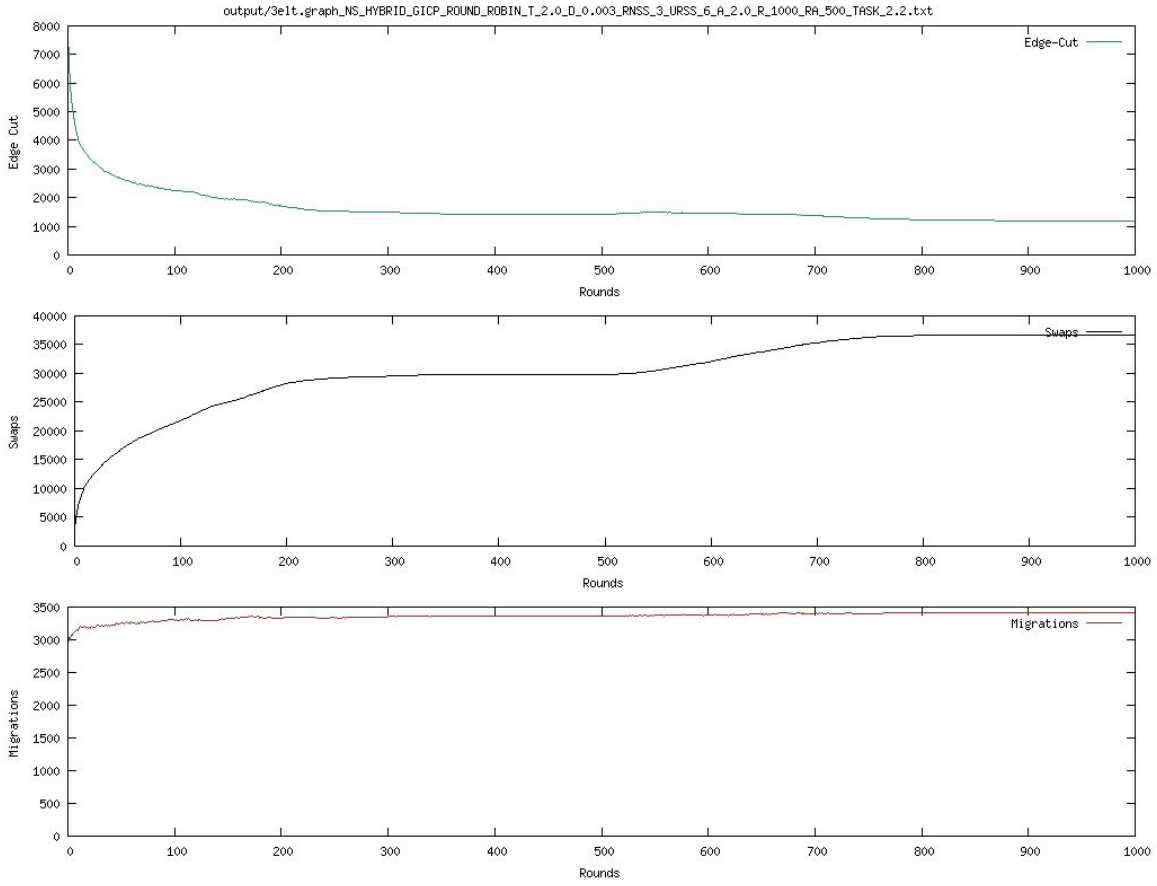


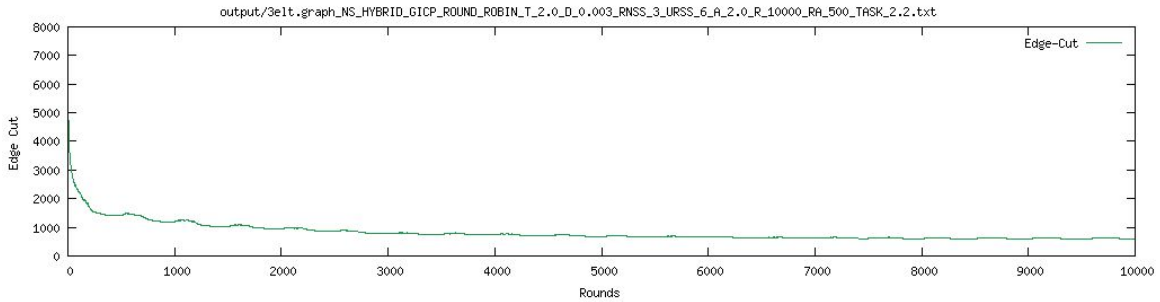

output/3elt.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_1.0_RNSS_3_URSS_6_A_2.0_A2_0.9_R_1000_TASK_2.1.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

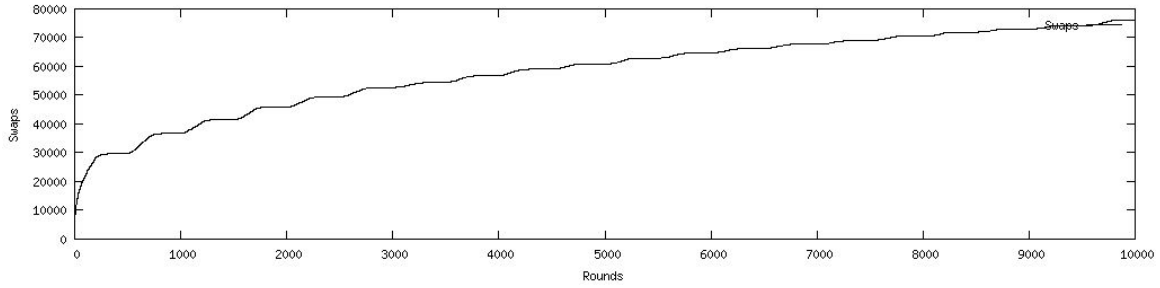

output/3elt.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_1.0_RNSS_3_URSS_6_A_2.0_A2_0.9_R_10000_TASK_2.1.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

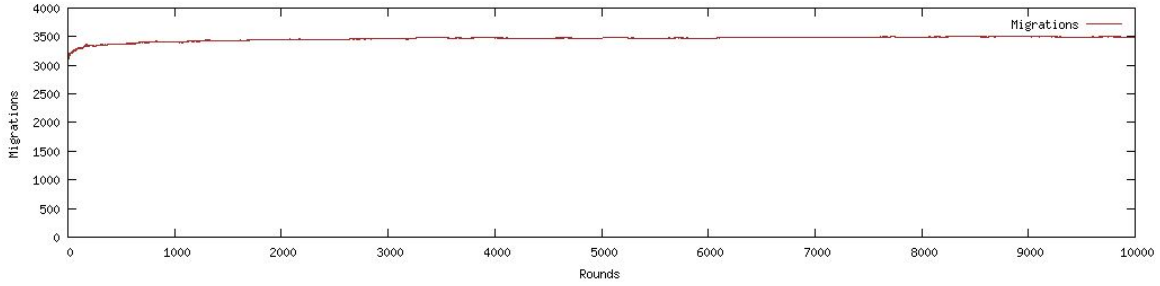

output/3elt.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_1.0_RNSS_3_URSS_6_A_2.0_A2_0.99_R_1000_TASK_2.1.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

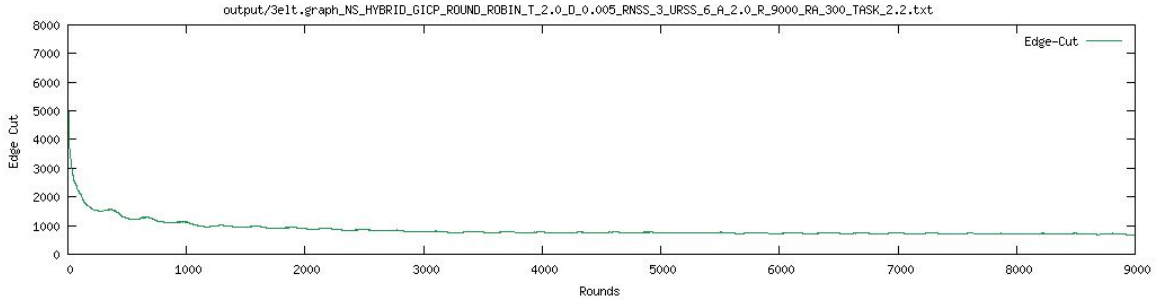

output/3elt.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_1.0_RNSS_3_URSS_6_A_2.0_A2_0.99_R_10000_TASK_2.1.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

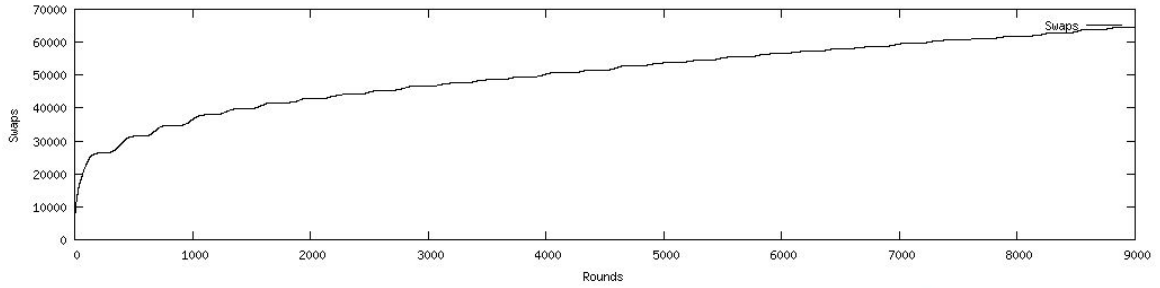

output/3elt.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_2.0_D_0.003_RNSS_3_URSS_6_A_2.0_R_1000_RA_500_TASK_2.2.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

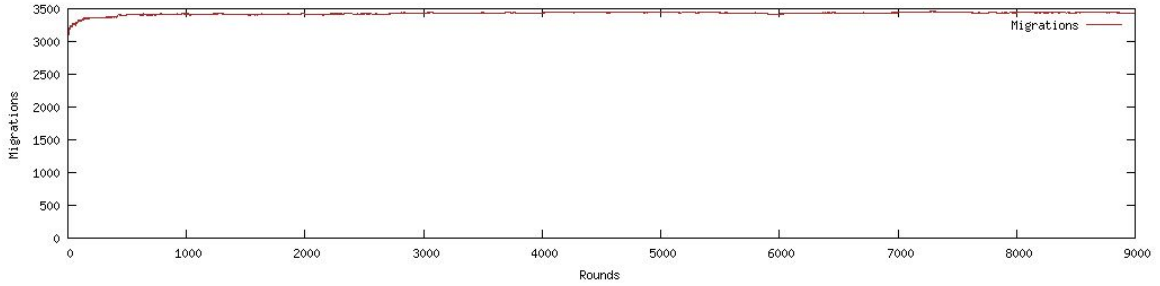

output/3elt.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_2.0_D_0.003_RNSS_3_URSS_6_A_2.0_R_10000_RA_500_TASK_2.2.txt
Edge-Cut
Edge Cut
Rounds

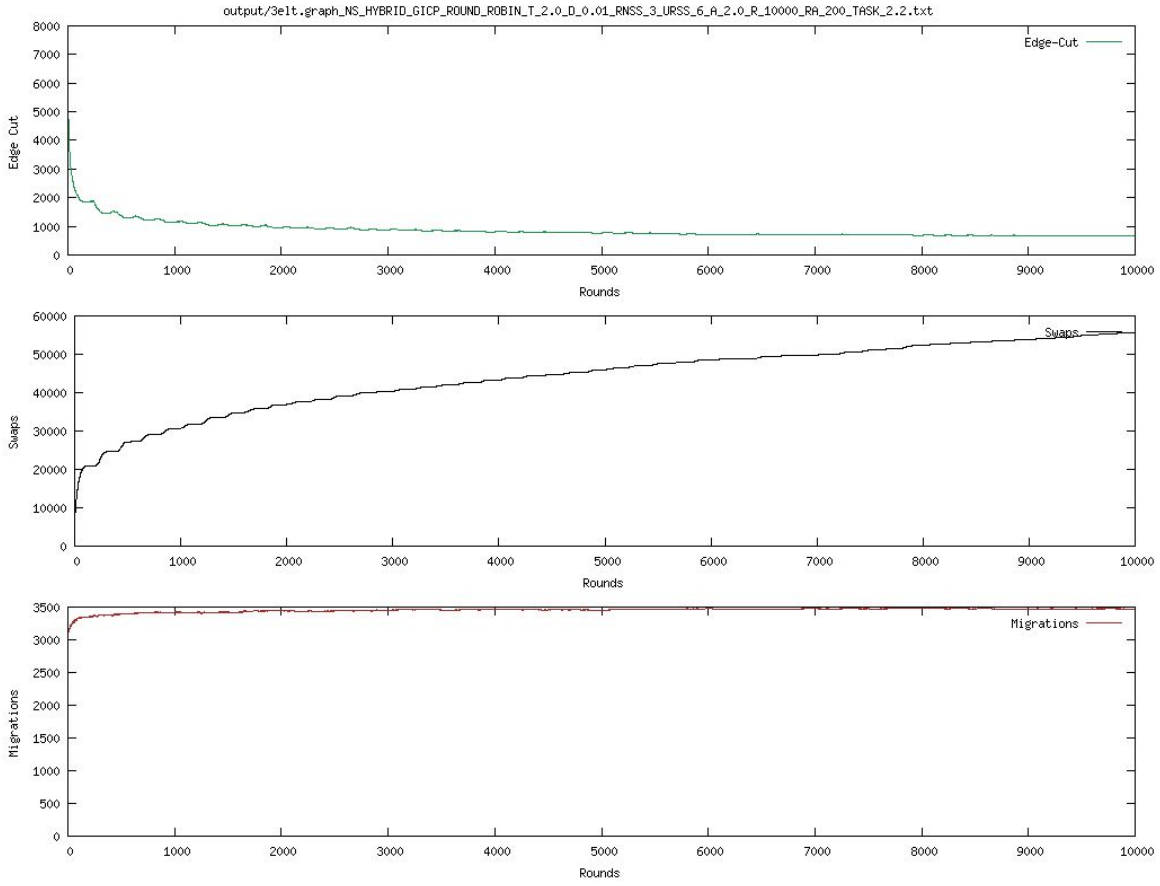

Swaps
Swaps
Rounds

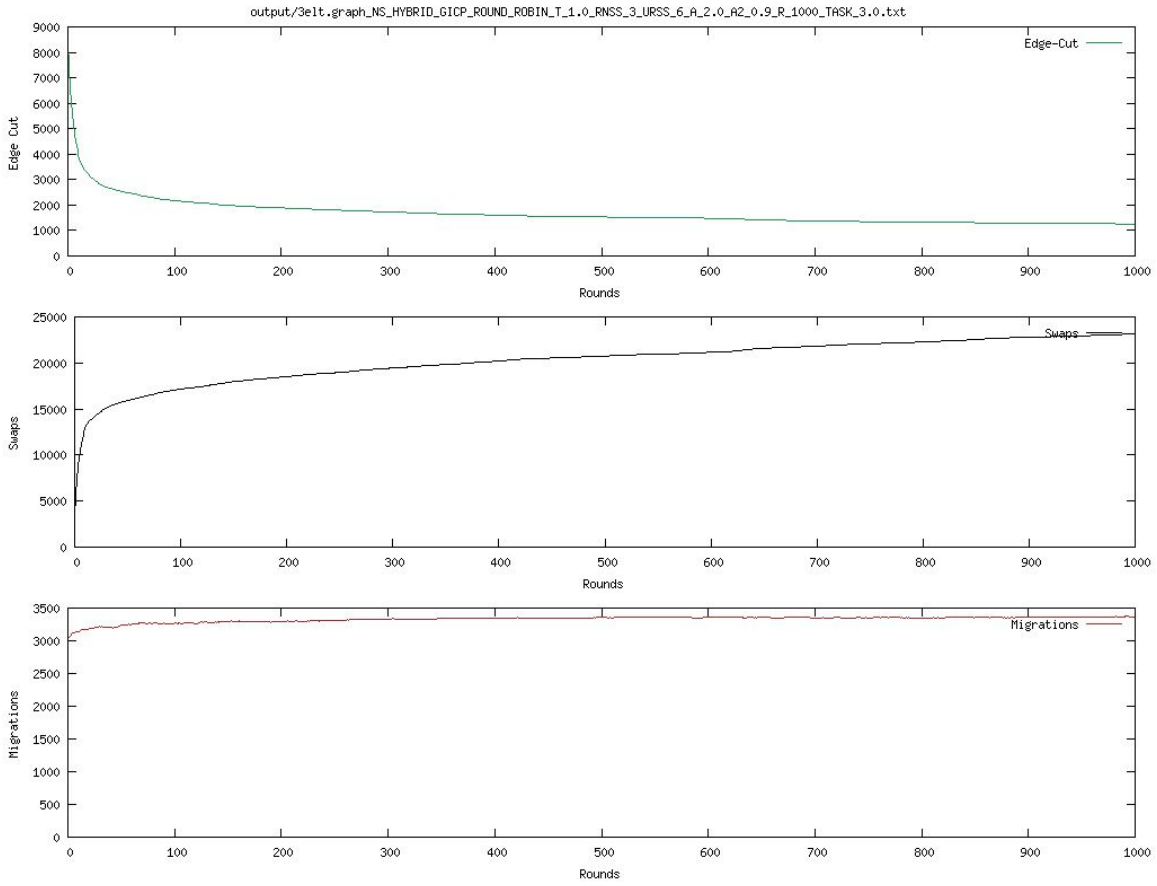

Migrations
Migrations
Rounds


output/3elt.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_2.0_D_0.005_RNSS_3_URSS_6_A_2.0_R_9000_RA_300_TASK_2.2.txt
Edge-Cut
Edge Cut
Rounds

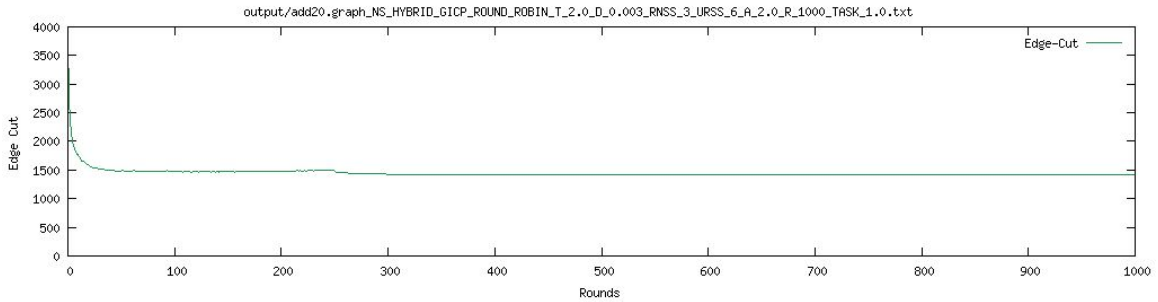

Swaps
Swaps
Rounds

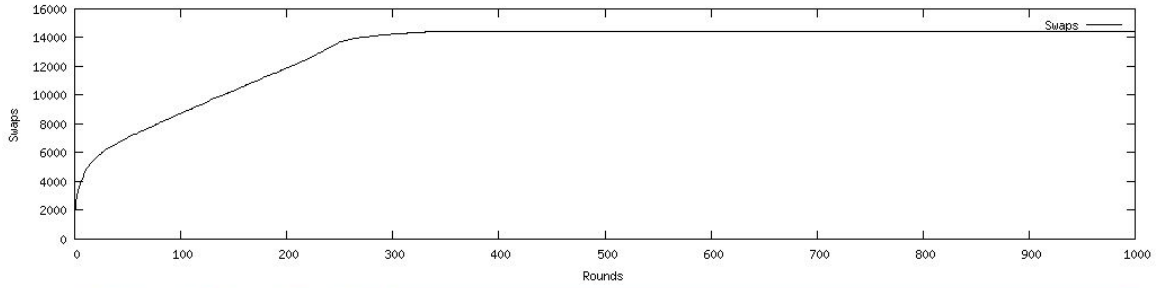

Migrations
Migrations
Rounds

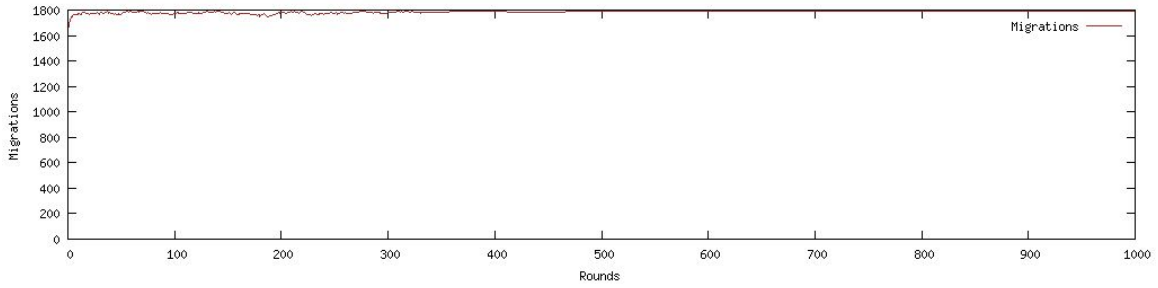

output/3elt.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_2.0_D_0.01_RNSS_3_URSS_6_A_2.0_R_10000_RA_200_TASK_2.2.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

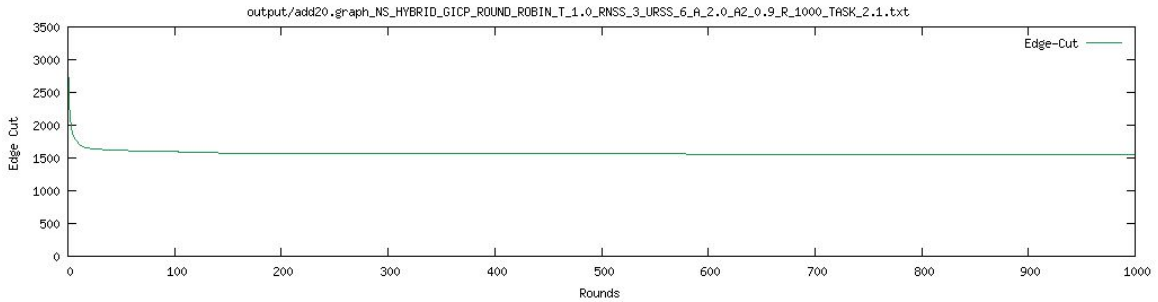

output/3elt.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_1.0_RNSS_3_URSS_6_A_2.0_A2_0.9_R_1000_TASK_3.0.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

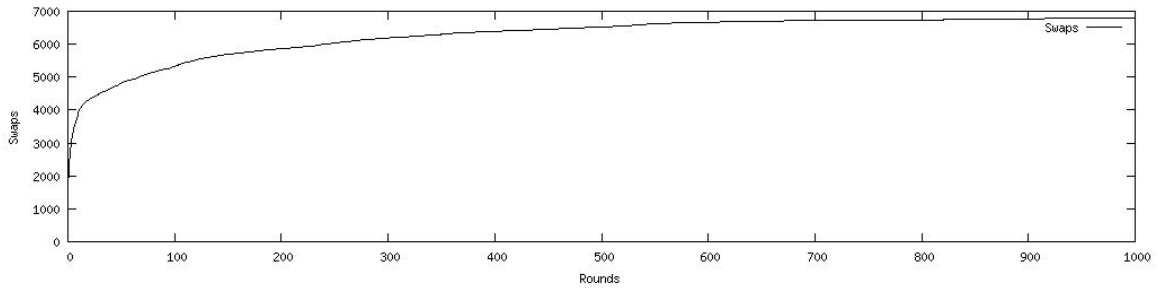
add20.graph

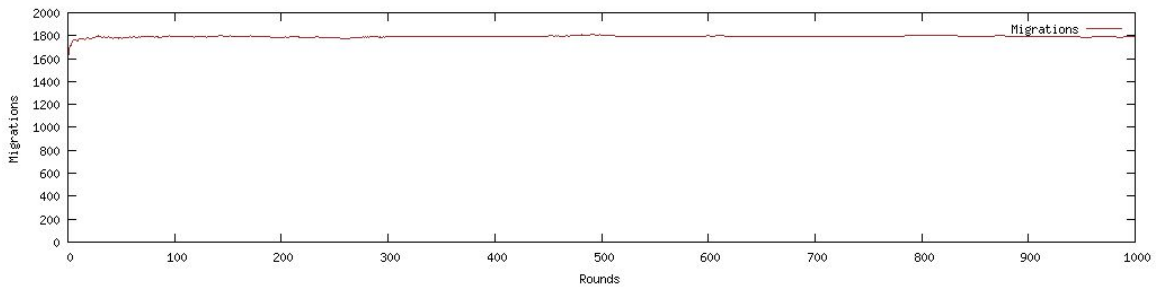

output/add20.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_2.0_D_0.003_RNSS_3_URSS_6_A_2.0_R_1000_TASK_1.0.txt
Edge-Cut
Edge Cut
Rounds

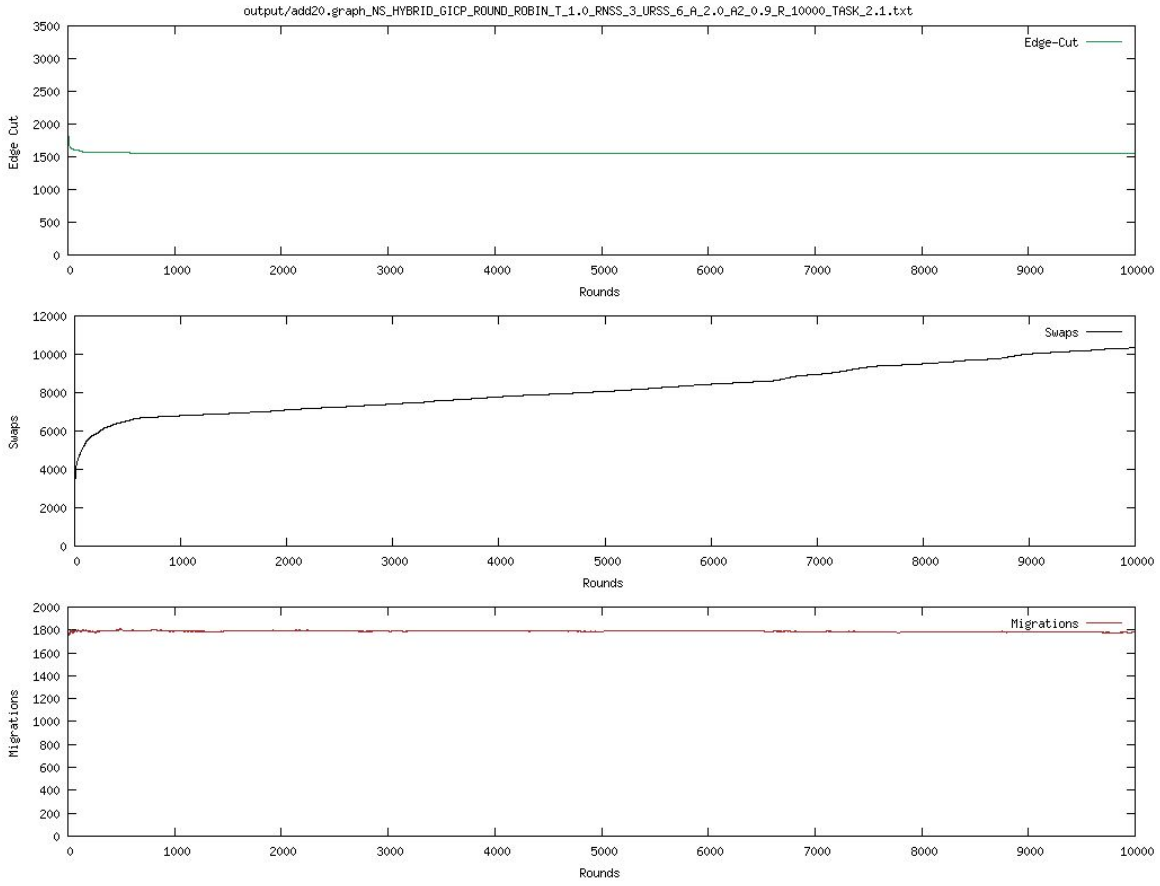

Swaps
Swaps
Rounds

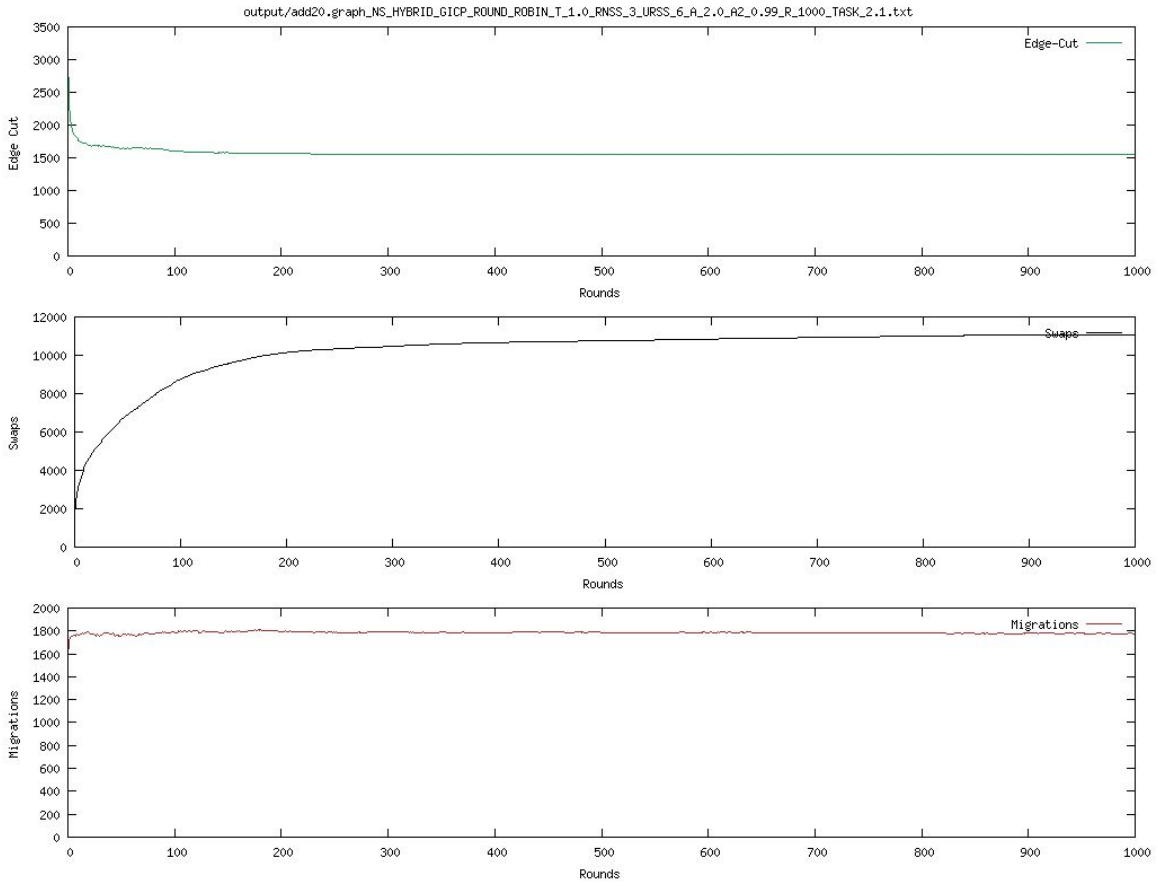

Migrations
Migrations
Rounds

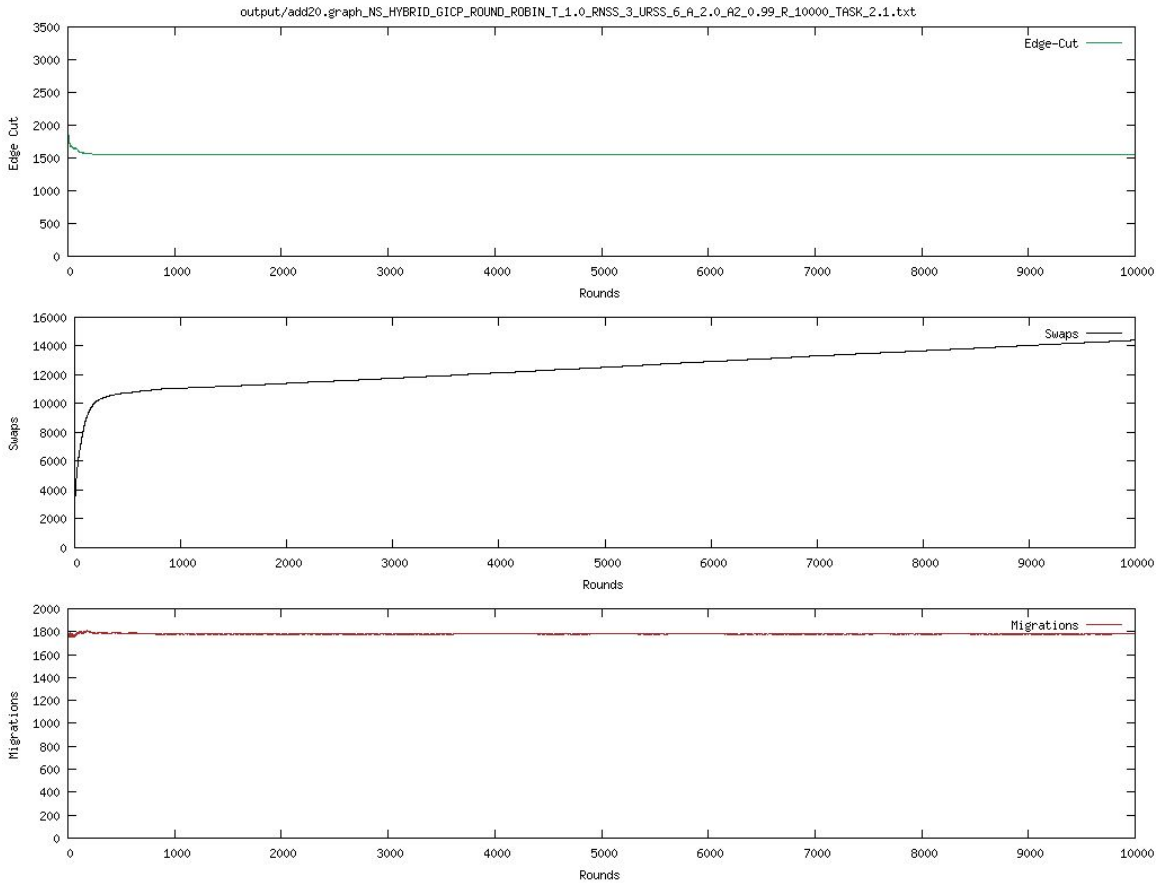

output/add20.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_1.0_RNSS_3_URSS_6_A_2.0_A2_0.9_R_1000_TASK_2.1.txt
Edge-Cut
Edge Cut
Rounds

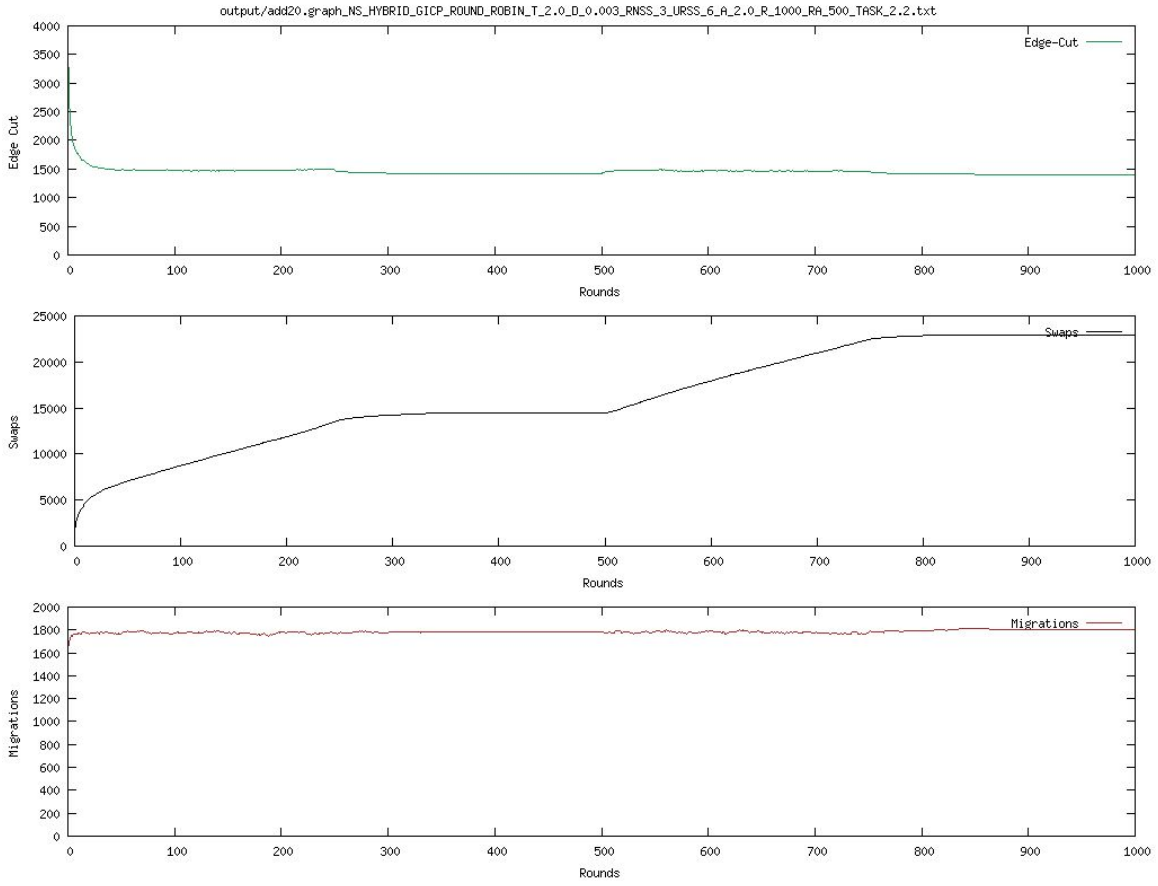

Swaps
Swaps
Rounds

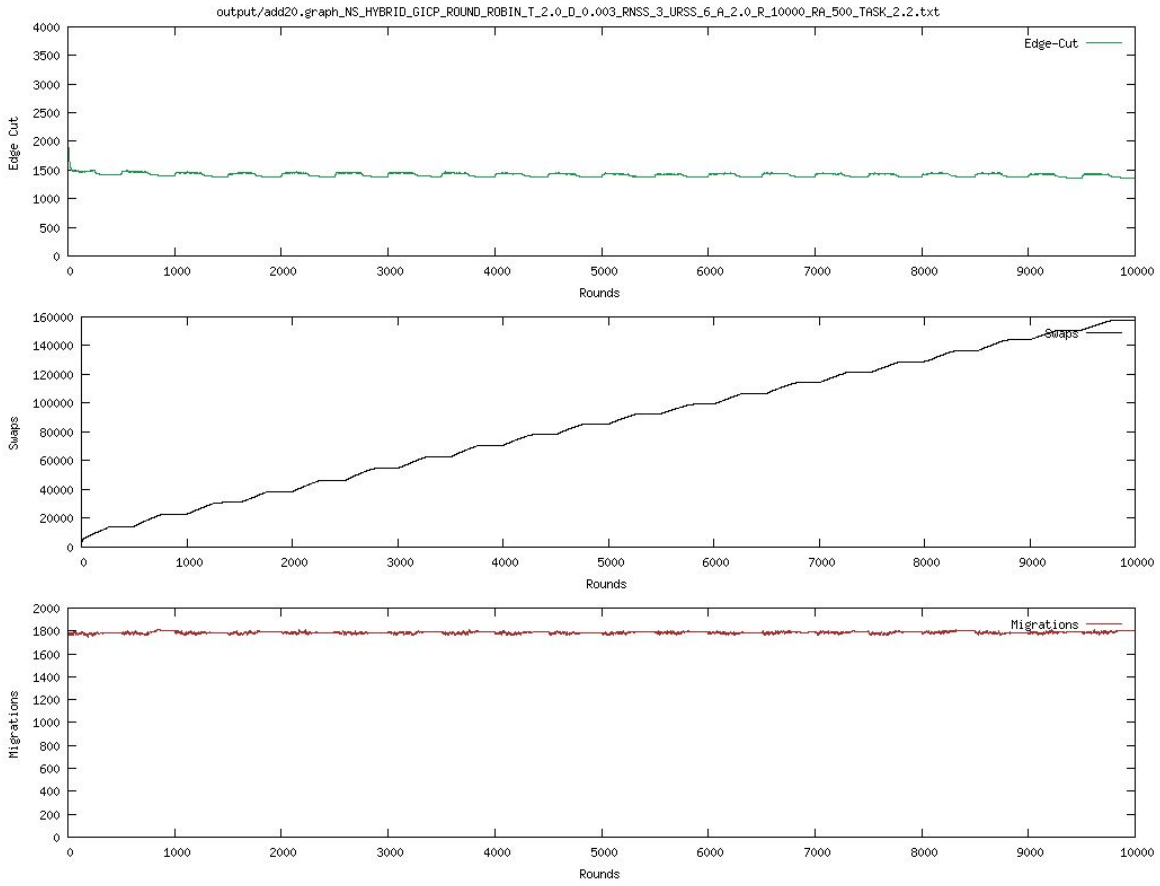

Migrations
Migrations
Rounds

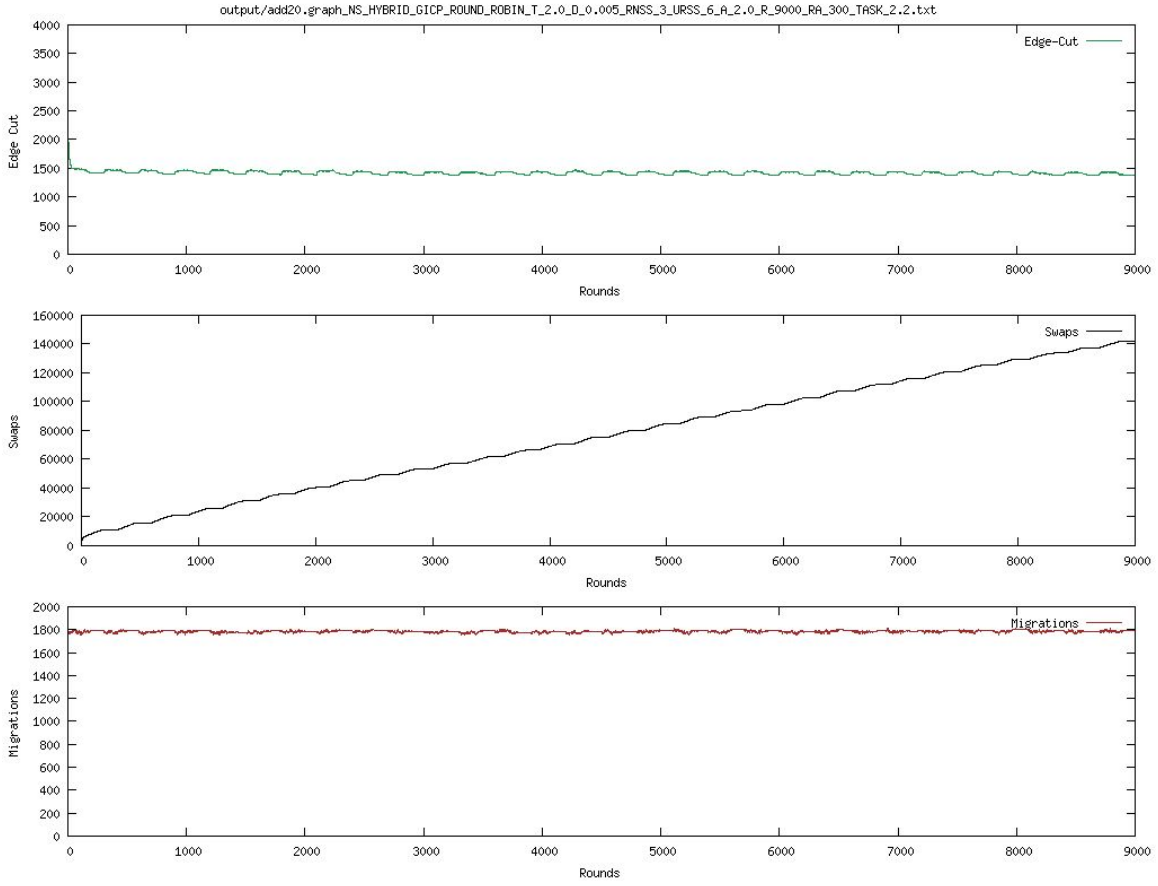

output/add20.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_1.0_RNSS_3_URSS_6_A_2.0_A2_0.9_R_10000_TASK_2.1.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

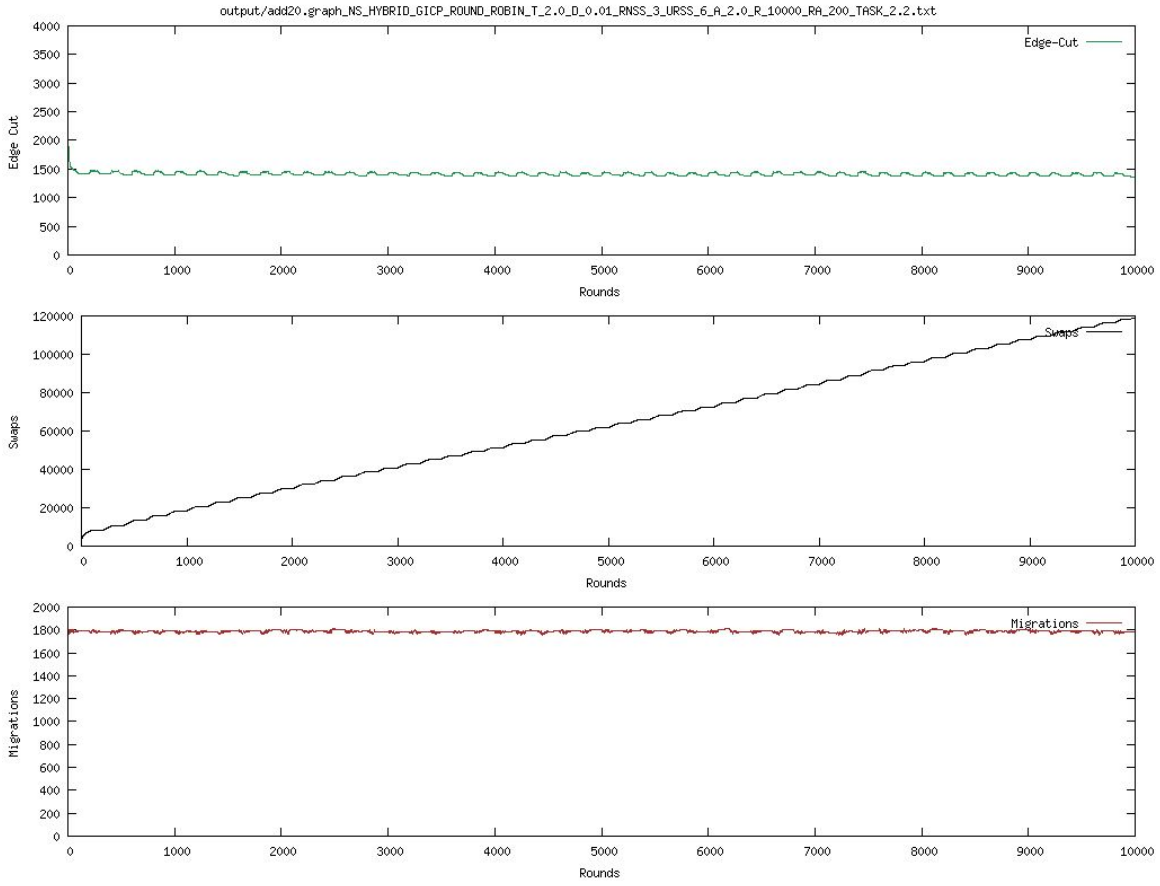

output/add20.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_1.0_RNSS_3_URSS_6_A_2.0_A2_0.99_R_1000_TASK_2.1.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

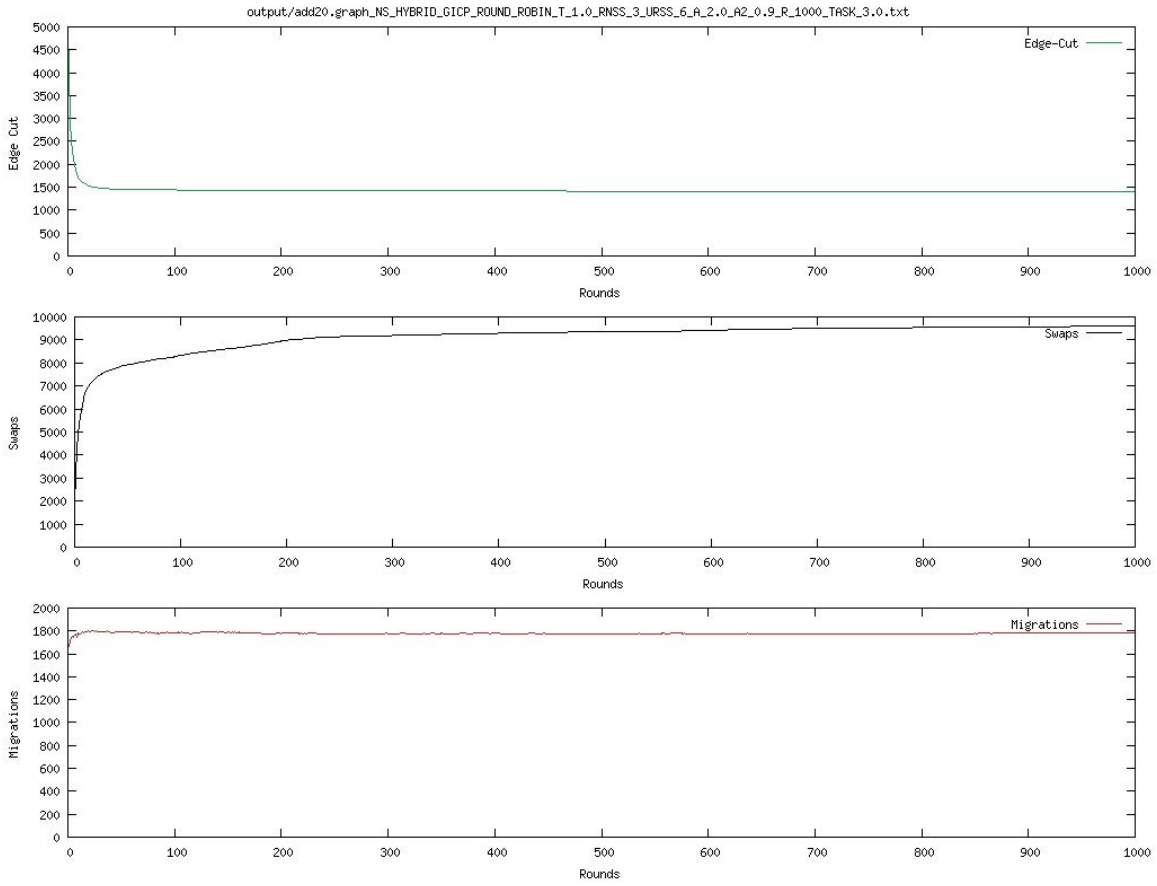

output/add20.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_1.0_RNSS_3_URSS_6_A_2.0_A2_0.99_R_10000_TASK_2.1.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds


output/add20.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_2.0_D_0.003_RNSS_3_URSS_6_A_2.0_R_1000_RA_500_TASK_2.2.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

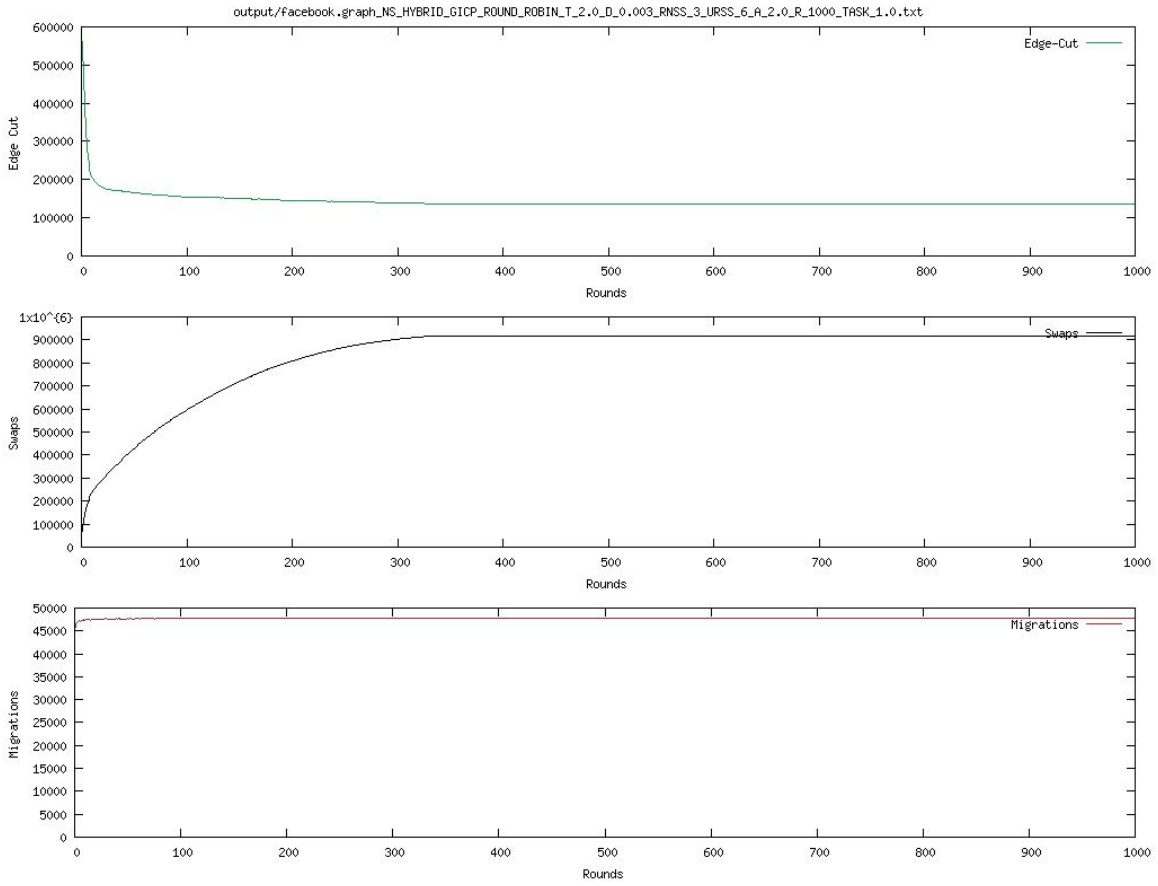

output/add20.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_2.0_D_0.003_RNSS_3_URSS_6_A_2.0_R_10000_RA_500_TASK_2.2.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

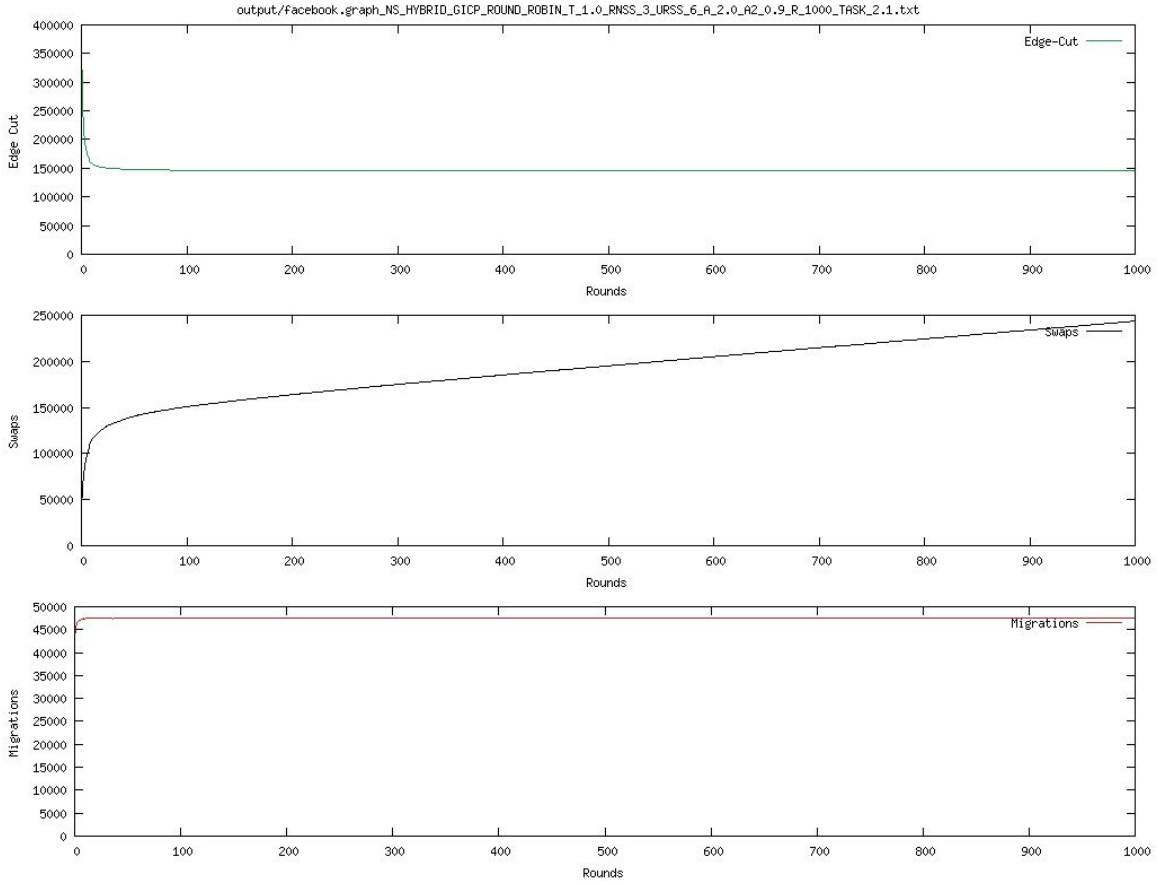

output/add20.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_2.0_D_0.005_RNSS_3_URSS_6_A_2.0_R_9000_RA_300_TASK_2.2.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

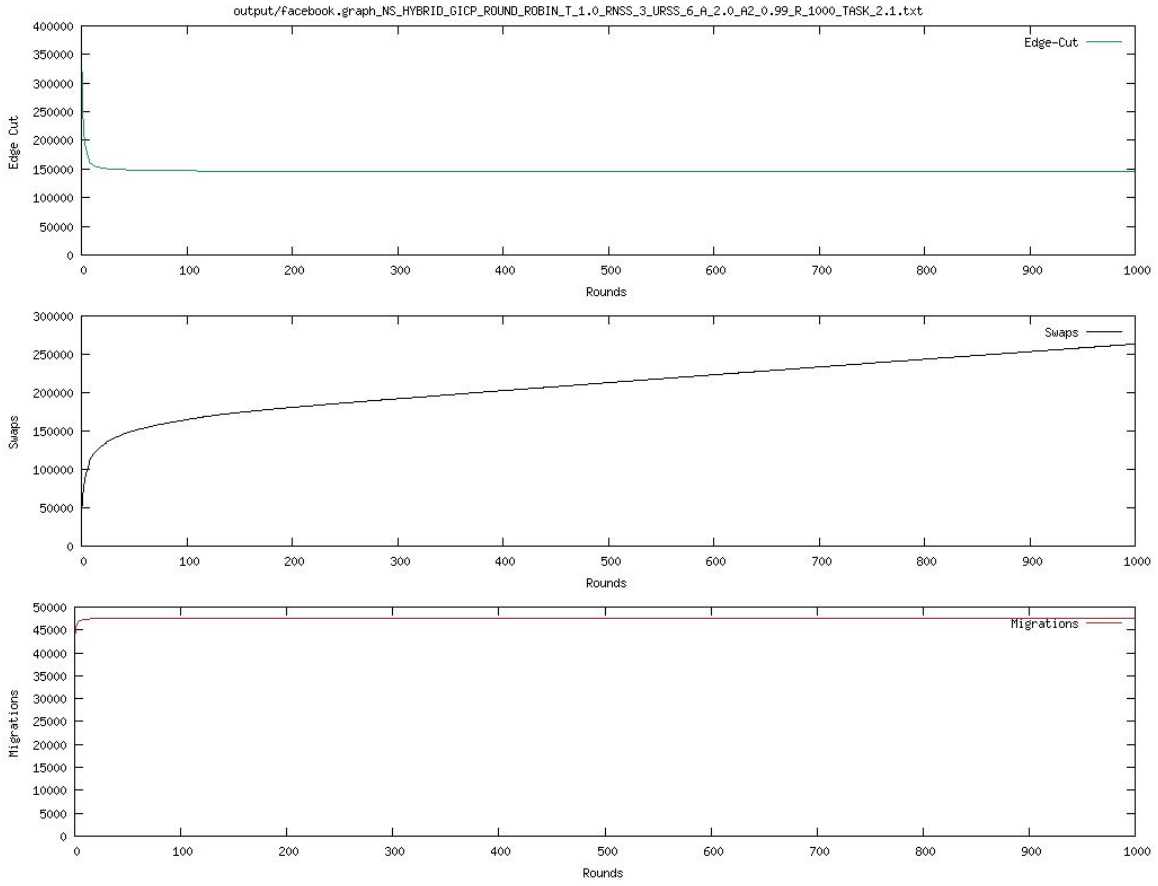

output/add20.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_2.0_D_0.01_RNSS_3_URSS_6_A_2.0_R_10000_RA_200_TASK_2.2.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

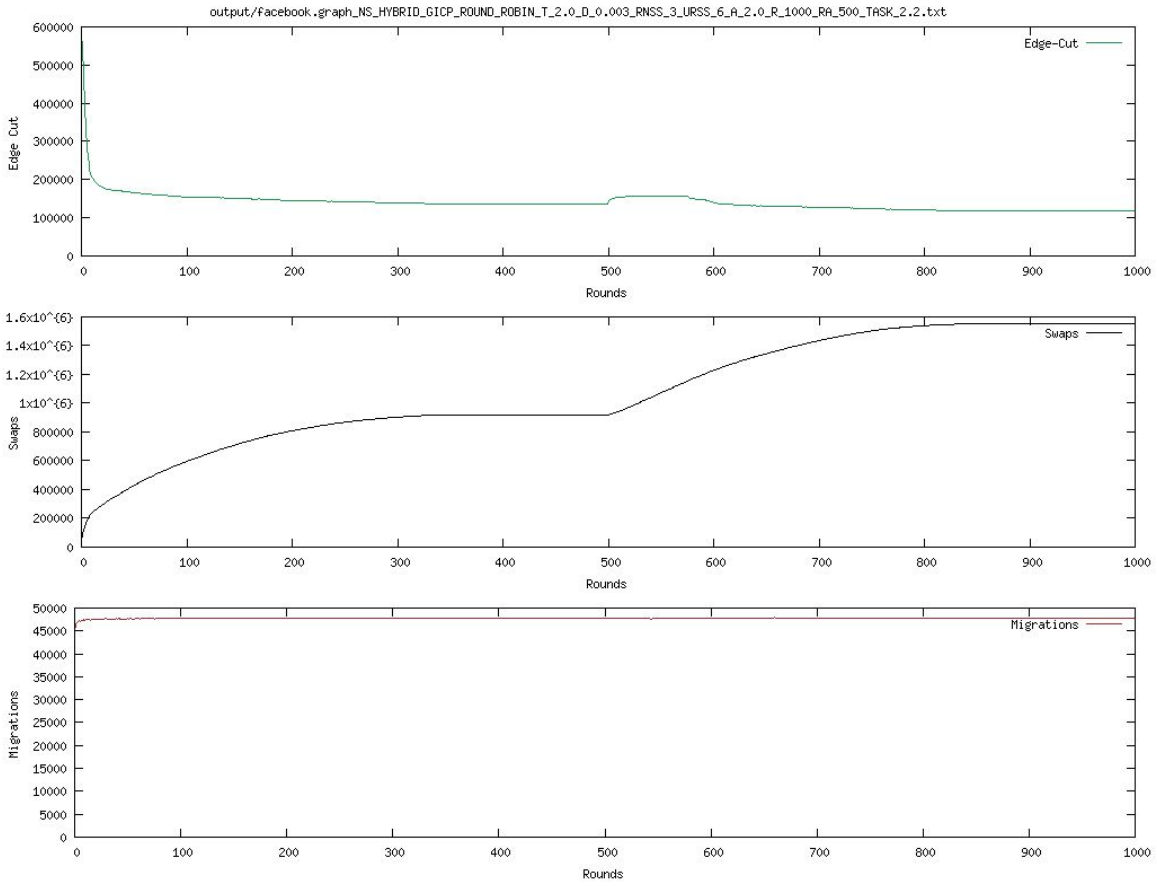


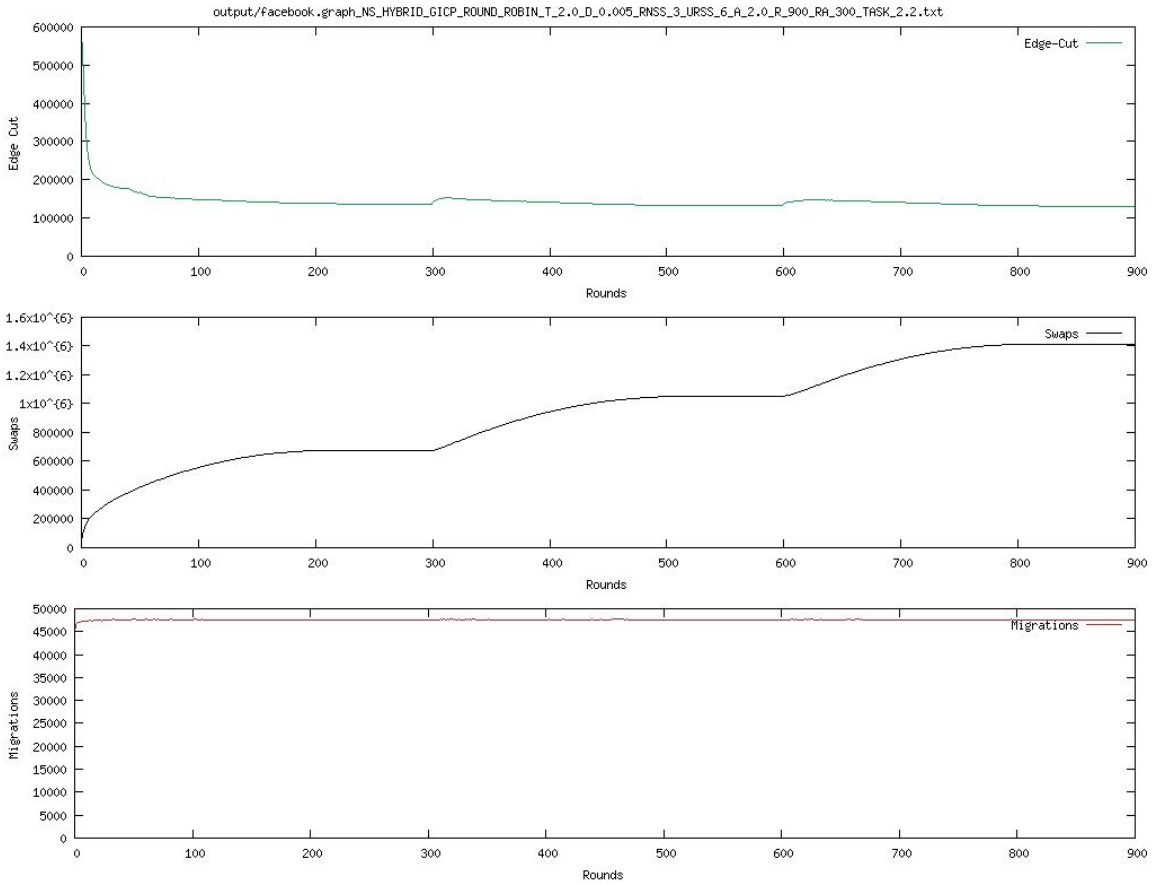
facebook.graph

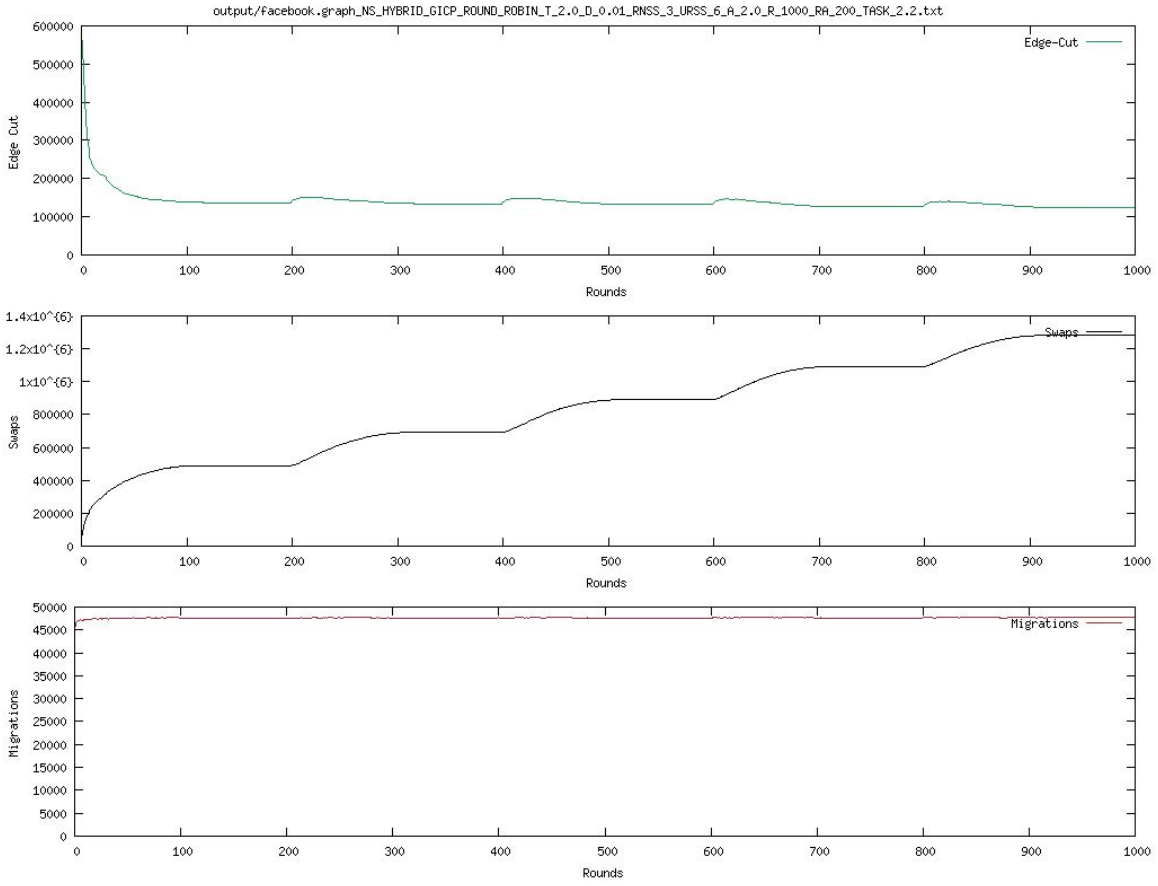

output/facebook.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_2.0_D_0.003_RNSS_3_URSS_6_A_2.0_R_1000_TASK_1.0.txt
Edge-Cut
Edge Cut
Rounds
Swaps
1x10^{6}
Migrations

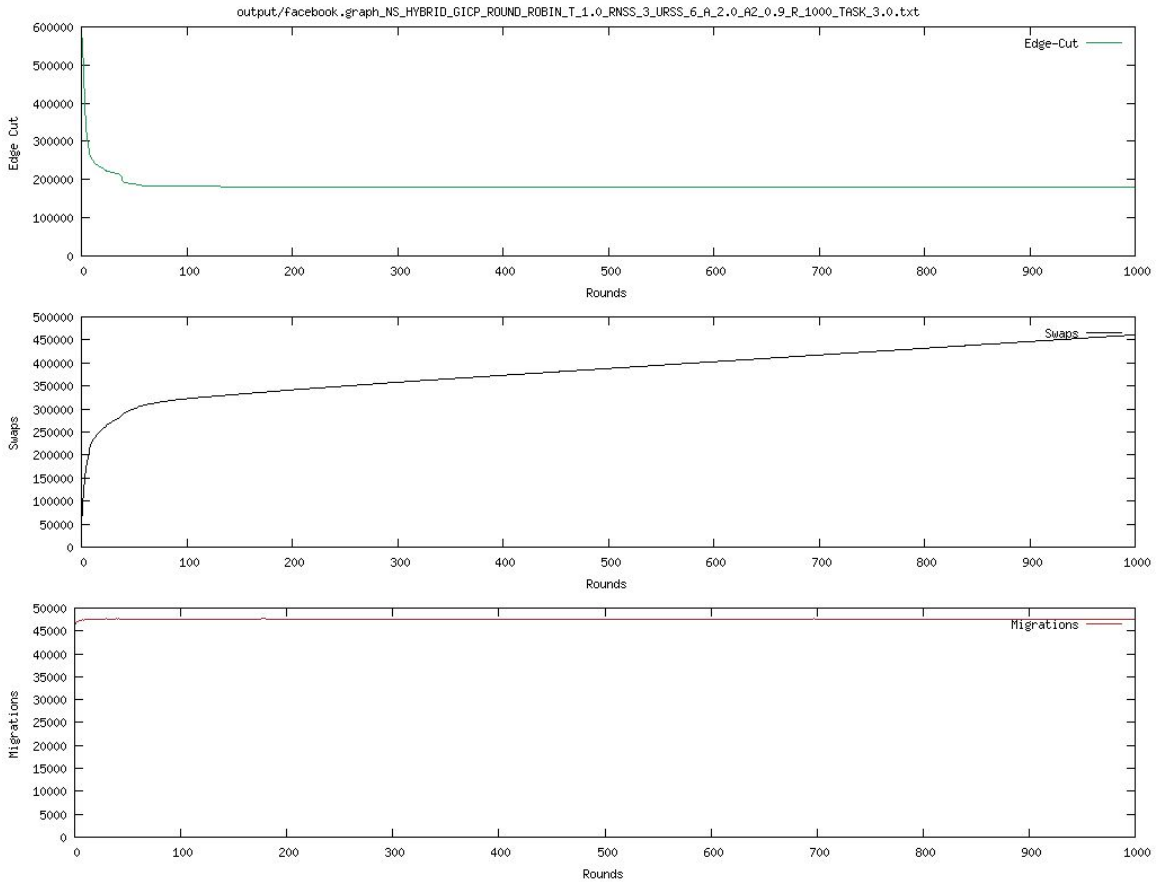

output/facebook.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_1.0_RNSS_3_URSS_6_A_2.0_A2_0.9_R_1000_TASK_2.1.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Migrations


output/facebook.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_1.0_RNSS_3_URSS_6_A_2.0_A2_0.99_R_1000_TASK_2.1.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

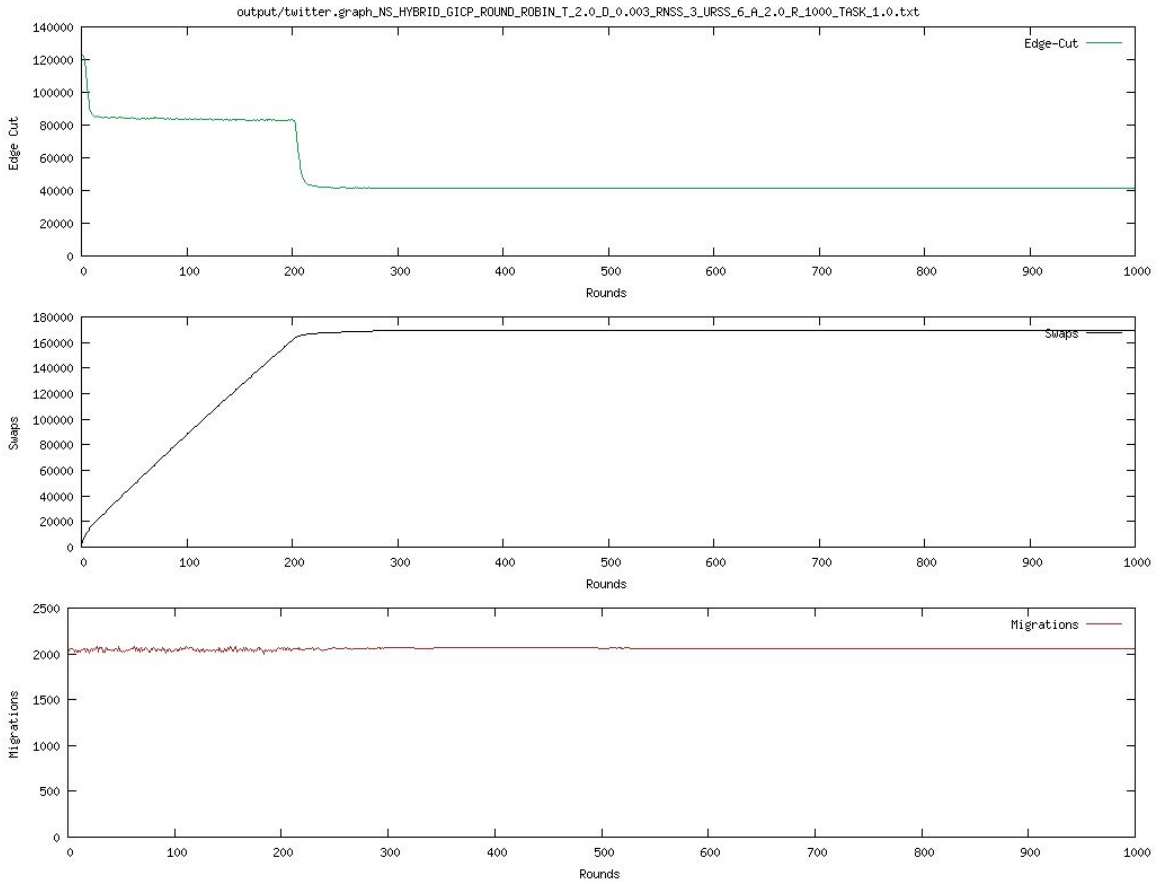

output/facebook.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_2.0_D_0.003_RNSS_3_URSS_6_A_2.0_R_1000_RA_500_TASK_2.2.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

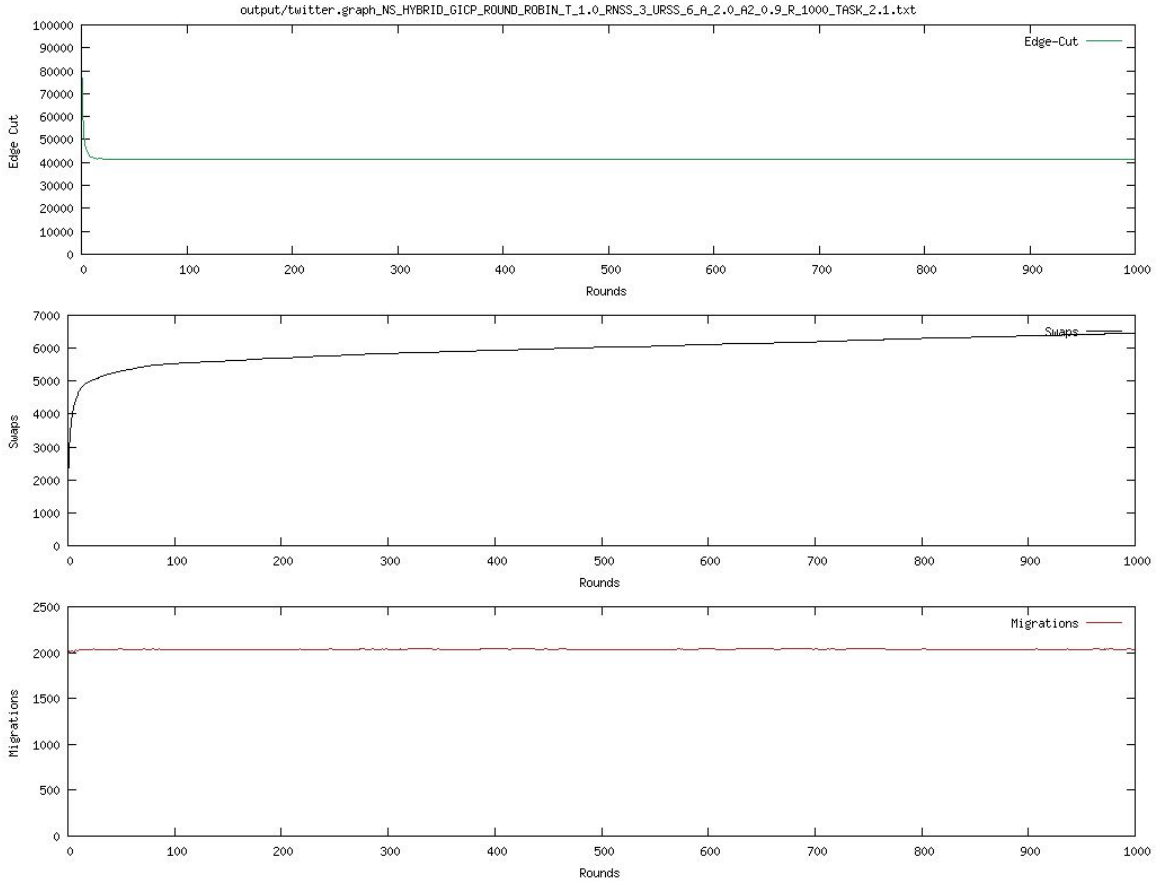

output/facebook.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_2.0_D_0.005_RNSS_3_URSS_6_A_2.0_R_900_RA_300_TASK_2.2.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

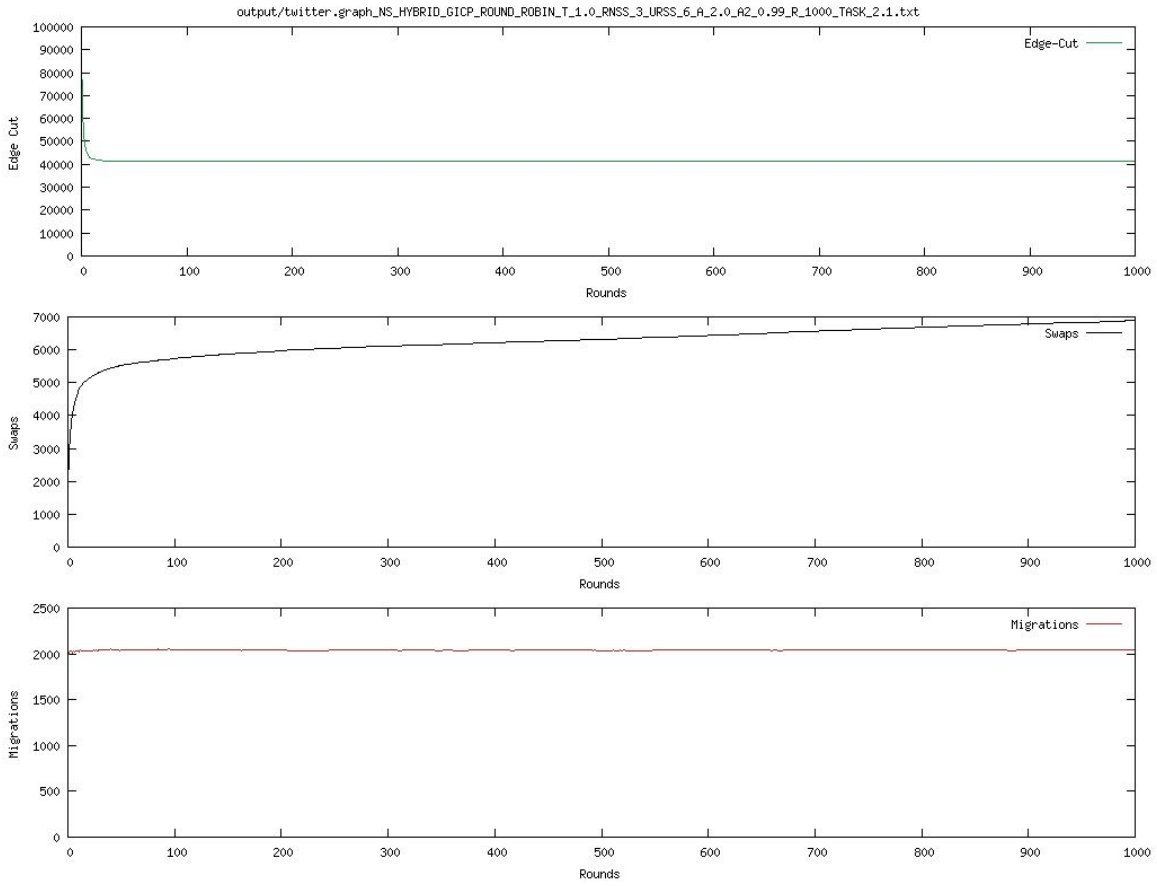

output/facebook.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_2.0_D_0.01_RNSS_3_URSS_6_A_2.0_R_1000_RA_200_TASK_2.2.txt
Edge Cut
Edge-Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

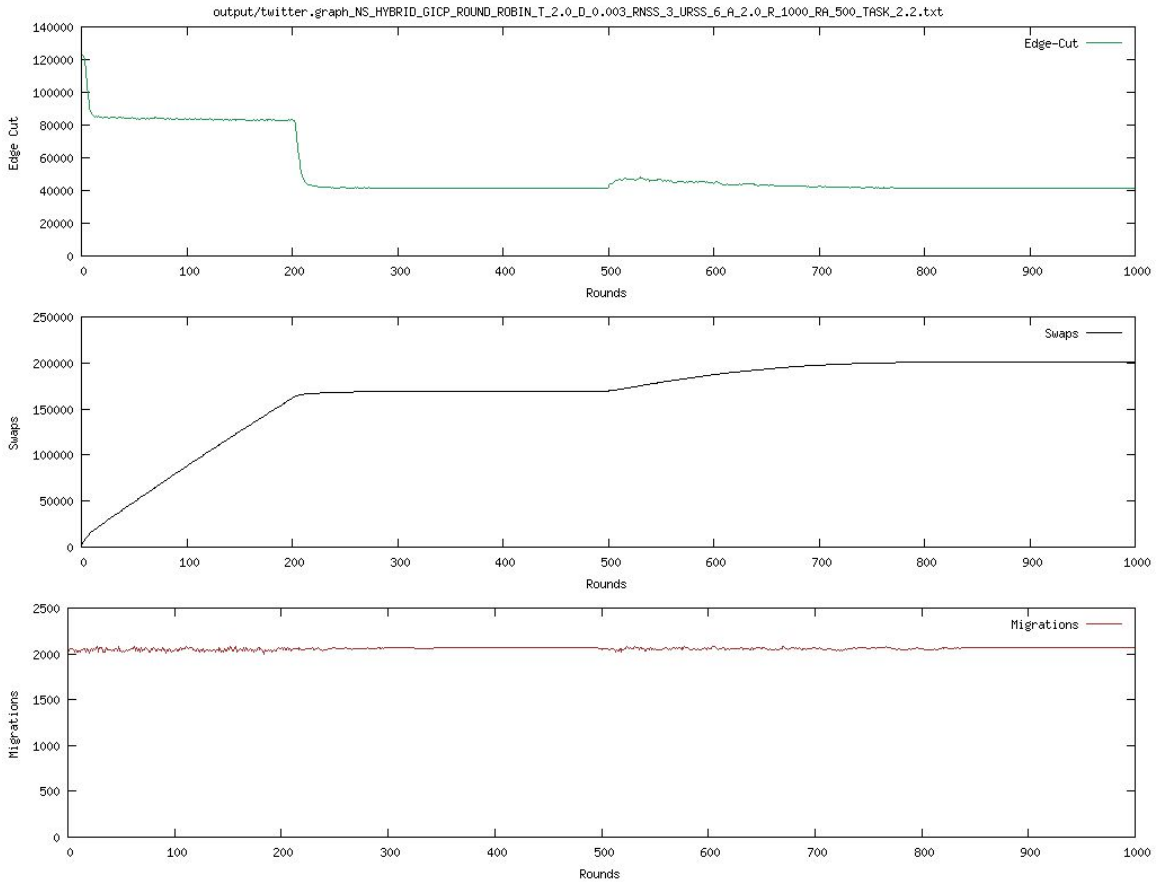


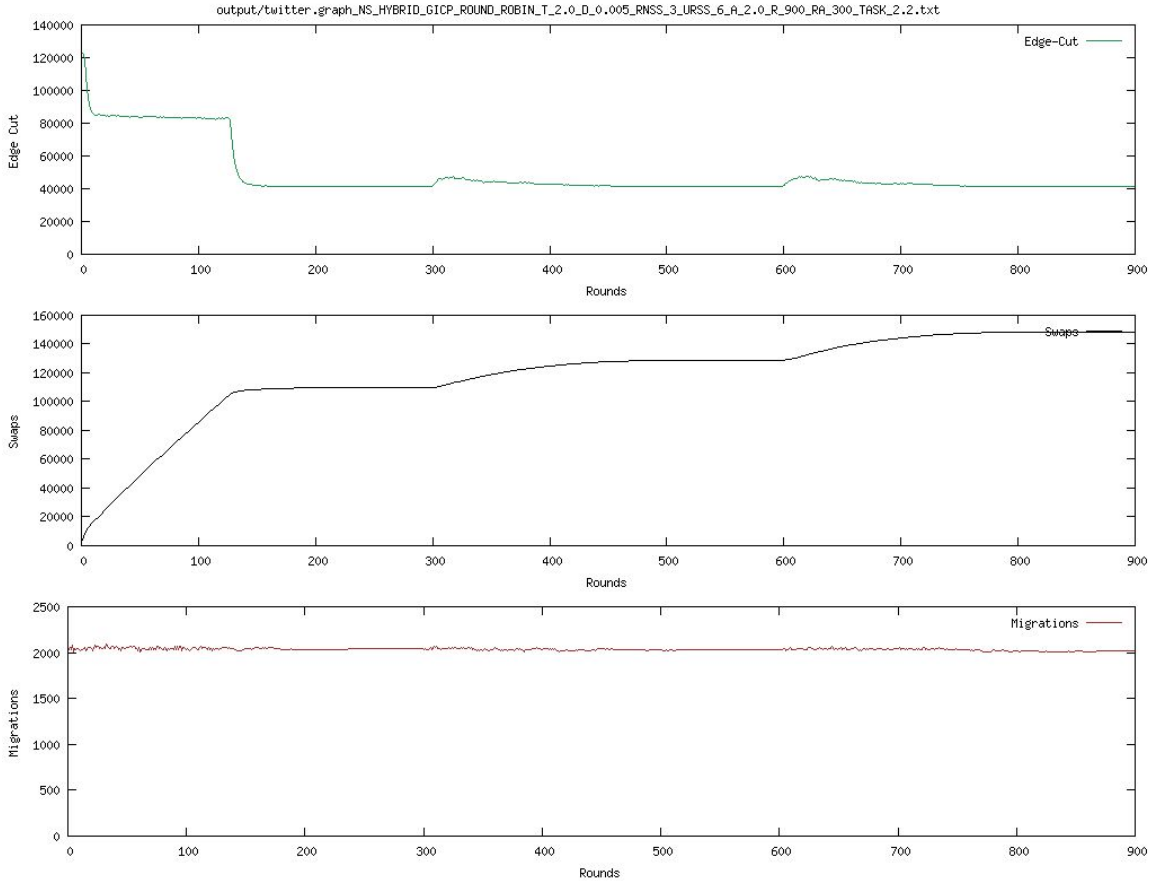
twitter.graph

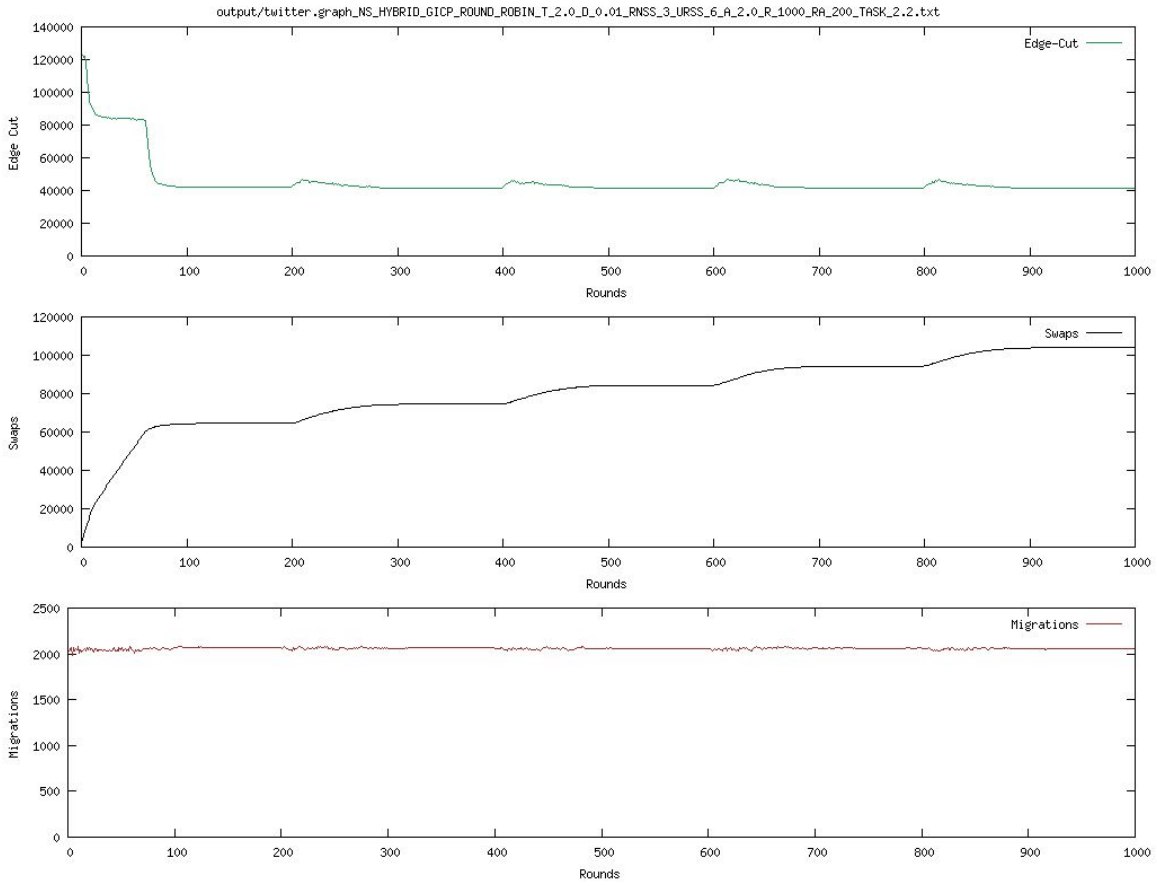

output/twitter.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_2.0_D_0.003_RNSS_3_URSS_6_A_2.0_R_1000_TASK_1.0.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds

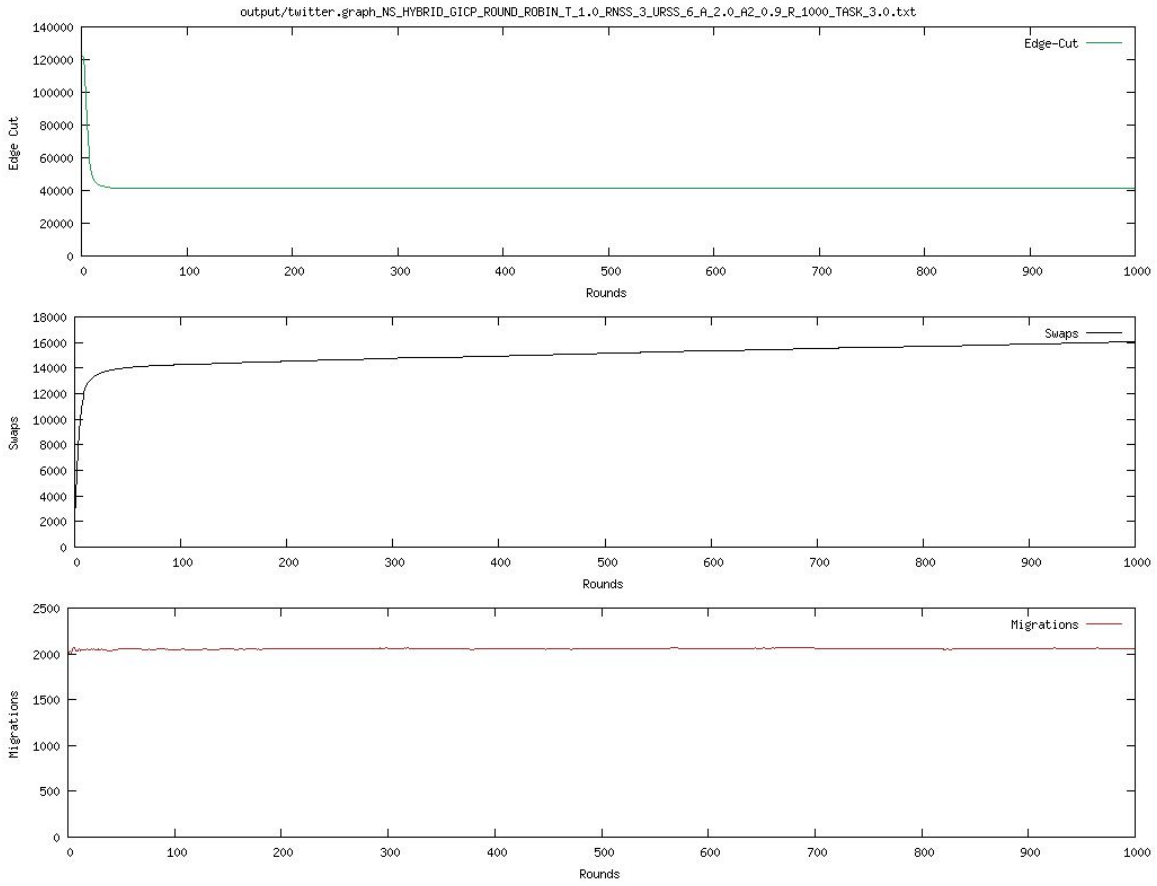

output/twitter.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_1.0_RNSS_3_URSS_6_A_2.0_A2_0.9_R_1000_TASK_2.1.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds


output/twitter.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_1.0_RNSS_3_URSS_6_A_2.0_A2_0.99_R_1000_TASK_2.1.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds


output/twitter.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_2.0_D_0.003_RNSS_3_URSS_6_A_2.0_R_1000_RA_500_TASK_2.2.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds


output/twitter.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_2.0_D_0.005_RNSS_3_URSS_6_A_2.0_R_900_RA_300_TASK_2.2.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds


output/twitter.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_2.0_D_0.01_RNSS_3_URSS_6_A_2.0_R_1000_RA_200_TASK_2.2.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds


output/twitter.graph_NS_HYBRID_GICP_ROUND_ROBIN_T_1.0_RNSS_3_URSS_6_A_2.0_A2_0.9_R_1000_TASK_3.0.txt
Edge-Cut
Edge Cut
Rounds
Swaps
Swaps
Rounds
Migrations
Migrations
Rounds