Virtual infrastructure partitioning and provisioning under nearly real-time constraints

Student:
Andrey Afanasyev

Andrey.Afanasyev@os3.nl

Supervisors:
dr. Z. (Zhiming) Zhao, dr. ir. A. (Arie) Taal, Mr H. (Huan) Zhou MSc

MSc Security and Network Engineering
Research Project 1 (#50)
Presentation

February 5, 2017

Introduction

- Virtual infrastructure partitioning according constraints of:
  - Application(Network - QoS),
  - Developer(geographical location, service usage and other KPI),
  - Data center (CPU, RAM, Storage usage - out of scope).
- Graph Partitioning problem
  - NP-hard problem,
  - Heuristic algorithms,
  - Big O notation (time complexity).

Outline

- Related Research
- Research Question
- Methodology
- Results
- Discussion

Related work

- By Buluç et al did a survey of recent trends in in computational methods, software and benchmarks.

  Buluç, Aydin & Meyerhenke, Henning & Safro, Ilya & Sanders, Peter & Schulz, Christian. (2016). *Recent Advances in Graph Partitioning.* 9220. . 10.1007/978-3-319-49487-6_4. (Accessed on 15-Jan-2017)

- In the paper of LaSalle et al were discussed and compared different approaches for parallelizing each of the three phases of multilevel graph partitioning: coarsening, initial partitioning, and uncoarsening. The design space of creating a multi-threaded graph partitioner was explored.

  LaSalle, Dominique, and George Karypis. *Multi-threaded graph partitioning.* Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on. IEEE, 2013

Research question

How profiling of the graph partitioning algorithms might improve fractioning of a virtual infrastructure?

- Which user and application constrains might be applied during profiling of the graph partitioning algorithms?
- What key algorithms are implemented for evaluation?
- Which graph partitioning algorithms are most desirable in the specific scenario's?

Methodology

This research was conducted in the period from 8 January 2018 till 11 of February 2018.

- In the first stage, constraints of user and application constraints which may influence the partitioning procedure while using implemented algorithms are organized.
- The focus of the second stage was in selecting graph partitioning software and investigating implemented algorithms. For this research was assumed that a virtual infrastructure is represented as a weighted undirected graphs. Only edge cutting algorithms are in scope of this research.
- By the third stage, an infrastructure topology is selected to benchmark only selected software implemented algorithms. Measurements done in terms of comparing algorithms time execution on defined topology. No new or existing algorithms are created or coded during this stage.

User and Application constraints

Stage 1 Result

User Constrains:

- Speed requirements (How fast network need to be partitioned?)
- **Network costs** (How to reduce network costs?)
- Graph segments relocation (How to spread application across multiple geographical locations?)

Application Constrains:

- Links status (Weight of communicated data sizes.)
- Nodes status (Weight of nodes in terms of resources usage. Out of Scope)

Selecting graph partitioning software

Stage 2 Result

OpenSource software:

- Chaco v2.0: Latest stable version was released in 1996, Developed in C, Designed for UNIX.
- KaHIP v2.0: Latest stable version was released in 2017, Developed in C, Successfully compiled only on Linux.
- **METIS v5.1.0: Latest stable version was released in 2013, Developed in C, Can be compiled on Linux and Windows.**

METIS

Algorithms and Python Wrappers

Supports multilevel recursive-bisection, multilevel k-way partitioning schemes. Might be used for both ***edge cutting*** and node clustering.

- Fiduccia-Mattheyses algorithm
  - 1sided One-sided node-based refinement [default].
  - 2sided Two-sided node-based refinement.
- Greedy algorithm

Python wrappers for METIS are:

- NetworkX-METIS a NetworkX module add-on, with limited functionality.(2015)
- PyMetis 2016.2, it only wraps the most basic graph partitioning functionality.(2016)
- **METIS for Python, wraps full graph partitioning functionality.(2018)**

Experiment

Stage 3. Environment

Machine [1]

- Operating System:Ubuntu Server 16.04.3 LTS
- CPU: AMD A10-7870K Radeon R7, 12 Compute Cores 4C+8G
- RAM: 7106MiB System memory
- HDD: 512GB Crucial SSD
- NIC: 1x 1Gb/s

Scripts

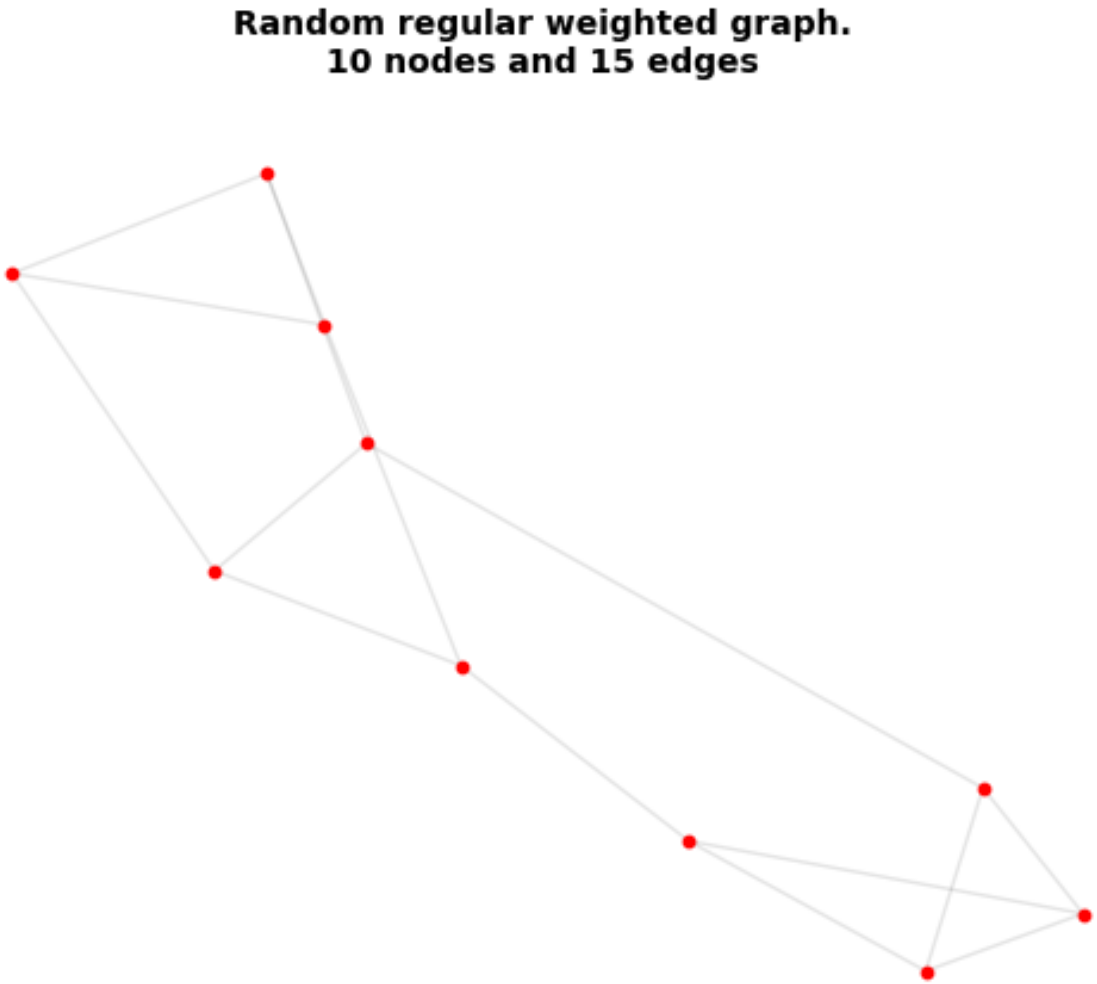
- Python 3.5.2 environment
- metis, matplotlib and all dependencies
- METIS 5.1.0

[1] lshw, iperf3

Experiment

Stage 3

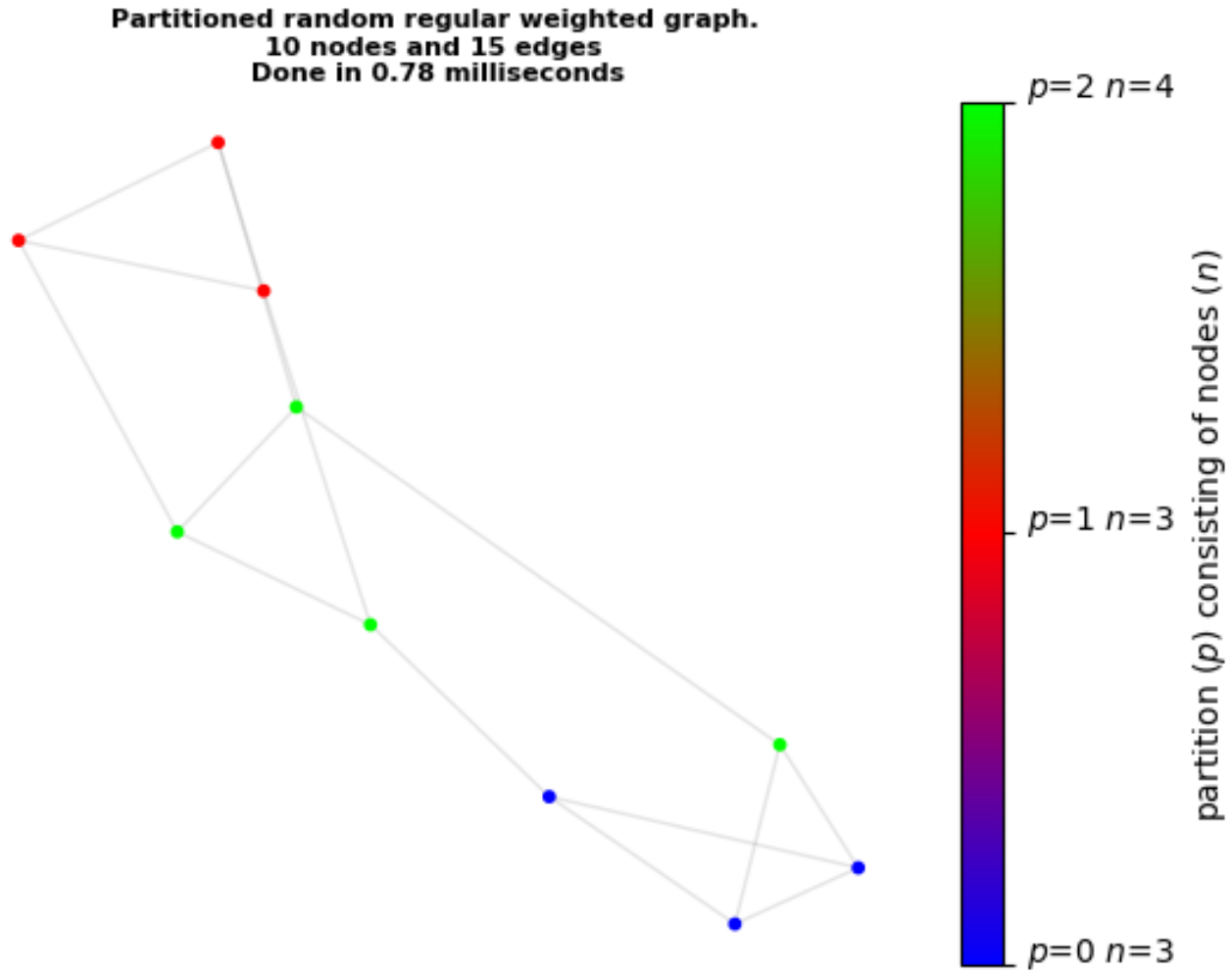
For this presentation following graph was use (maximum degree of 3):



Experiment

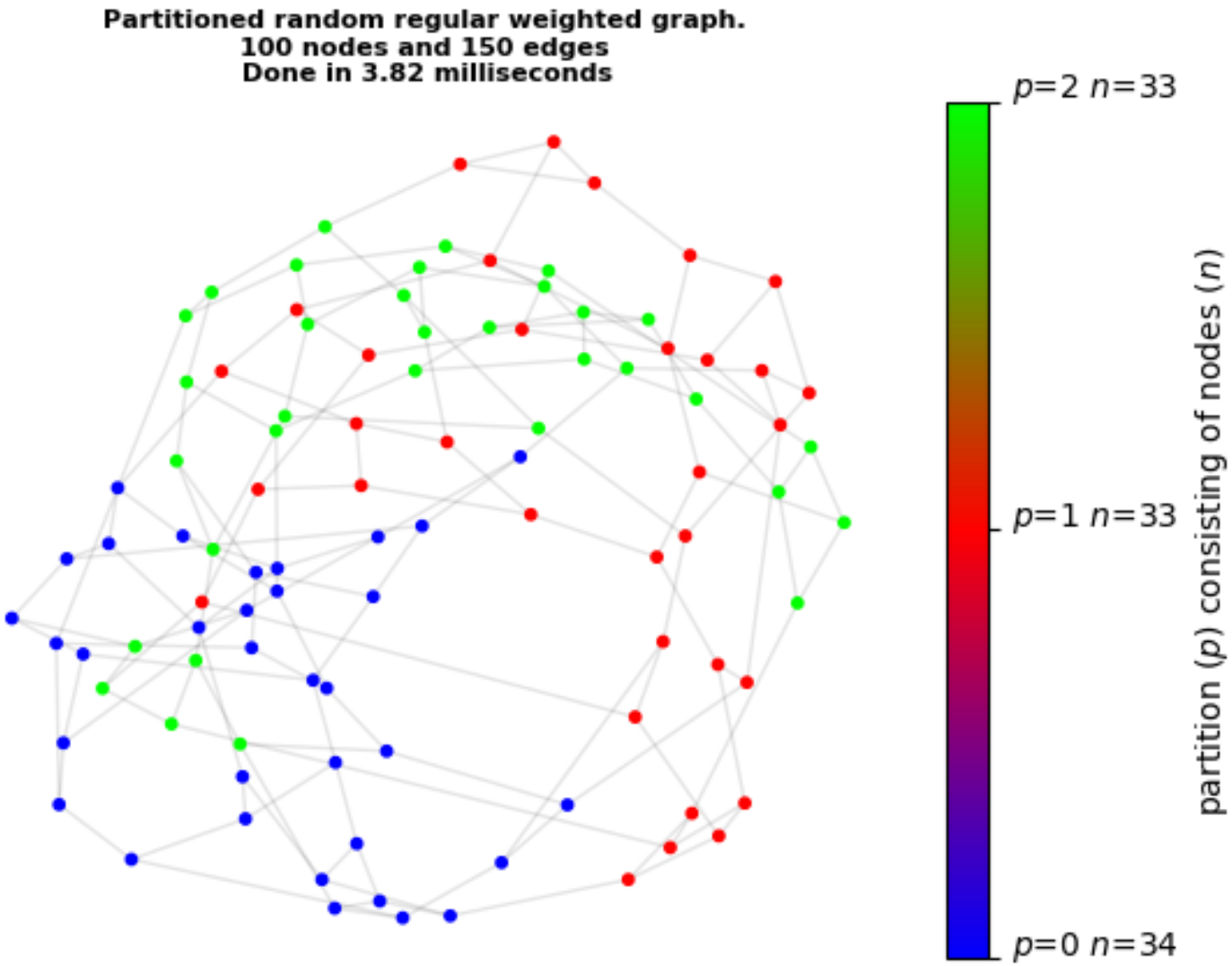
Stage 3

A possible representation of a partitioning:



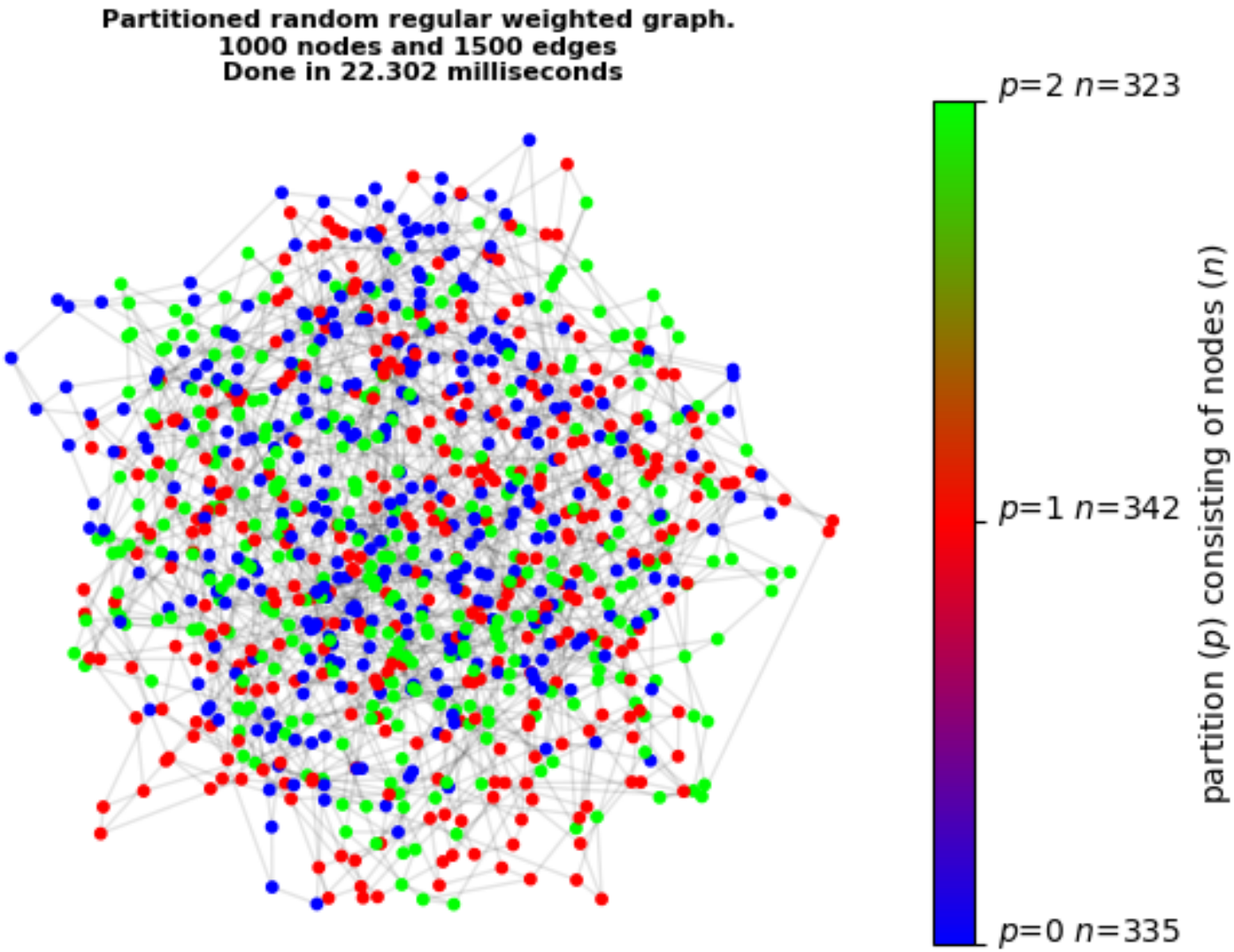
Experiment

Stage 3



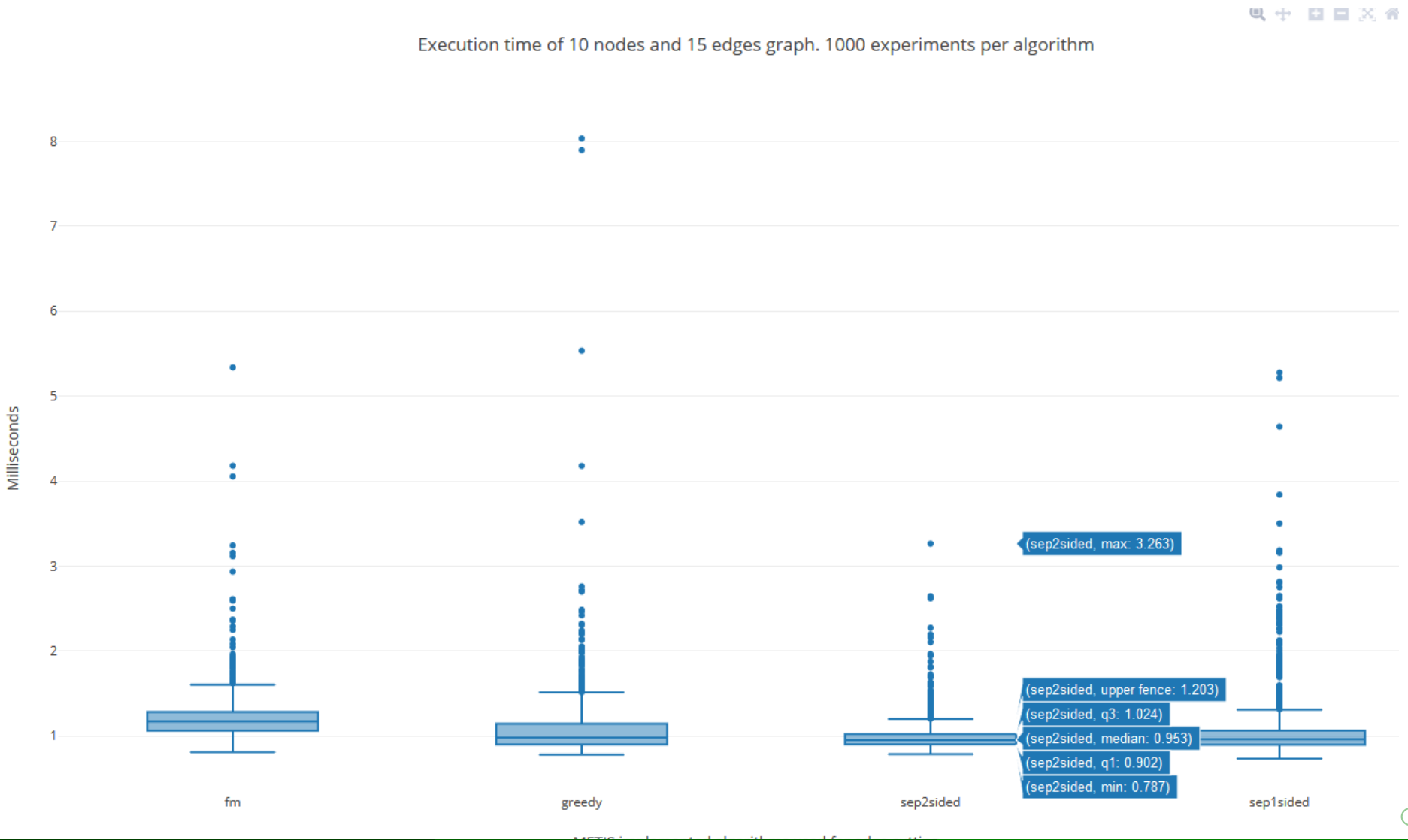
Experiment

Stage 3



Experiment

Stage 3 Benchmarks results



Observations

- Not all graph partitioning algorithms are implemented in METIS yet.
- Graph type detection system is not implemeted by NetworkX either by METIS.

Discussion

Conclusion:

- Selecting a correct partition algorithm might be profitable for developer to save costs.
- Fiduccia-Mattheyses with Two-sided node-based refinement requires less execution time to partition graphs with small amount of nodes and low maximum degree.

Future work:

- Detection of a virtual infrastructure topology as a graph type.
- Combination of nodes clustering and edges cutting.
- Researching and implementing more algorithms might be profitable.
- Additional research on user and virtual infrastructure application constrains.

Thank you

Questions?