Applying Q-Learning Agents to Distributed Graph Problems

Jeffrey McCrea, Munehiro Fukuda
School of Science, Technology, Engineering & Mathematics
Contact Email: mccreajeff@gmail.com

UNIVERSITY *of* WASHINGTON | BOTHELL

IARIA

About Me

Jeff McCrea – mccreajeff@gmail.com

> Master of Science in Computer Science & Software Engineering – University of Washington Bothell.

> Research Interests: Distributed computing, agent-based modeling, and reinforcement learning.

> Professional Experience: IT consultant specializing in enterprise IT environments.

> Academic & Industry Focus: Scalable computing solutions, multi-agent systems, and AI-driven optimization.

UNIVERSITY of WASHINGTON | BOTHELL

Distributed Systems Laboratory(DSL)

- Led by Prof. Munehiro Fukuda at UW Bothell, focusing on parallel & distributed computing.
- Current research includes agent-based computing, graph database, and large-scale simulations.
- Future directions focus on improving scalability, optimizing parallel performance, and integrating AI/ML techniques.
- https://depts.washington.edu/dslab/

UNIVERSITY *of* WASHINGTON | BOTHELL

Introduction - Why Do We Need Q-learning for Graph Problems

- Large, complex graphs require scalable solutions
- Static frameworks struggle with dynamic graphs
- Q-learning agents can learn & adapt to changes dynamically

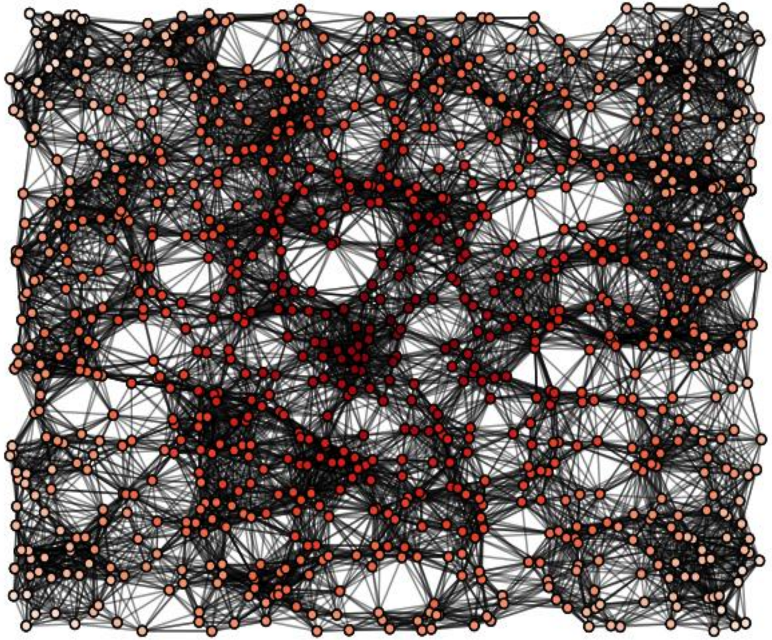


Image Source: Algorithms for Large-Scale Graph Processing

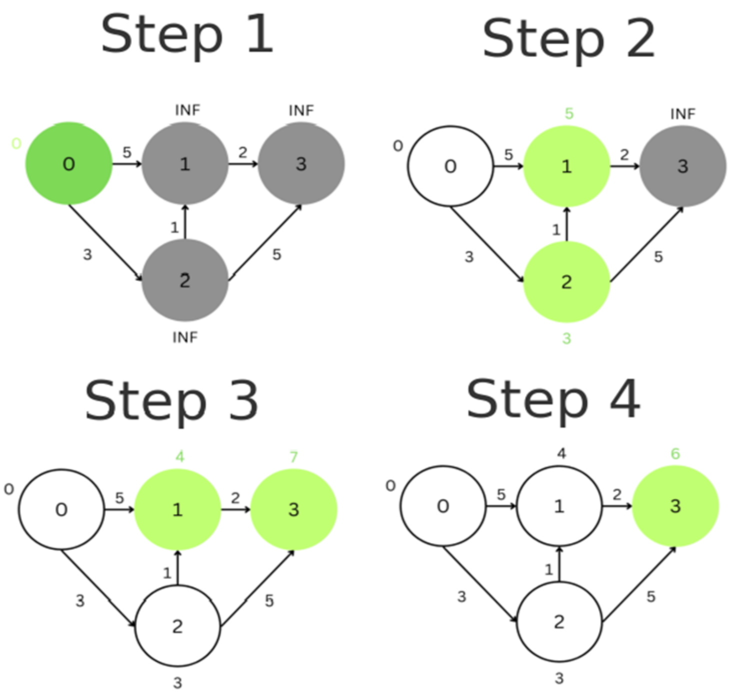
Introduction - Our Research Goals

Four main project goals:

1. Design agent-based Q-learning applications in MASS
    - Shortest Path
    - Closeness & Betweenness Centrality
2. Improve scalability & adaptability in dynamic graphs
3. Leverage MASS’ multi-agent capabilities to improve performance
4. Evaluate MASS agent-based machine learning capabilities

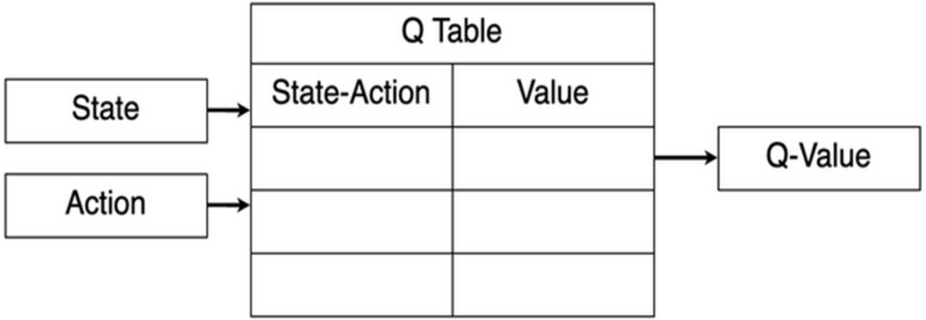
Background – Traditional Graph Computing vs. Q-learning

- Google's Pregel & Spark GraphX
  - Static, precomputed models
- Q-learning
  - Dynamic, adaptive, and reinforcement-driven
- Research Gap
  - Focuses on small, static graphs
  - Require preprocessing and computation to be effective



UNIVERSITY *of* WASHINGTON | BOTHELL

Related Work – How Q-learning Works



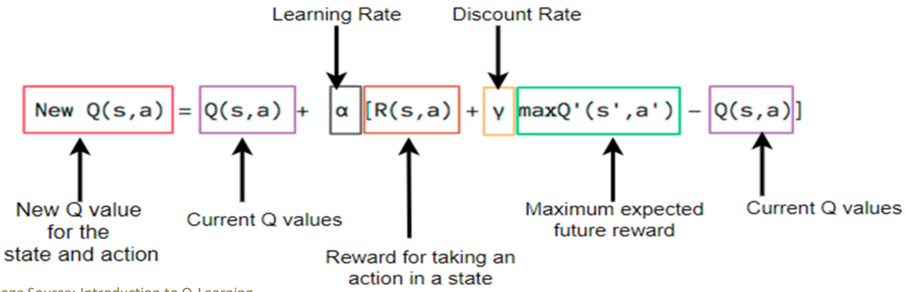
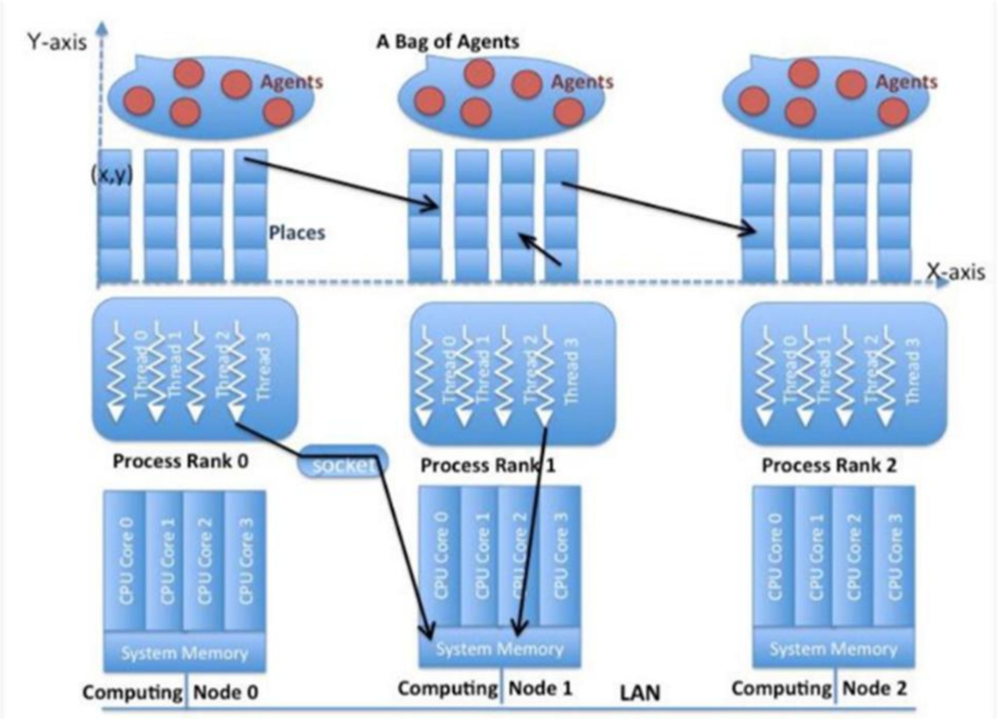


Image Source: Introduction to Q-Learning

- Model-free and off-policy
- Trial & error learning
- Q-Learning Process:
  1. Initialize Q-Table
  2. Set hyperparameters
  3. Choose an action
  4. Perform action & observe the outcome
  5. Update Q-value with:
  6. Repeat until training episodes are completed or convergence

Implementation - Multi-Agent Spatial Simulation Framework

> **Multi-Agent Spatial Simulation Library(MASS)**
> **Designed to facilitate spatial simulations and big data analysis in a parallel environment**
> **Two primary components:**
  – **Places – distributed individual dataset members**
  – **Agents – computation entities that traverse data**
> **Threads are assigned to Place objects and can communicate with other Places and agents that reside on them**



Implementation - Multi-Agent Spatial Simulation Framework

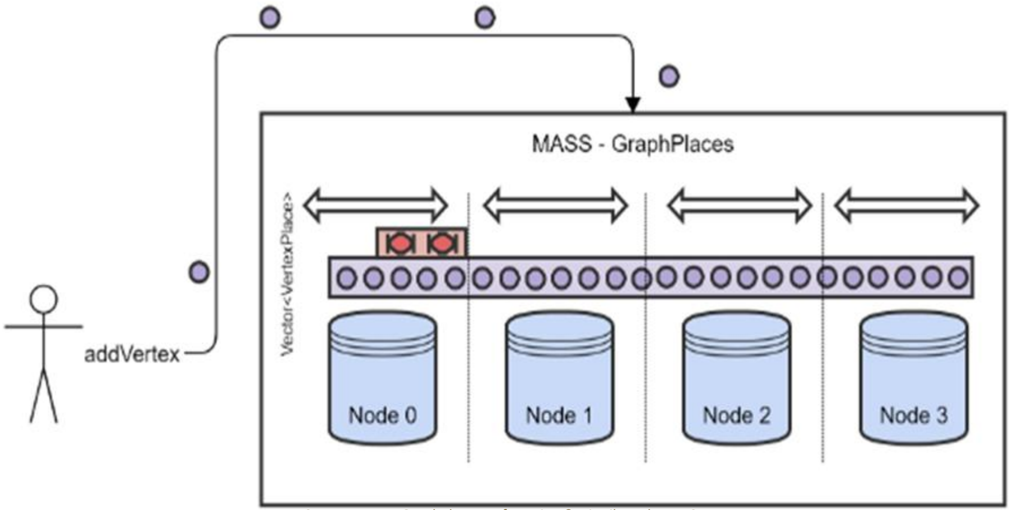


Image Source: Luger, GraphPlaces Refactoring & Distributed Data Structures

- MASS has been extended to support explicit graph structures

GraphPlaces
- Place -> VertexPlace

Agents
- Agent -> GraphAgent

Dynamic graph creation:
- AddVertex & AddEdge
- LoadDSL()

Balanced vertex distribution

Implementation - Q-Learning in MASS

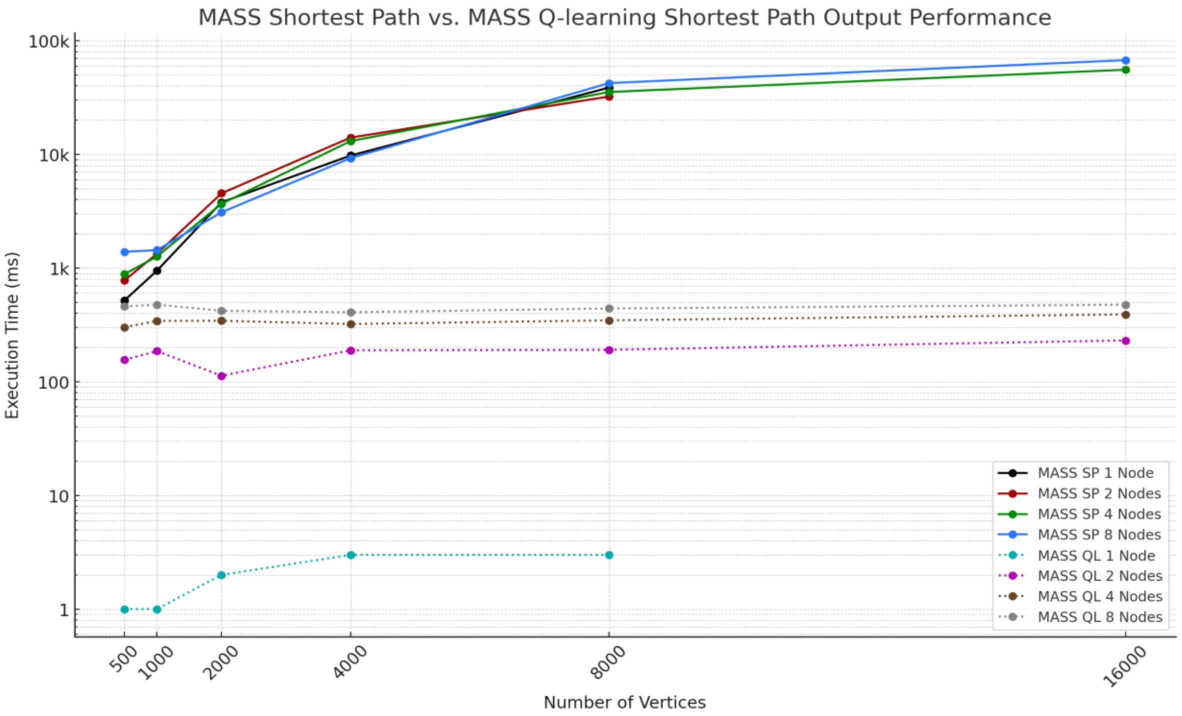
- Comprised of four main classes:
  - ShortestPath – driver
  - Node – environment
  - QLAgent – intelligent agent
  - PathAgent – path enumeration agent
- Agent-Based Learning: Q-learning agents explore a distributed graph, updating a shared Q-table to learn optimal paths.
- MASS-enabled Q-learning Improvements
  - Multi-agent Training
  - Distributed Q-table & Reward Window
  - Dynamic Hyperparameter Tuning

Evaluation - Experimental Setup

> Synthetic Graph Dataset:
  - 500-16,000 Nodes
  - Random graph generation
> Road Network Dataset:
  - 1,861-19,096 Nodes
  - OpenStreetMap data
> Synthetic Centrality Dataset:
  - 8-256 Nodes
  - Random Graph Generation
> Computing Cluster:
  - 8 VMs, Intel Xeon Gold 6130, 16GB RAM
> Performance measured in training time & adaptability

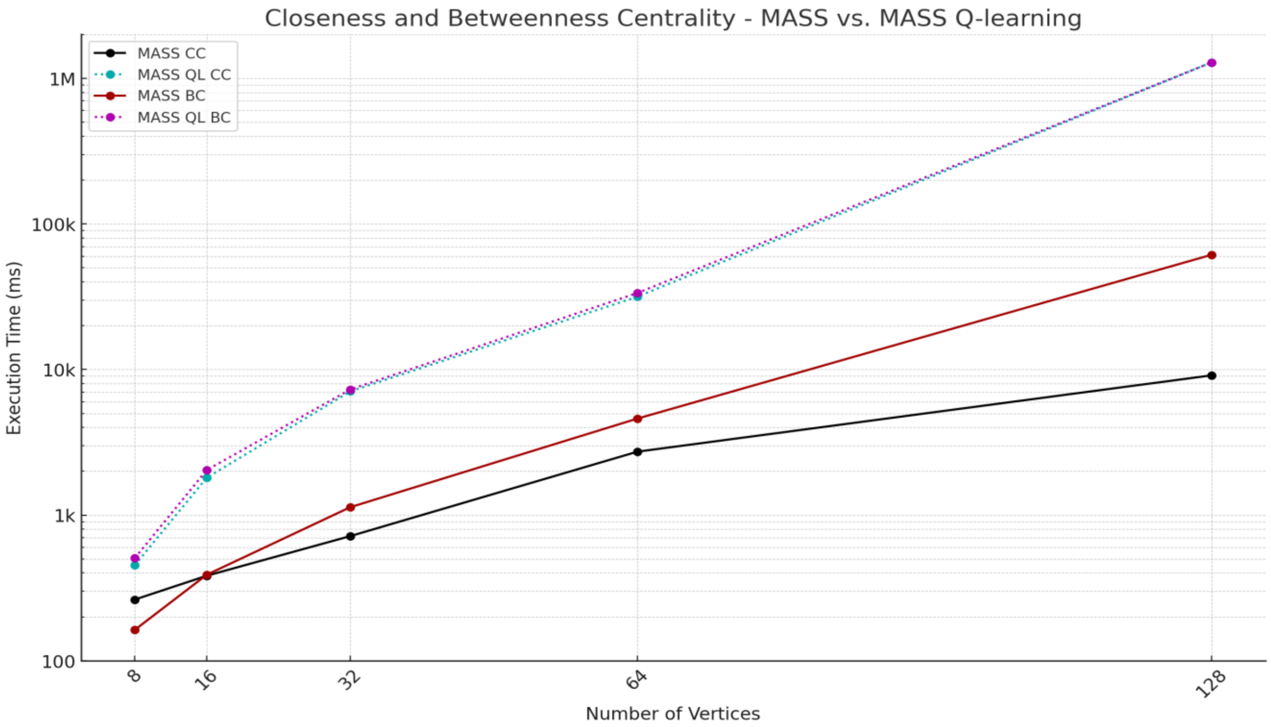
UNIVERSITY *of* WASHINGTON | BOTHELL

Evaluation – Q-learning Shortest Path



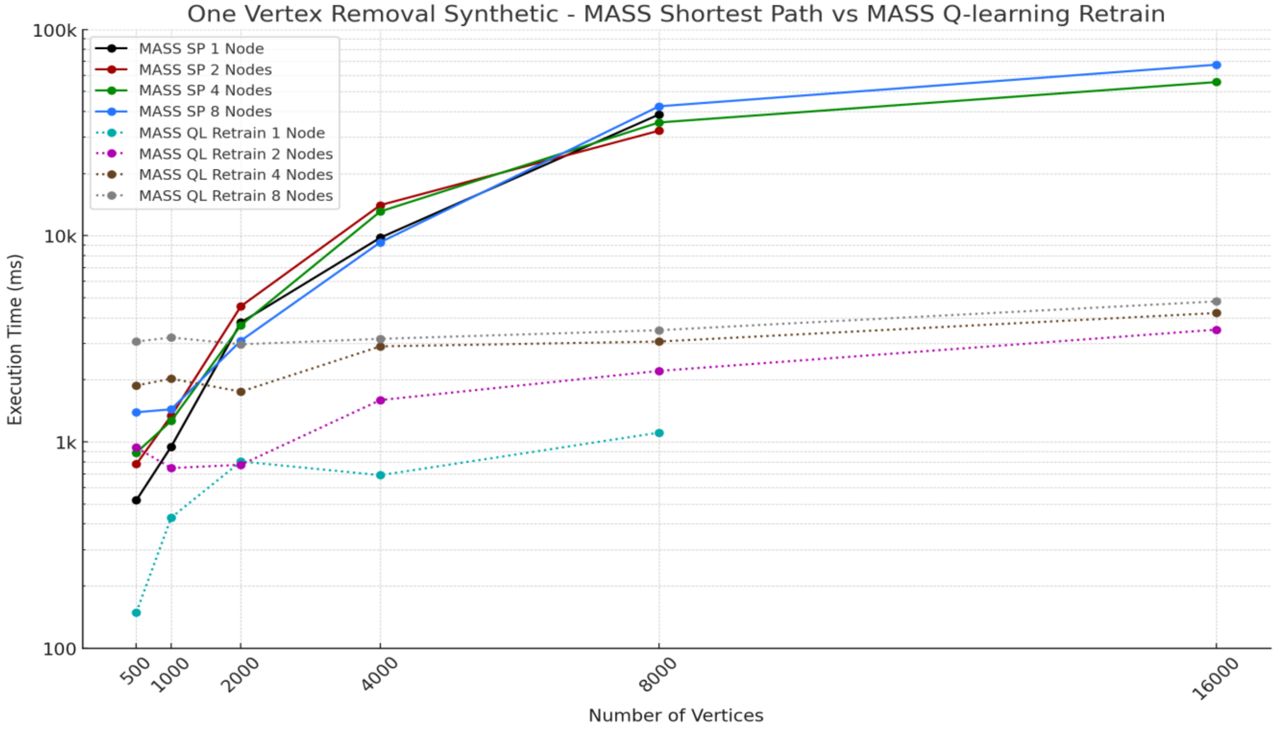
- Single node is optimal up to 8,000 nodes
- Above 8,000, multi-node execution is required
- Multi-agent training cuts training time by 190%

Evaluation – Closeness & Betweenness Centrality



- Performs well on small graphs
- Quadratically scaling as size grows – inefficient for large graphs
- High memory demands limit scalability

Evaluation – Dynamic Graph Adaptability



- Handles small changes efficiently
- Outperforms static methods in single-node removal
- Requires more significant retraining for larger topology shifts

Conclusion – What We Achieved

- Shortest path performance gains on static and dynamic graphs; centrality still needs optimization
- MASS-enabled features significantly improved performance
- Multi-agent training → 190% reduction in training time
- Distributed reward window → faster convergence
- Dynamic hyperparameter tuning → self-optimizing agents

Conclusion - Challenges & Future Work

Challenges:

- Scalability of centrality metrics
- Long training time for large graphs
- Necessity of hyperparameter fine-tuning for different graph types

Future Work:

- Graph Convolutional Networks (GCNs)
- Improved agent communication
- Online Q-learning for real-time updates

Q&A

Questions?

References

R. S. Xin, D. Crankshaw, A. Dave, J. E. Gonzalez, M. J. Franklin, and I. Stoica, "GraphX: Unifying Data-Parallel and Graph-Parallel Analytics," Feb. 11, 2014, arXiv: arXiv:1402.2394. Accessed: Jul. 19, 2024. [Online]. Available: http://arxiv.org/abs/1402.2394

G. Malewicz et al., "Pregel: a system for large-scale graph processing," in Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, Indianapolis Indiana USA: ACM, Jun. 2010, pp. 135–146. doi: 10.1145/1807167.1807184.

M. Fukuda, MASS: A Parallelizing Library for Multi-Agent Spatial Simulation. [Online]. Available: https://depts.washington.edu/dslab/MASS/

M. Kipps, W. Kim, and M. Fukuda, "Agent and Spatial Based Parallelization of Biological Network Motif Search," in 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems, New York, NY: IEEE, Aug. 2015, pp. 786–791. doi: 10.1109/HPCC-CSS-ICESS.2015.222.

Zhiyuan Ma and M. Fukuda, "A multi-agent spatial simulation library for parallelizing transport simulations," in 2015 Winter Simulation Conference (WSC), Huntington Beach, CA, USA: IEEE, Dec. 2015, pp. 115–126. doi: 10.1109/WSC.2015.7408157.

J. Woodring, M. Sell, M. Fukuda, H. Asuncion, and E. Salathe, "A Multi-agent Parallel Approach to Analyzing Large Climate Data Sets," in 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, USA: IEEE, Jun. 2017, pp. 1639–1648. doi: 10.1109/ICDCS.2017.106.

J. Zhang and Y. Luo, "Degree Centrality, Betweenness Centrality, and Closeness Centrality in Social Network," in Proceedings of the 2017 2nd International Conference on Modelling, Simulation and Applied Mathematics (MSAM2017), Bankog, Thailand: Atlantis Press, 2017. doi: 10.2991/msam-17.2017.68.