SIMULATION DESIGN AND ANALYSIS OF ENERGY-EFFICIENT DATA REDISTRIBUTION IN SELFISH BASE STATION-LESS SENSOR NETWORKS

Andre Chen

October 26, 2018

Overview

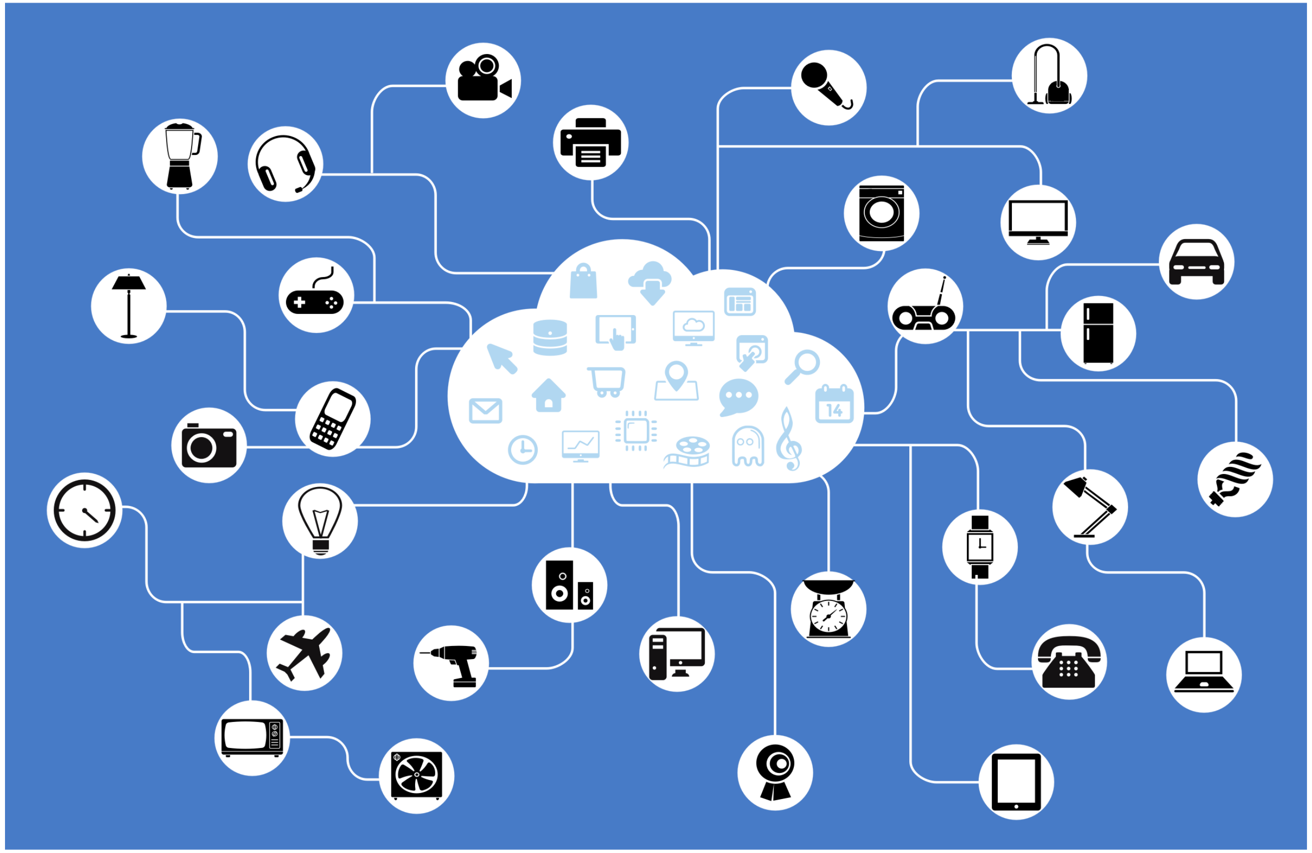
- Problem description and significance
- Assumptions
- Theory
- Simulation
- Results
- Conclusions

Problem Description

- Wireless sensor network with limited battery power and storage capacity collects data from environment
- Can we minimize energy consumption and store all data if all nodes are selfish and do not willingly cooperate?
- Can we design and run a simulation to empirically verify our theory?

Source (public domain): openclipart.org

Problem Significance

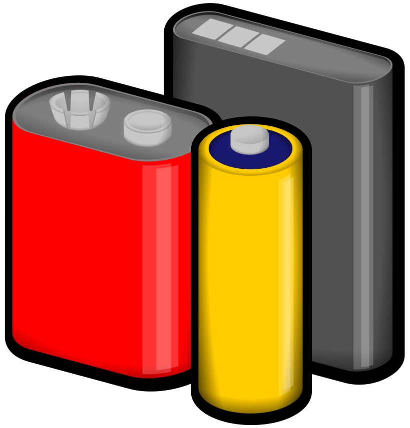


Source (public domain): openclipart.org

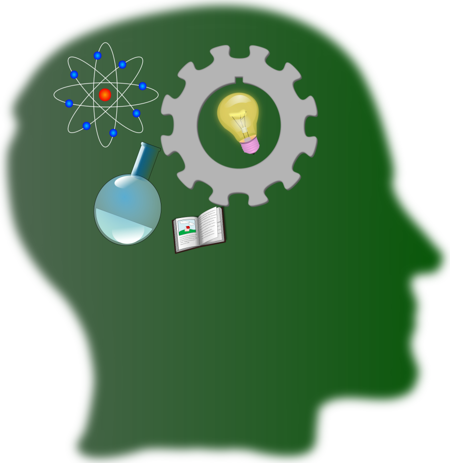
- Interconnected world
  - Data-driven decisions in network
  - Cooperation not guaranteed
- Reality vs. Simulation
  - Supports analysis and prediction
  - Mitigate risk by reducing uncertainty

Assumptions

- Biconnectivity
  - Prevents monopoly of control
  - Guarantees competition
- Feasibility
  - Sufficient battery power for participation
  - Guarantees all nodes present
- Knowledge
  - Everyone knows payment and utility
  - Everyone knows rules of the game



Source (public domain): openclipart.org



Source (public domain): openclipart.org

Assumptions *(continued)*

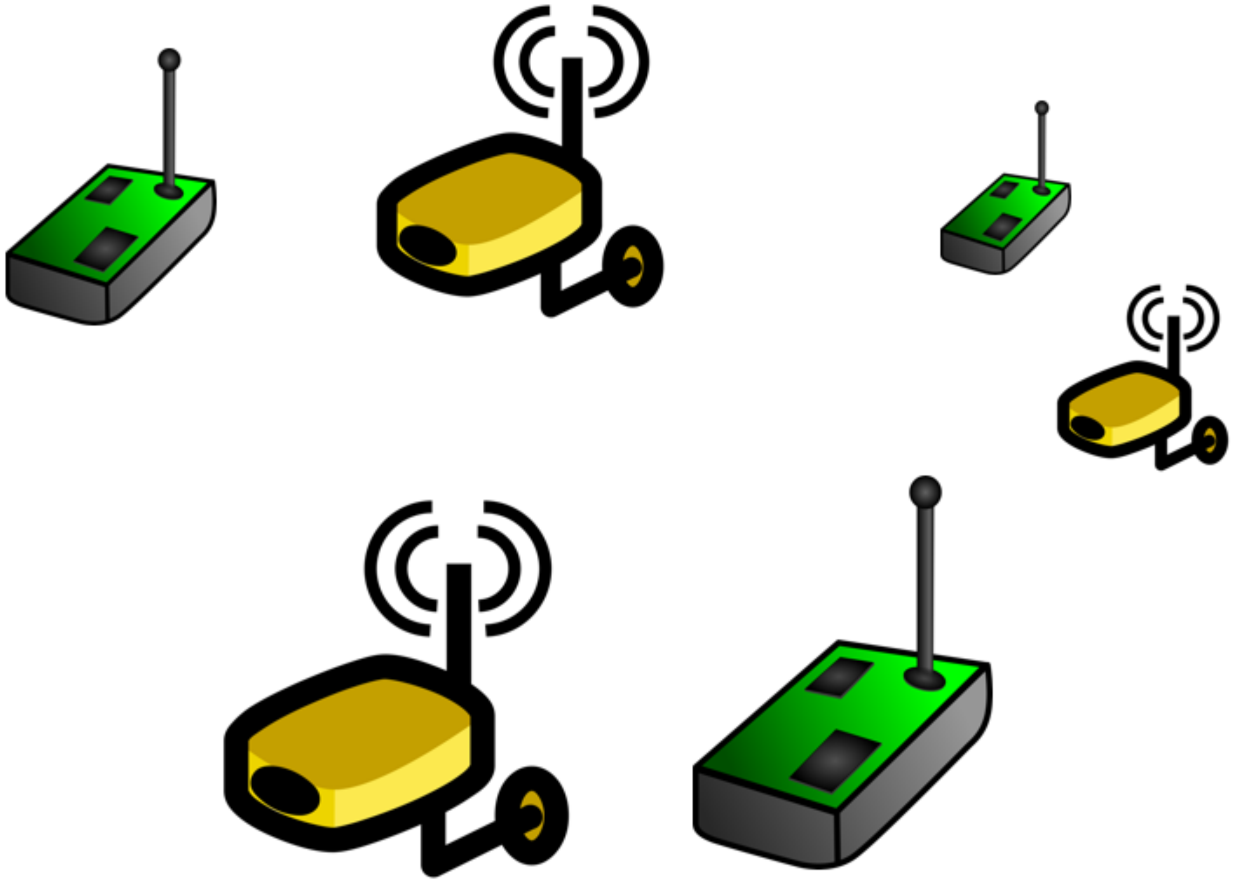
Source (public domain): openclipart.org

Source (public domain): openclipart.org

- Selfishness
  - No cooperation
  - Maximize self-interest
- Storage
  - Limited individual capacity
  - Sufficient total capacity
- Generator Incentivization
  - Generators already motivated to participate

Main Concepts

- Network consists of data generators and data storage nodes
- Storage nodes are selfish and must be incentivized to cooperate
- Goals
  - Store all data
  - Minimize energy consumption



Source (public domain): openclipart.org

Related Work

- Cooperation without storage (Tang, Jaggi, Wu, Kurkal, 2013)
  - All nodes cooperate
  - No storage costs

- Algorithmic mechanism design (Nisan, Ronen, 1999)
  - Algorithmic Analysis + Game Theory
  - Task scheduling and Least Cost Paths

- Selfishness with storage (Chen, Tang, 2016)
  - Theoretical solution developed
  - Empirical verification needed

Network Model

- Network is modeled by a weighted directed graph
- Each vertex is a network node with properties
  - Position in 2-D space, (x,y) coordinates
  - Transmission range
  - Cost parameters
    - ε_{elec} = cost to transmit one bit on a circuit
    - ε_{amp} = cost to transmit one bit on transmit amplifier
    - ε_{store} = cost to store one bit

- Each edge (n_1, n_2) has a weight equal to cost to route data from n_1 to n_2, where n_1 and n_2 are within transmission range
  - [Cost to **transmit** from n_1 to n_2] $+$ [cost to **receive** at n_2] + [cost to **store** at n_2]
  - Cost to store at n_2 is 0 if n_2 will relay data to another node

Algorithmic Mechanism Design Model

- Combine algorithmic analysis with mechanism design
  - Mechanism design – “reverse game theory”
  - Design the rules of the game to meet our needs

- Motivate selfish nodes by providing them with payment

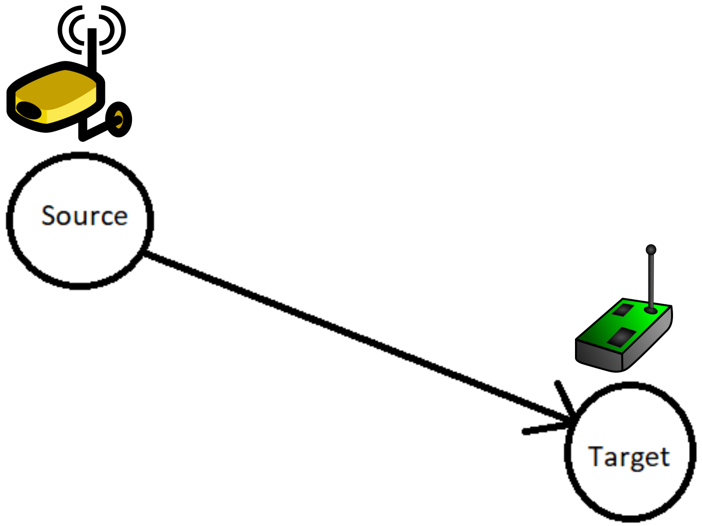
- Each node follows one of two strategies
  - Truth-telling – node reports true values of its cost parameters
  - Non-truth-telling – node lies about one of the true values of its parameters

- Selfish nodes select strategy to maximize their utility **regardless of its effect on the total energy consumption**

- Design a mechanism where node utility is maximized under truth-telling.

Costs

- Let b be the number of bits to transmit from source to target
- Transmission cost = $b * \varepsilon_{amp} * dist(n_{source}, n_{target})^2 + b * \varepsilon_{elec}$
  - Cost parameters from **source** node only
- Receiving cost = $b * \varepsilon_{elec}$
  - Cost parameter from **target** node only
- Storage cost = $b * \varepsilon_{store}$
  - Cost parameter from **target** node only



Payment and Utility

- Payment to each node = $p_i(\tilde{c}_i, c_{-i}) = c_{V-\{i\}} - (\tilde{c}_V - \tilde{c}_i)$
- Utility of each node = $\pi_i(\tilde{c}_i, c_{-i}) = p_i - c_i = c_{V-\{i\}} - (\tilde{c}_V - \tilde{c}_i) - c_i$
- Formulas and results (Chen, Tang, 2016) apply to **single data item** case
- Simulation shows formulas also apply to the **multiple data item** case

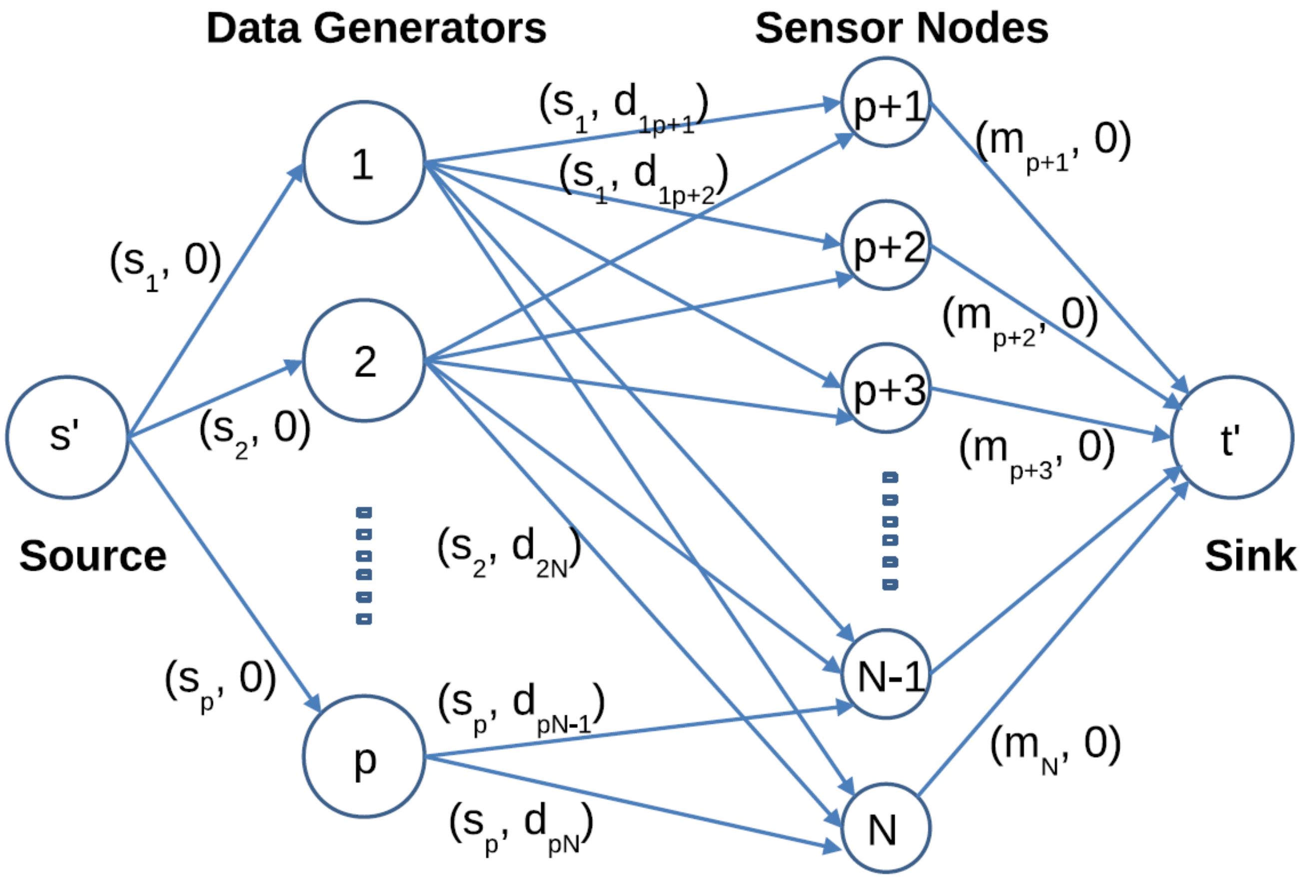
Symbol	Meaning
$\tilde{c}_i$	The sum of all costs that node i incurs based on its reported costs.
c_{-i}	The strategies of all nodes other than node i
$c_{V-\{i\}}$	The minimum total cost required to route all data when node i does not participate
$\tilde{c}_V$	The minimum total cost required to route all data when node i participates
c_i	The sum of all costs that node i incurs based on its true costs
$p_i(\tilde{c}_i, c_{-i})$	The payment owed to node i based on its reported costs $\tilde{c}_i$ and the strategies of all nodes other than node i (c_{-i})
$\pi_i(\tilde{c}_i, c_{-i})$	The utility of node i based on its reported costs $\tilde{c}_i$ and the strategies of all nodes other than node i (c_{-i})

Dominant Strategy

- A node’s strategy is **dominant** if it maximizes that node’s utility regardless of the strategies of all other nodes. (Nisan, Ronen, 1999)
- Theorem (Chen, Tang, 2016): In the multiple data items case, for each storage node, truthfully reporting each of its cost parameters is a dominant strategy.
  - The utility each node receives when telling the truth about its parameters is never less than the utility it receives when lying about its parameters.
  - Result is still true even if other nodes change their own strategies

Network Transformation

- Transform underlying graph into a flow network graph
  - Image from (Tang, Jaggi, Wu, Kurkal, 2013)



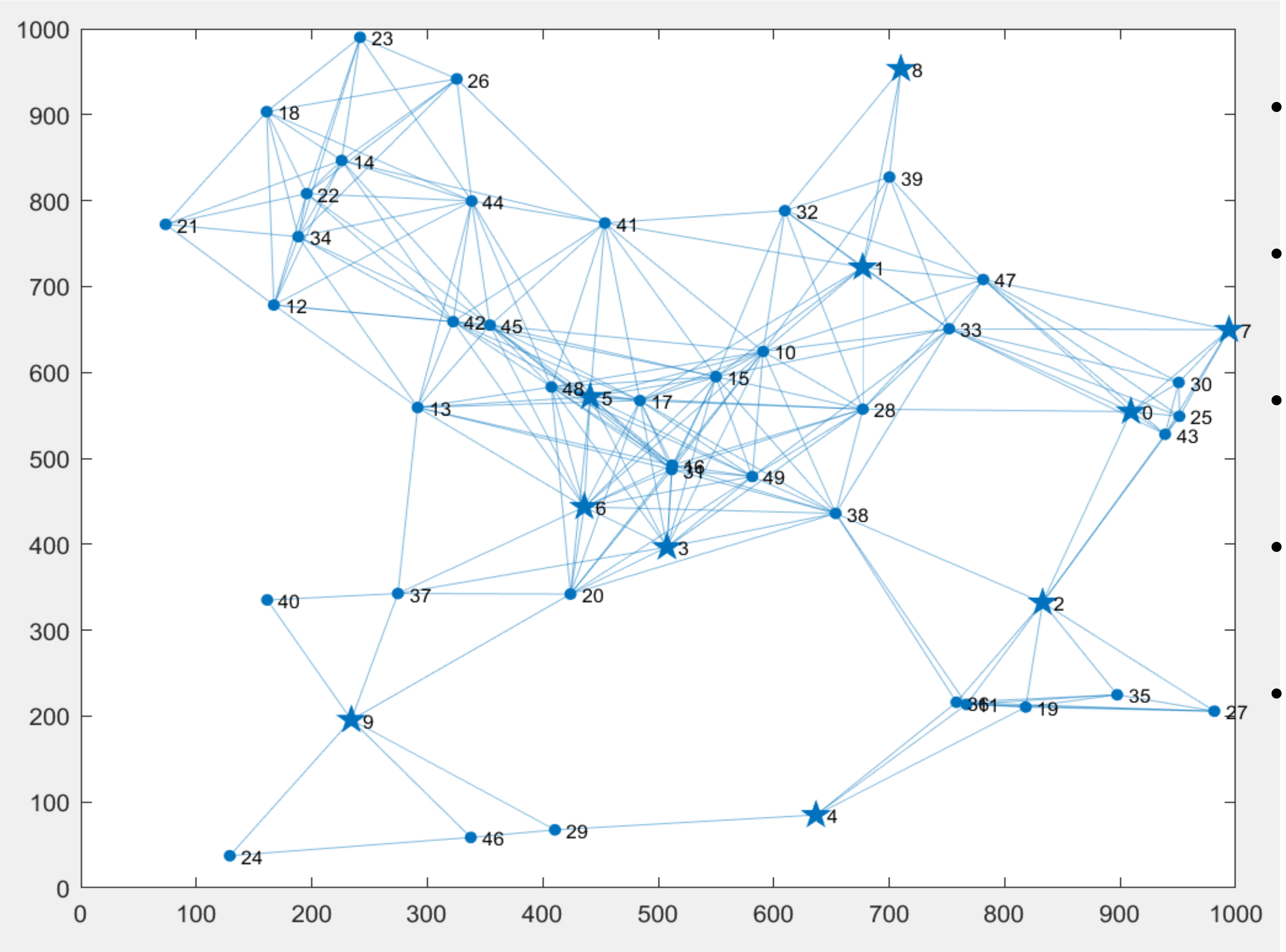
Network Properties

- Virtual source represents source of all data
  - Cost to transmit from source to data generators = 0
  - Source connected to every data generator

- Virtual sink represents all storage capacity
  - Cost to transmit from storage to sink = 0
  - Sink connected to every storage node

- There is an edge between each data generator and storage node
  - Edge weight is the cost to transmit data from generator to storage **along the least cost path (LCP)**

Network Visualization



- Nodes with ID 0-9 inclusive are data generators
- Nodes with ID 10-49 inclusive are storage nodes
- Nodes randomly distributed in 1000 * 1000 grid
- Line joining nodes means nodes are within transmission range
- Graph topology depends only on 2D position and transmission range, not cost parameters

Approach

1. Design simulation that models underlying network
2. Run simulation to compute true vs. reported utility
3. Modify parameters to verify each theoretical case
4. Analyze and interpret results

Simulation Overview

- Purpose – empirically verify theoretical results
- Language – Java 7
- Additional Dependencies
  - C program that implements Minimum Cost Flow
  - Compatible Linux distribution for running and testing BSD UNIX C programs
- Tested on Ubuntu 14.04

Simulation Design – Main Steps

1. Specify simulation parameters

2. Construct graph based on parameters
   a. Generate nodes and edges
   b. Construct graph
   c. Verify biconnectedness

3. Compute true and reported utilities
   a. Compute MCF cost with node removed
   b. Compute MCF cost with node present
   c. Compute costs based on reported parameters
   d. Compute true costs

4. Write results to CSV file for further analysis

Simulation Design – Data Structures

- Vertex represents each node in the network
    - Unique ID
    - 2D position
    - Transmission range
    - Cost parameters

- Edge represents each transmission path between adjacent nodes
    - Pair of Vertex objects
    - Unique Label
    - Weight

- Graph represents underlying network
    - Pair of two objects: collection of vertices, collection of edges
    - Internal methods for manipulating graph
    - Internal methods for verifying graph properties

Simulation Design – Algorithms

1. Graph Generation
   a. Generate nodes based on parameter input or randomized parameters
   b. Construct all edges based on node transmission ranges
   c. Construct graph based on nodes and edges
   d. Verify biconnectedness property by checking for articulation vertices

2. Generate inputs file to minimum cost flow (MCF) program.
   a. Set virtual source and construct edges to all data generators.
   b. Construct edges from generators to storage nodes.
   c. Construct edges from storage to virtual sink
   d. All edges incident to virtual nodes have no cost
   e. All edges between generators and storage nodes are the least-cost-paths from original network

Simulation Design – Main Algorithms (continued)

3. Compute utilities
    a. Compute MCF cost when node is removed
        i. Construct graph equivalent to original, but remove one node and its incident edges
        ii. Construct equivalent flow network based on this graph
        iii. Run MCF algorithm on equivalent flow network
    b. Compute MCF cost when node is present
        i. Construct equivalent flow network
        ii. Run MCF algorithm on equivalent flow network
    c. Compute cost incurred under lying
        i. Instruct a specific node to lie; all other nodes stay the same
        ii. Identify all data routing paths for which a node participates
        iii. Compute cost incurred based on reported parameter
    d. Compute cost incurred under truth-telling
        i. Instruct all nodes to be truthful
        ii. Identify all data routing paths for which a node participates
        iii. Compute cost incurred based on true parameters

Verification Cases

- Cases to investigate
  - Half-full homogeneous
  - Full homogeneous
  - Half-full heterogeneous
  - Full heterogeneous

- Total of 21 different sub-cases to verify in each main case
  - In all cases, show utility is maximized under truth-telling
  - Understand real-world implication of each case

Simulation Results Summary

- The utility under a truth-telling strategy is always greater than or equal to (i.e. never less than) the utility under a non-truthful strategy, regardless of the strategies of other nodes

- Parameter amp (ε_{amp}) has the most dramatic effect on true vs. reported utilities compared to the effects of scaling elec (ε_{elec}) and store (ε_{store}) parameters
  - Utility is more sensitive to changes in ε_{amp} compared to changes in ε_{elec} and ε_{store}
  - Effect of ε_{amp} scales quadratically, while effects of ε_{elec} and ε_{store} scale linearly

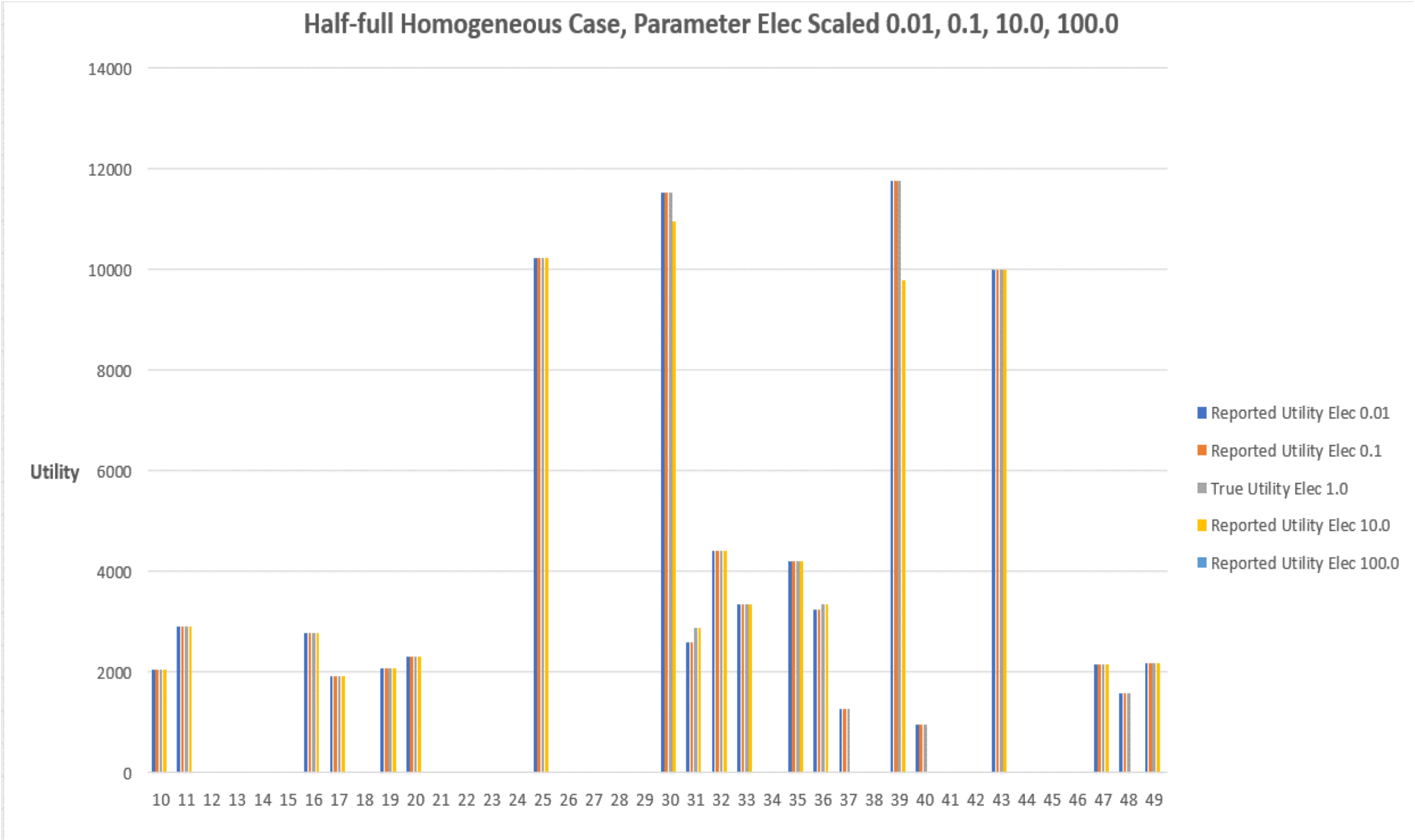
- Scaling down can result in negative utility
- Scaling up can result in at least zero utility

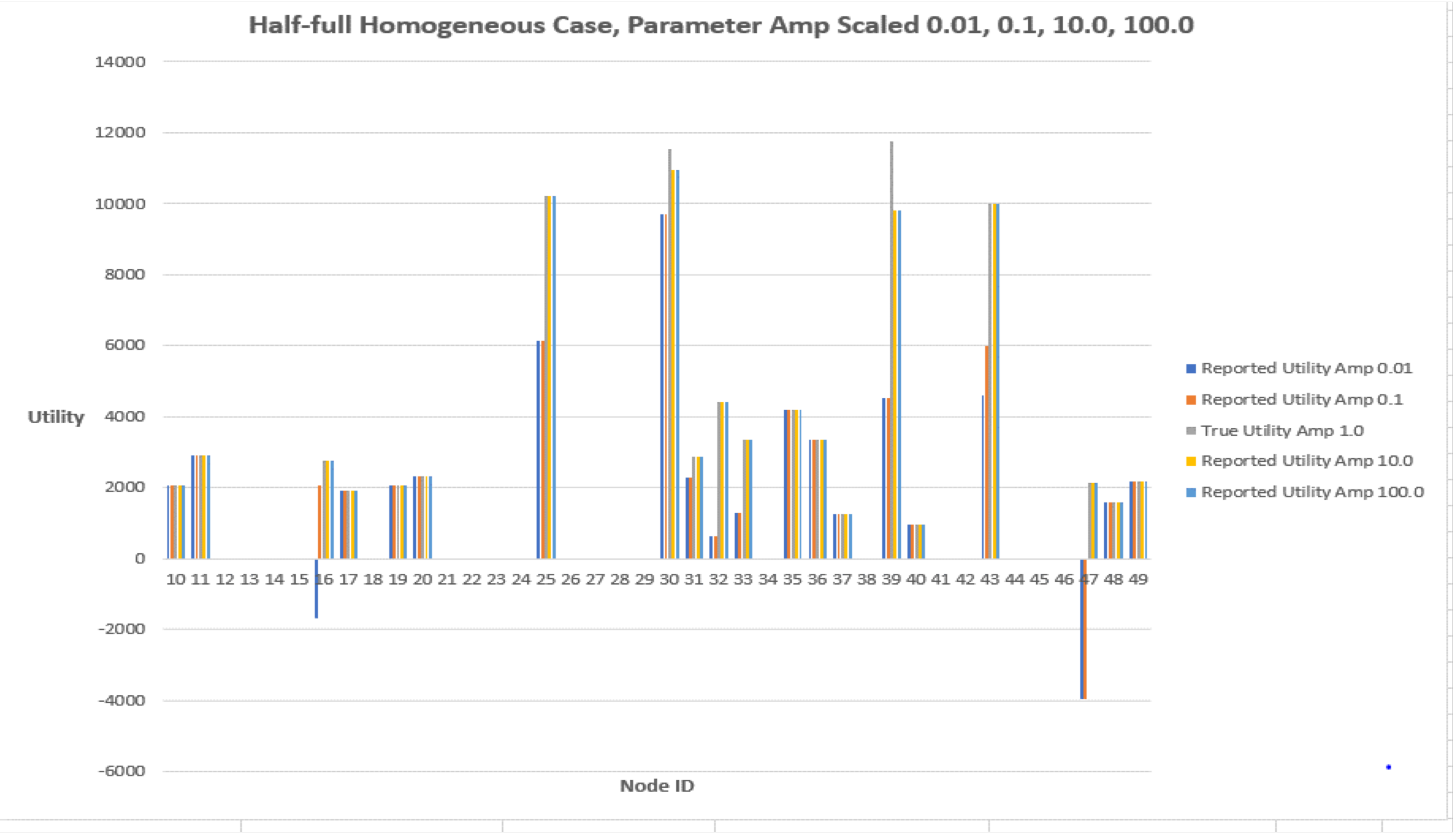
Half-full Homogeneous Case

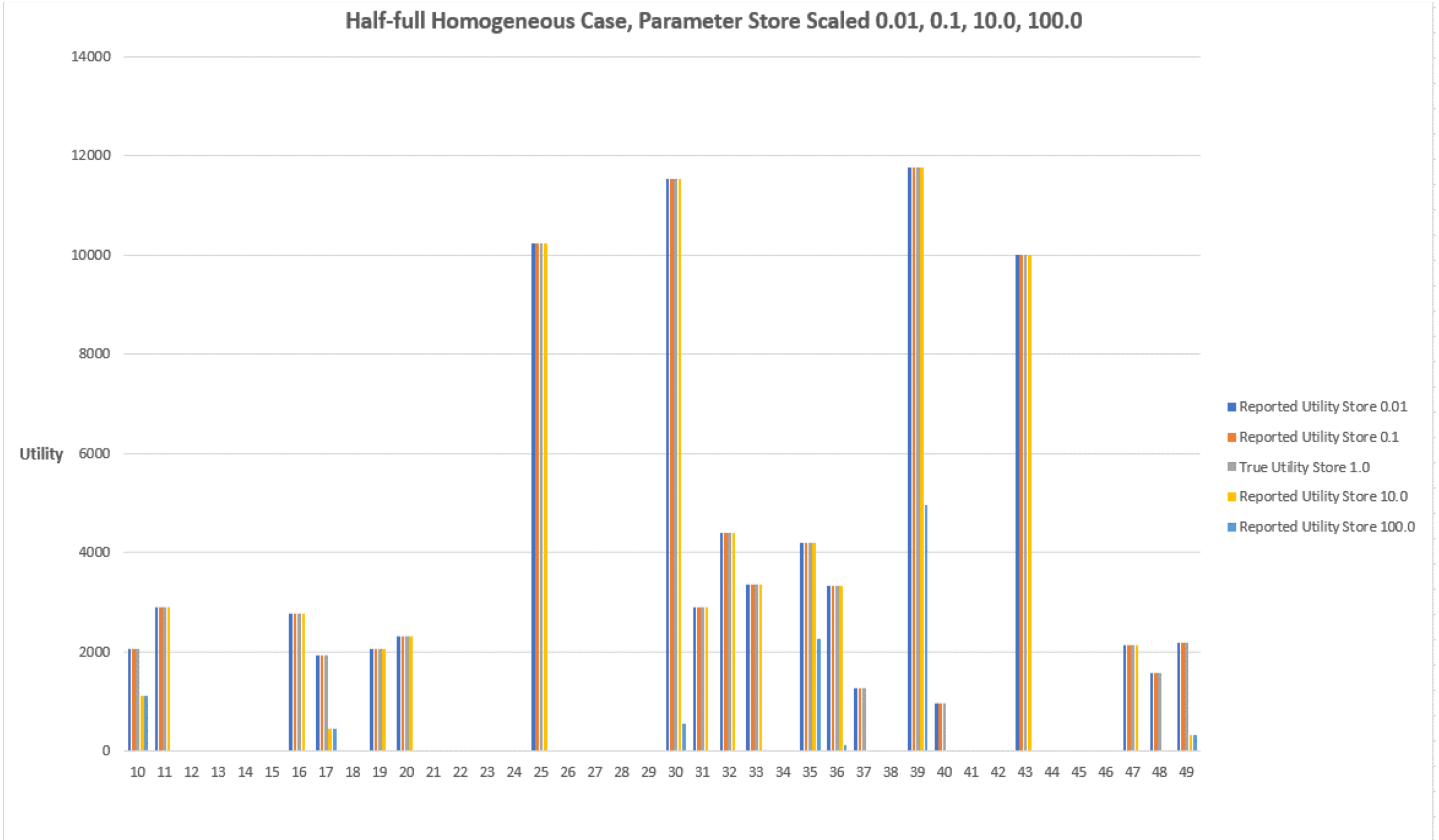
- Data generators
  - Number of data generators = 10
  - Number of items generated = 100 data items
- Storage Nodes
  - Number of storage nodes = 40
  - Storage per node = 50 data items

- All nodes have the same true cost parameter values
  - Default ε_{elec} = 100 nanojoules = 100 * 10^{-9} Joules
  - Default ε_{amp} = 100 picojoules = 100 * 10^{-12} Joules
  - Default ε_{store} = 100 nanojoules = 100 * 10^{-9} Joules

- Total data generated is half of total network capacity


Half-full Homogeneous Case, Parameter Elec Scaled 0.01, 0.1, 10.0, 100.0
Utility
14000
12000
10000
8000
6000
4000
2000
0
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
Reported Utility Elec 0.01
Reported Utility Elec 0.1
True Utility Elec 1.0
Reported Utility Elec 10.0
Reported Utility Elec 100.0

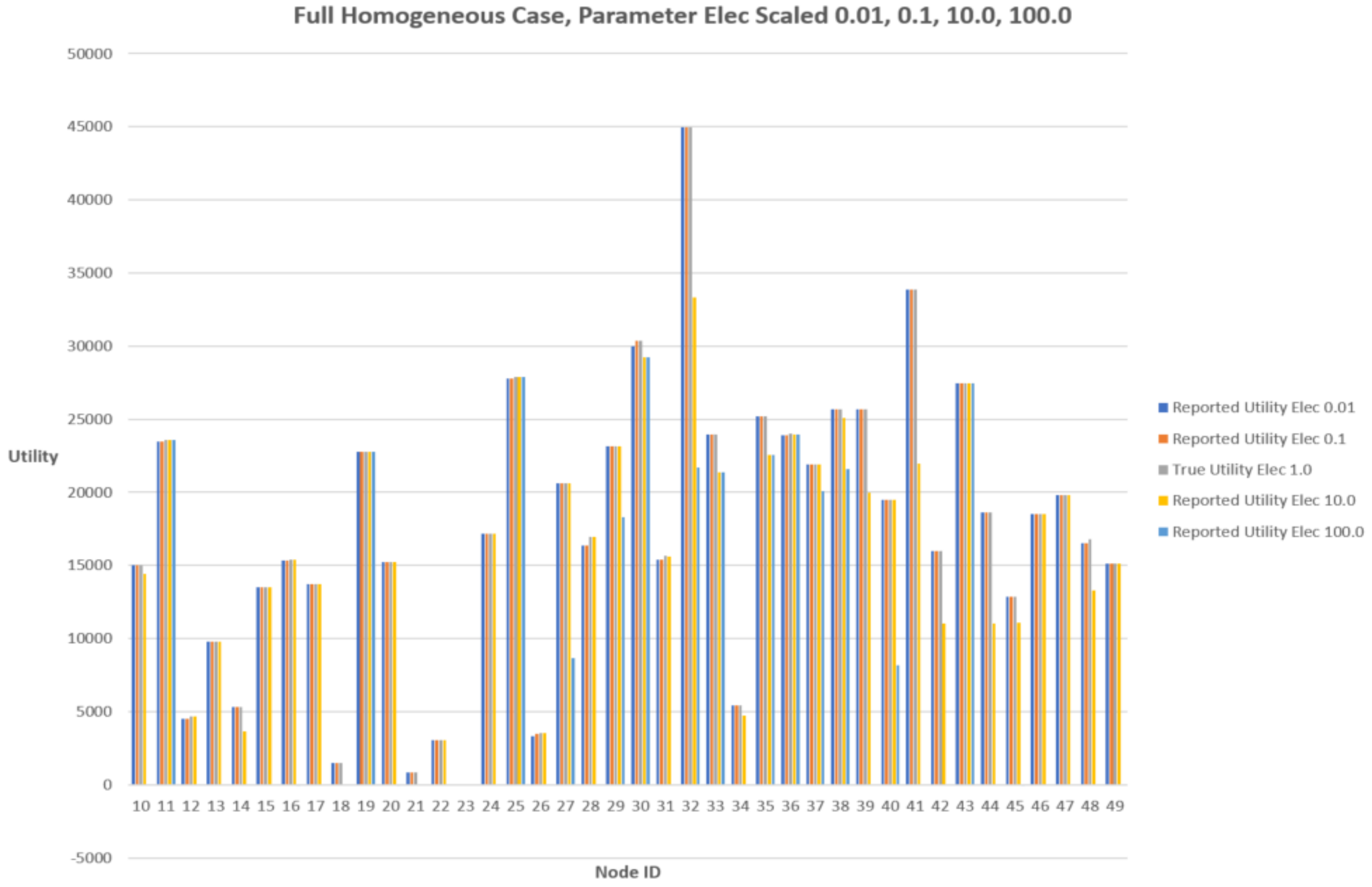

Half-full Homogeneous Case, Parameter Amp Scaled 0.01, 0.1, 10.0, 100.0
Utility
14000
12000
10000
8000
6000
4000
2000
0
-2000
-4000
-6000
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
Node ID
Reported Utility Amp 0.01
Reported Utility Amp 0.1
True Utility Amp 1.0
Reported Utility Amp 10.0
Reported Utility Amp 100.0

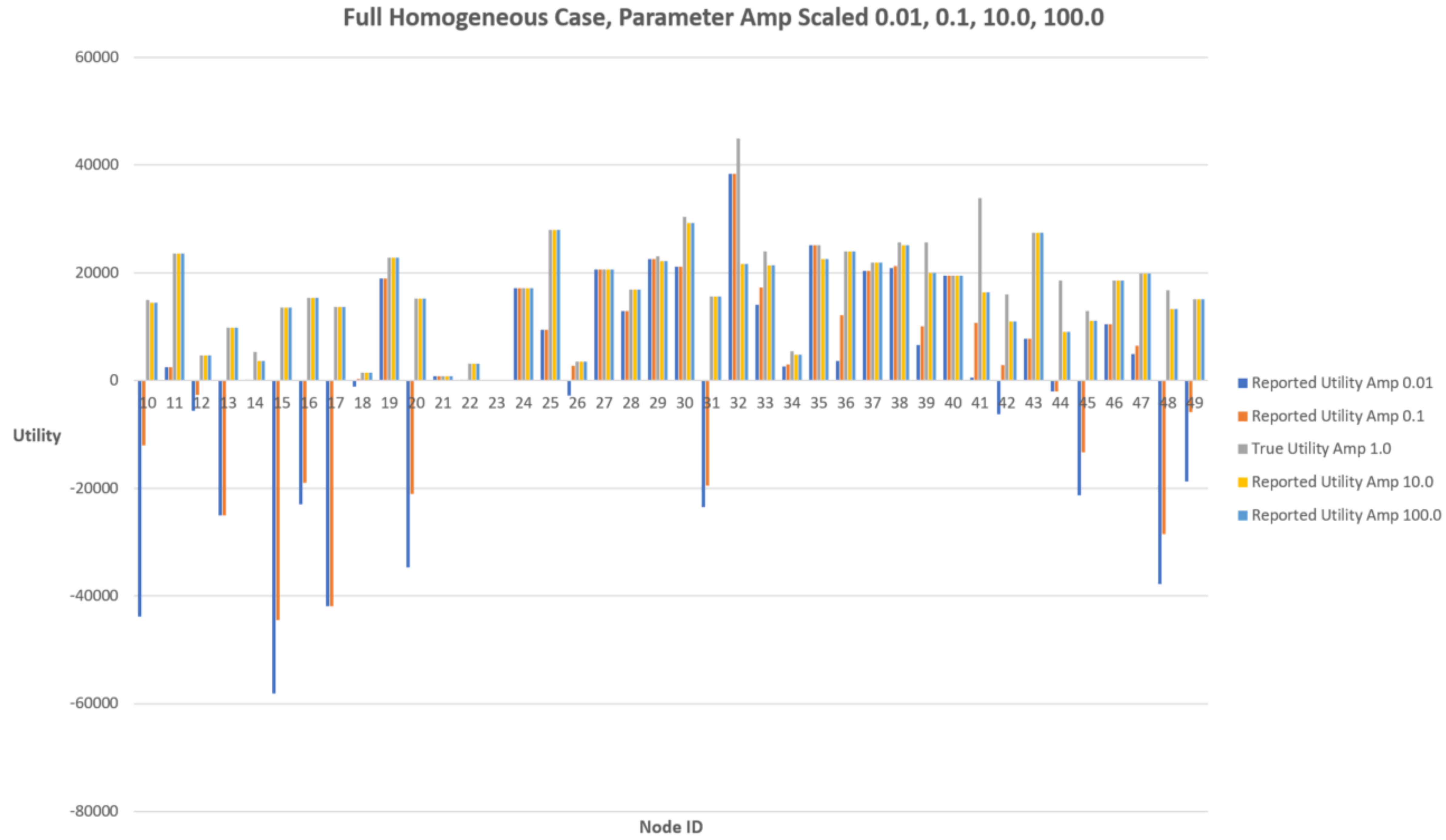

Half-full Homogeneous Case, Parameter Store Scaled 0.01, 0.1, 10.0, 100.0
Utility
14000
12000
10000
8000
6000
4000
2000
0
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
Reported Utility Store 0.01
Reported Utility Store 0.1
True Utility Store 1.0
Reported Utility Store 10.0
Reported Utility Store 100.0

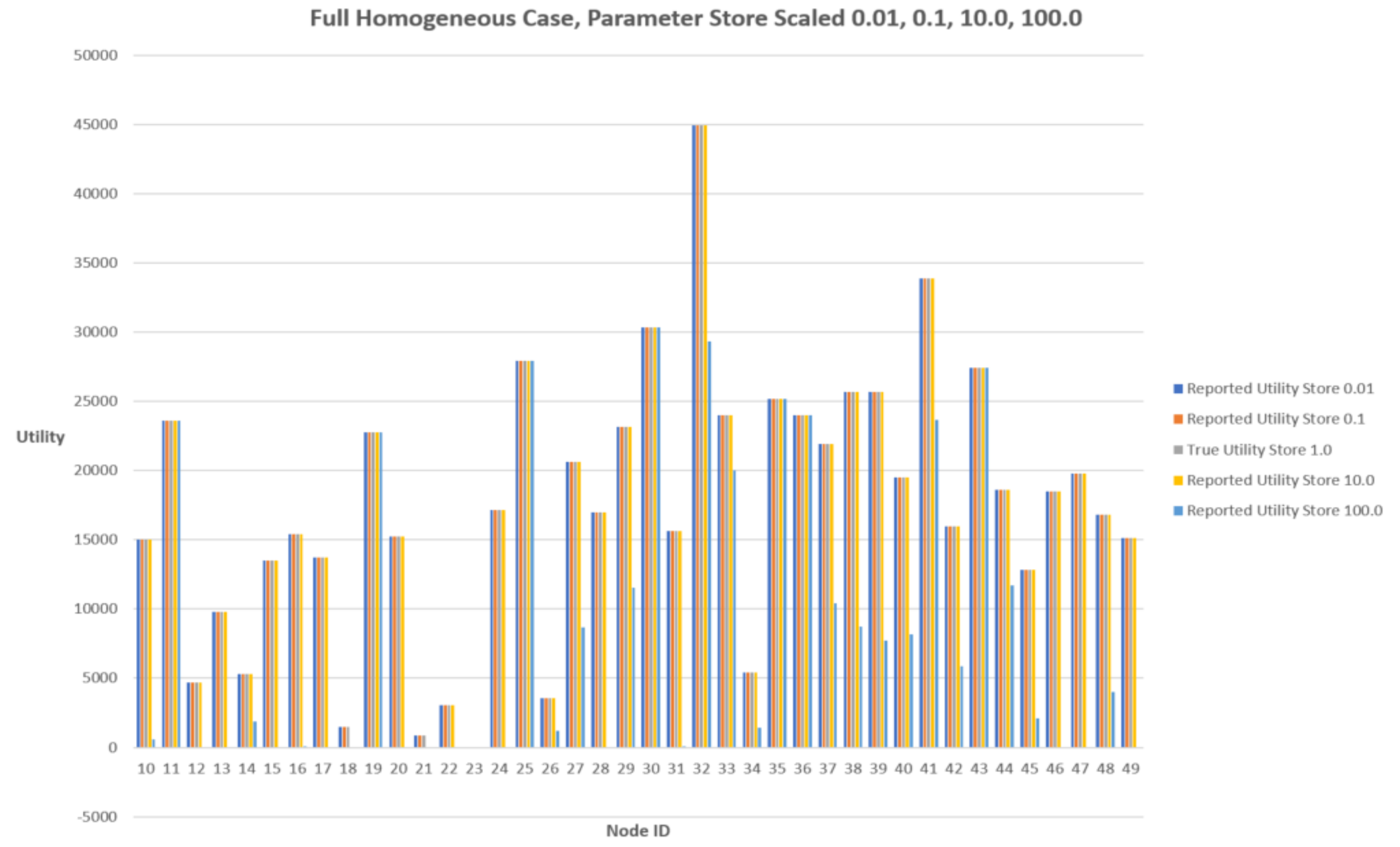
Full Homogeneous Case

- Same configuration as half-full homogeneous case except:
  - Data generators now generate **195** items instead of **100** items
  - Network will be nearly filled to capacity (1950 out of 2000 capacity)

- Excess capacity is required so network is still feasible if one node is removed


Full Homogeneous Case, Parameter Elec Scaled 0.01, 0.1, 10.0, 100.0
Utility
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
-5000
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
Node ID
Reported Utility Elec 0.01
Reported Utility Elec 0.1
True Utility Elec 1.0
Reported Utility Elec 10.0
Reported Utility Elec 100.0


Full Homogeneous Case, Parameter Amp Scaled 0.01, 0.1, 10.0, 100.0
Utility
60000
40000
20000
0
-20000
-40000
-60000
-80000
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
Node ID
Reported Utility Amp 0.01
Reported Utility Amp 0.1
True Utility Amp 1.0
Reported Utility Amp 10.0
Reported Utility Amp 100.0

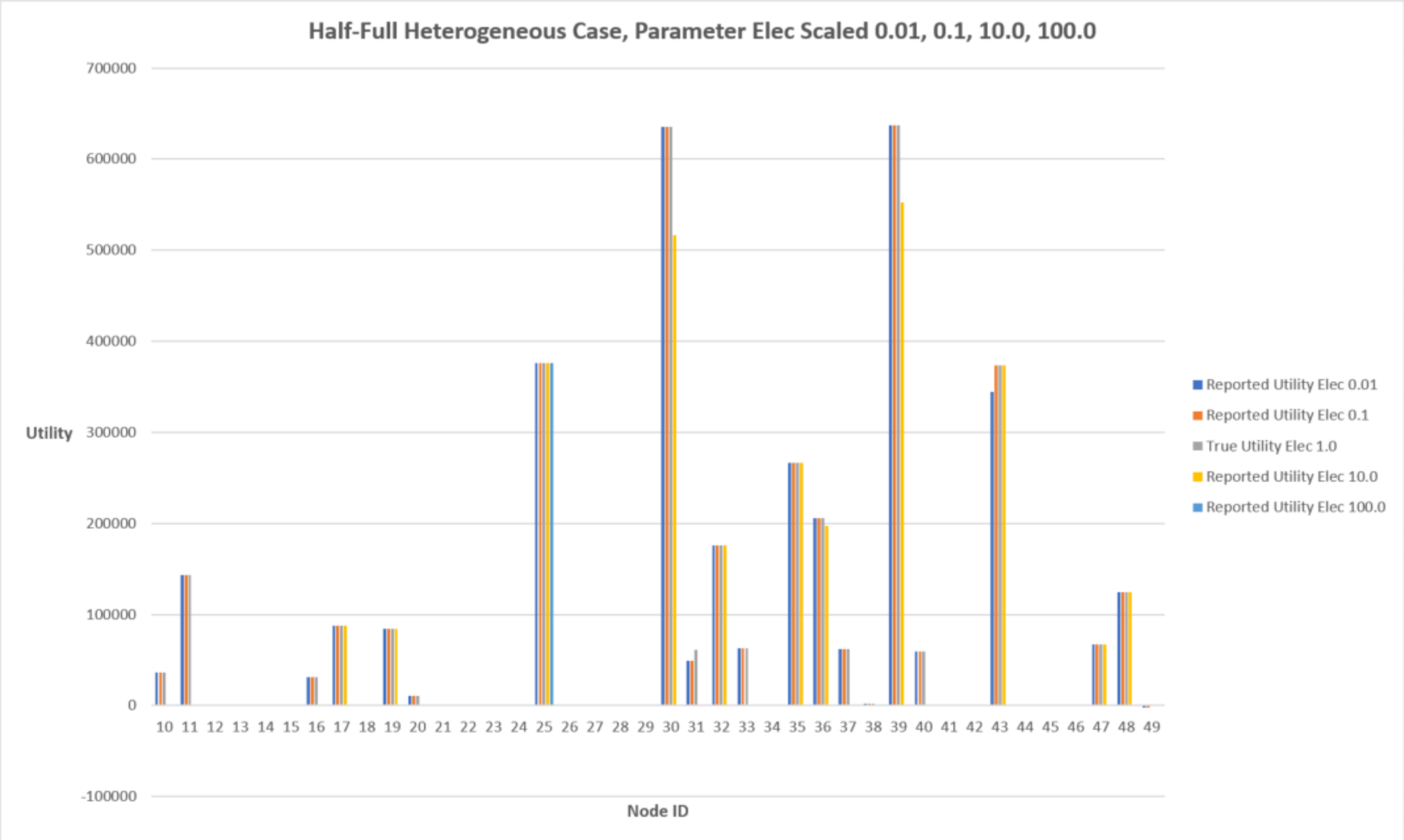

Full Homogeneous Case, Parameter Store Scaled 0.01, 0.1, 10.0, 100.0
Utility
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
-5000
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
Node ID
Reported Utility Store 0.01
Reported Utility Store 0.1
True Utility Store 1.0
Reported Utility Store 10.0
Reported Utility Store 100.0

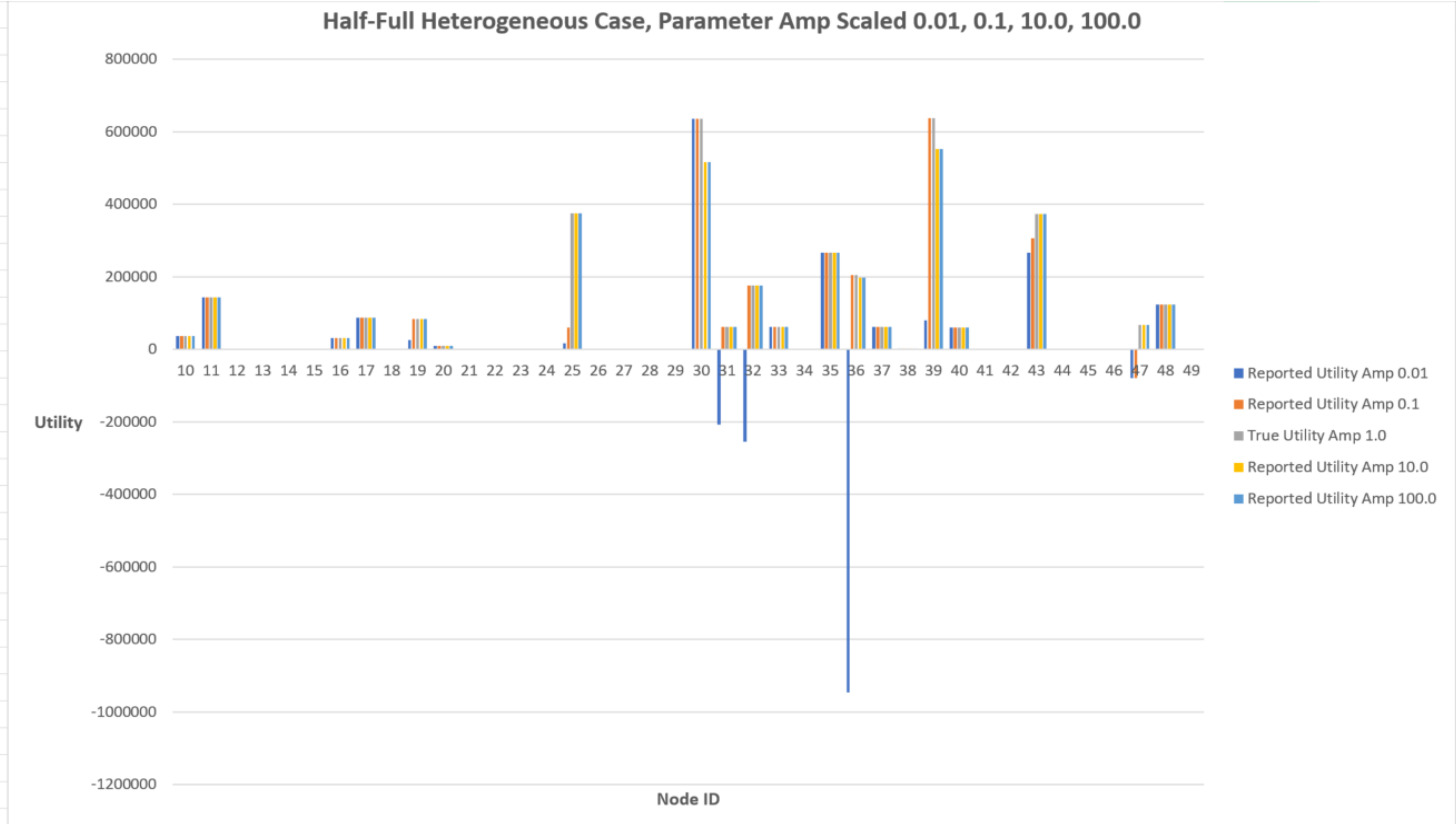
Half-full Heterogeneous Case

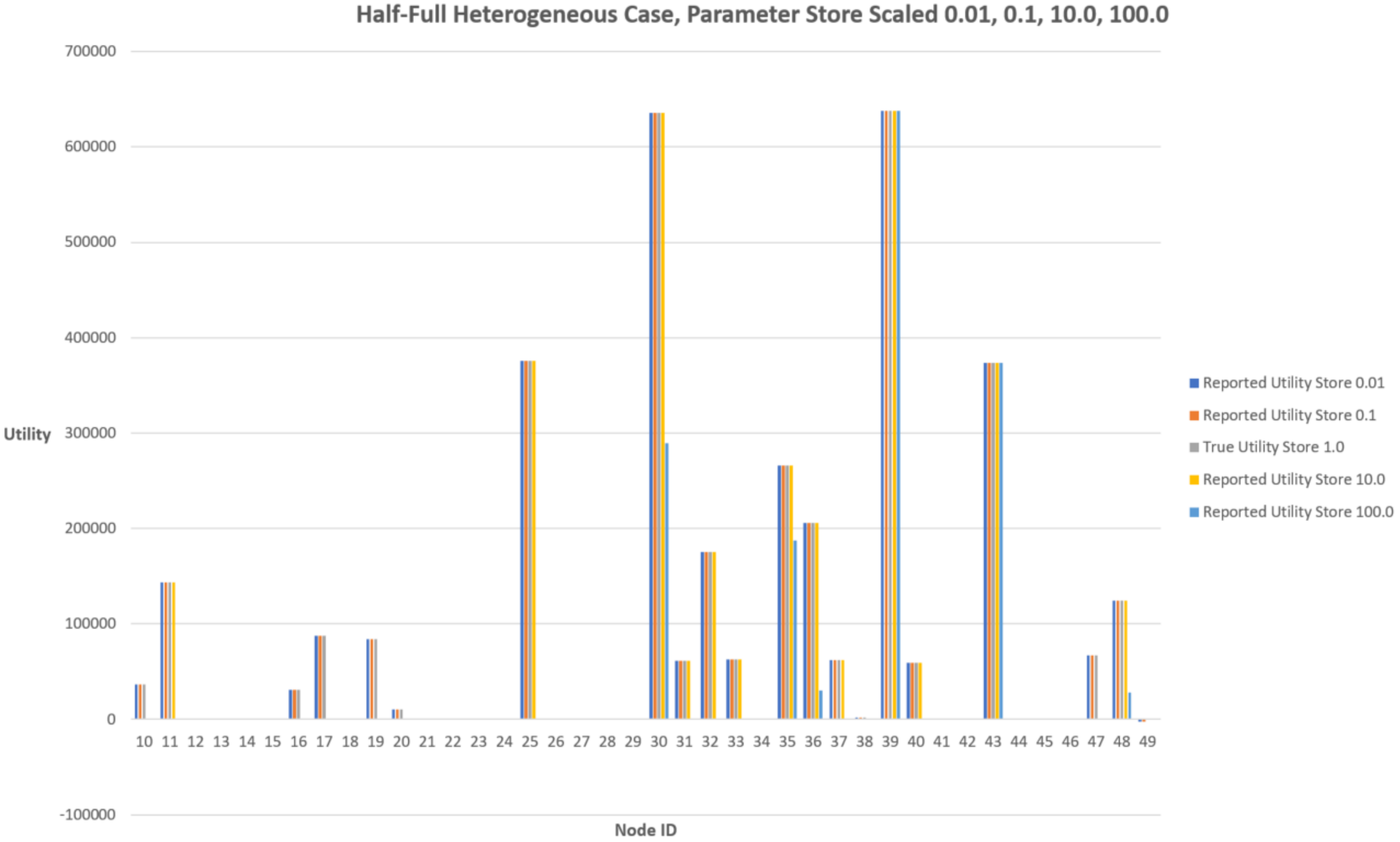
- Data generators
  - Number of data generators = 10
  - Number of items generated = 100 data items
- Storage Nodes
  - Number of storage nodes = 40
  - Storage per node = 50 data items

- All nodes have randomized true parameter values
  - Default ε_{elec} = random number from interval [100, 10000] nanojoules
  - Default ε_{amp} = random number from interval [100, 10000] picojoules
  - Default ε_{store} = random number from interval [100, 10000] nanojoules

- Total data generated is half of total network capacity


Half-Full Heterogeneous Case, Parameter Elec Scaled 0.01, 0.1, 10.0, 100.0
Utility
700000
600000
500000
400000
300000
200000
100000
0
-100000
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
Node ID
Reported Utility Elec 0.01
Reported Utility Elec 0.1
True Utility Elec 1.0
Reported Utility Elec 10.0
Reported Utility Elec 100.0

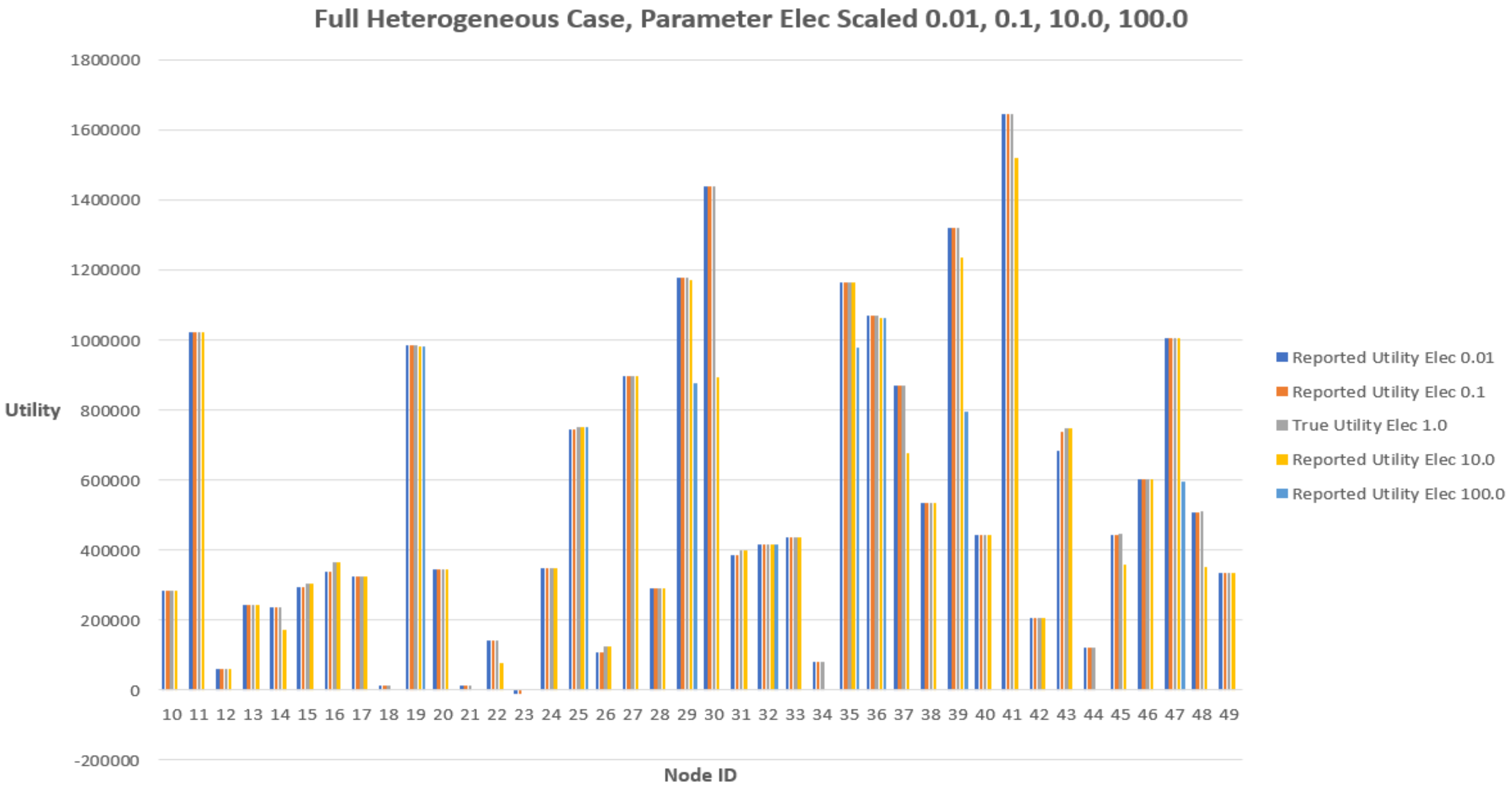

Half-Full Heterogeneous Case, Parameter Amp Scaled 0.01, 0.1, 10.0, 100.0
Utility
800000
600000
400000
200000
0
-200000
-400000
-600000
-800000
-1000000
-1200000
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
Node ID
Reported Utility Amp 0.01
Reported Utility Amp 0.1
True Utility Amp 1.0
Reported Utility Amp 10.0
Reported Utility Amp 100.0

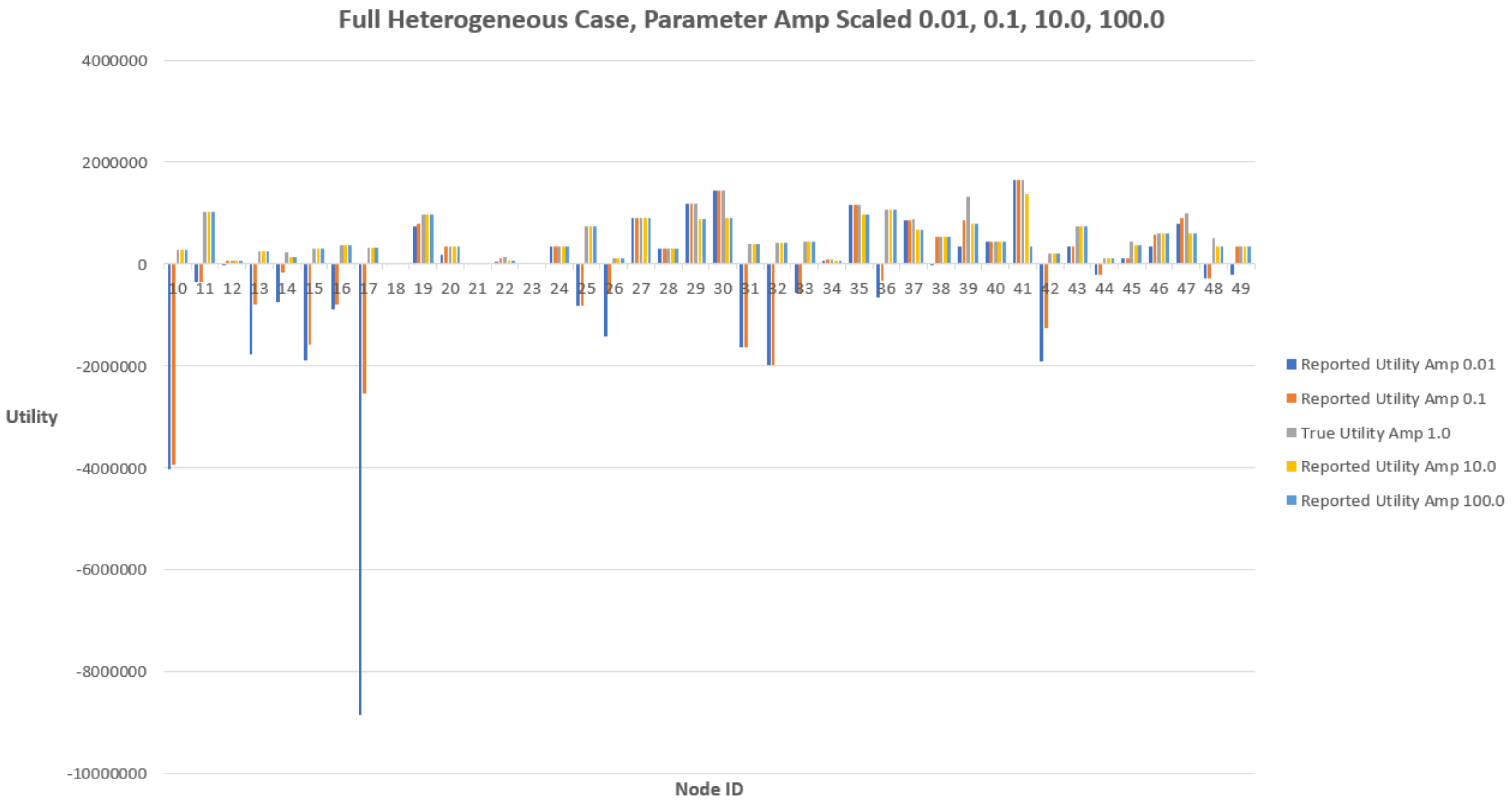

Half-Full Heterogeneous Case, Parameter Store Scaled 0.01, 0.1, 10.0, 100.0
Utility
700000
600000
500000
400000
300000
200000
100000
0
-100000
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
Node ID
Reported Utility Store 0.01
Reported Utility Store 0.1
True Utility Store 1.0
Reported Utility Store 10.0
Reported Utility Store 100.0

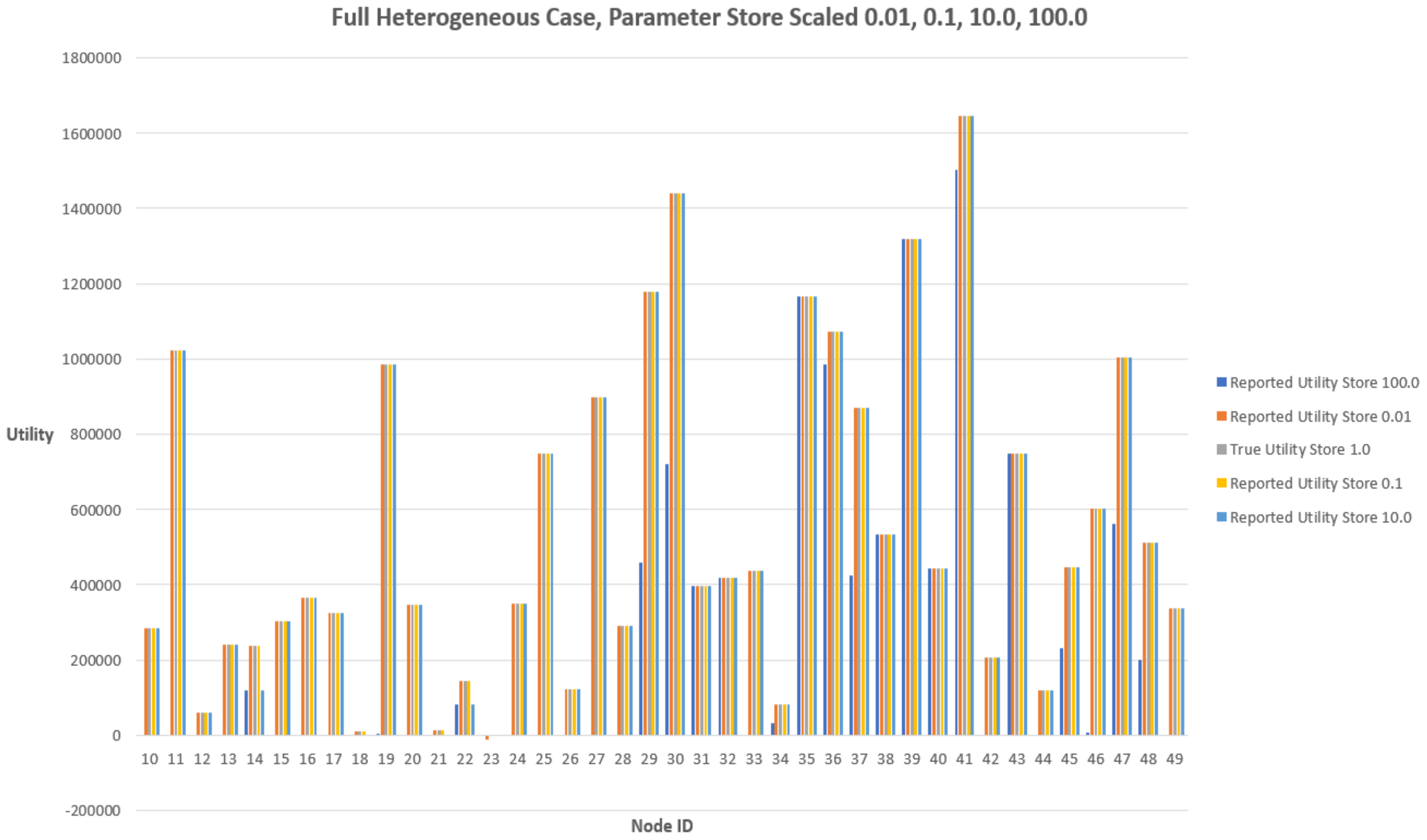
Full Heterogeneous Case

- Same configuration as half-full heterogeneous case except
  - Data generators now generate **195** items instead of **100** items
  - Network will be nearly filled to capacity (1950 out of 2000 capacity)

- Excess capacity is required so network is still feasible if one node is removed


Full Heterogeneous Case, Parameter Elec Scaled 0.01, 0.1, 10.0, 100.0
Utility
1800000
1600000
1400000
1200000
1000000
800000
600000
400000
200000
0
-200000
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
Node ID
Reported Utility Elec 0.01
Reported Utility Elec 0.1
True Utility Elec 1.0
Reported Utility Elec 10.0
Reported Utility Elec 100.0


Full Heterogeneous Case, Parameter Amp Scaled 0.01, 0.1, 10.0, 100.0
Utility
4000000
2000000
0
-2000000
-4000000
-6000000
-8000000
-10000000
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
Node ID
Reported Utility Amp 0.01
Reported Utility Amp 0.1
True Utility Amp 1.0
Reported Utility Amp 10.0
Reported Utility Amp 100.0


Full Heterogeneous Case, Parameter Store Scaled 0.01, 0.1, 10.0, 100.0
Utility
1800000
1600000
1400000
1200000
1000000
800000
600000
400000
200000
0
-200000
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
Node ID
Reported Utility Store 100.0
Reported Utility Store 0.01
True Utility Store 1.0
Reported Utility Store 0.1
Reported Utility Store 10.0

Conclusions

- In all scenarios and in all cases, utility is maximized under truth-telling. This is consistent with theory.

- By designing the correct payment function (incentive), nodes can be motivated to do what’s right, even if their motivation is primarily selfish.

- Simulation helps empirically verify results, understand real-world implications, and refine theory

Future Work

- Infeasible Case
  - Nodes can run out of power and disappear from the network
  - How do we deal with monopoly of control?
  - How do we compute expected utility? What is the underlying probability distribution?

- Collusion Scenario
  - Can we incentivize nodes to cooperate if some of them work together to cheat the system?
  - Can we restrict the effect of external incentives?

Acknowledgements

- Parents
- Advisor and Committee Chair – Dr. Bin Tang
- Committee Members
  - Dr. Mohsen Beheshti
  - Dr. Jianchao Han
- Mentors from The Aerospace Corporation
  - Rick Johnson
  - Dr. Supannika Mobasser
- National Science Foundation for funding support under award number 1419952

References

- Anderegg, L. and Eidenbenz, S. 2003. Ad hoc-VCG: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents. In Proceedings of the 9th annual international conference on Mobile computing and networking (MobiCom '03). ACM, New York, NY, USA, 245-259.
- Candell, R., Liu, Y., Lee, K. Moayeri, N. "A simulation framework for industrial wireless networks and process control systems", Proc. IEEE WFCS'16, pp. 111, May 2016.
- Chen, Y. and Tang, B. “Data Preservation in Base Station-less Sensor Networks: A Game Theoretic Approach”, Proceedings of the 6th EAI International Conference on Game Theory for Networks, Kelowna, BC, Canada, 2016.
- Dobzinski, S. and Nisan, N. 2007. Limitations of VCG-based mechanisms. In Proceedings of the thirty-ninth annual ACM symposium on Theory of computing (STOC '07). ACM, New York, NY, USA, 338-344.
- Feigenbaum, J., Papadimitriou C., Sami, R. and Shenker, S. 2005. A BGP-based mechanism for lowest-cost routing. Distributed Computing 18, 1 (July 2005), 61-72.
- Feigenbaum, J. and Shenker, S. 2002. Distributed algorithmic mechanism design: recent results and future directions. In Proceedings of the 6th international workshop on Discrete algorithms and methods for mobile computing and communications (DIALM '02). ACM, New York, NY, USA, 1-13.

References *(continued)*

- Felegyhazi, M. and Hubaux, J.-P. “Game theory in wireless networks: A tutorial,” École Polytechnique Fédérale de Lausanne. Technical Report LCA-REPORT-2006-002.
- Goldberg, A. V., 1997. An efficient implementation of a scaling minimum-cost flow algorithm. J. Algorithms 22, 1 (January 1997), 1-29
- Iyengar, S.S and Kannan, R. Game-theoretic models for reliable pathlength and energy-constrained routing with data aggregation in wireless sensor networks, IEEE Journal of Selected Areas in Communications (2004) 1141–1150.
- Jaggi, N., Kurkal, R., Tang B., and Wu H. "Energy-Efficient Data Redistribution in Sensor Networks," ACM Transactions on Sensor Networks, v.9, 2013.
- Machado, R. and Tekinay, S. 2008. A survey of game-theoretic approaches in wireless sensor networks. Computer Networks Volume 52, Issue 16 (November 2008), 3047-3061.
- Nisan, N. and Ronen, A. 1999. Algorithmic mechanism design. In Proceedings of the thirty-first annual ACM symposium on Theory of computing (STOC '99). ACM, New York, NY, USA, 129-140.