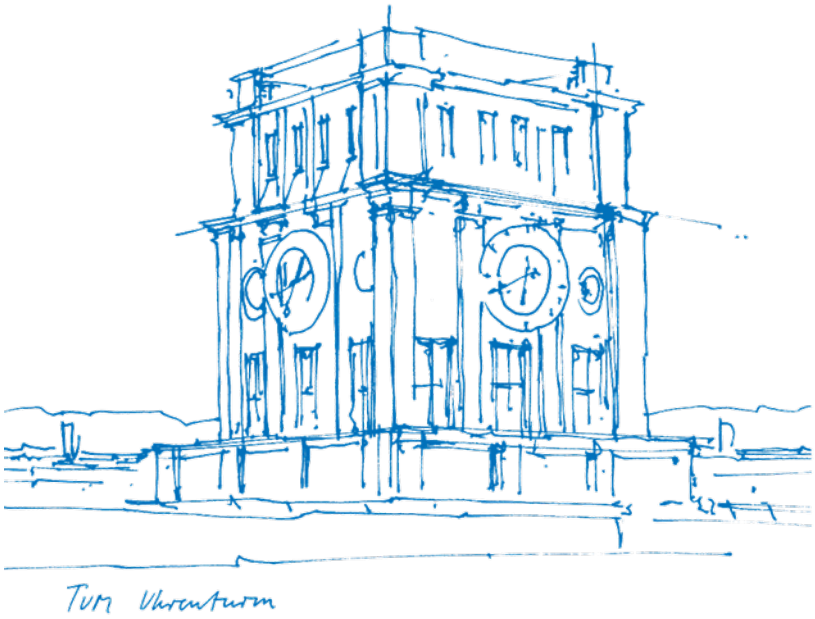
Master's Colloquium

B.Sc. Jonas Schulz
Technical University of Munich
Department of Informatics
München, 19.12.2022



Master's Colloquium

An Experimental Evaluation of Heuristics for Approximate Maxflow Algorithms

Maxflows: Runtimes

- Directed Graph: $\mathcal{O}(m \cdot \min(m^{1/2}, n^{1/3}))$ (Goldberg, Rao)

Maxflows: Runtimes

- Directed Graph: $\mathcal{O}(m \cdot \min(m^{1/2}, n^{1/3}))$ (Goldberg, Rao)
- Sparsification: $T(m,n) \rightarrow \mathcal{O}(m) + (m/n) \cdot T(\mathcal{O}(n), n))$ (Benczùr, Karger)

Maxflows: Runtimes

- Directed Graph: $\mathcal{O}(m \cdot \min(m^{1/2}, n^{1/3}))$ (Goldberg, Rao)
- Sparsification: $T(m,n) \rightarrow \mathcal{O}(m) + (m/n) \cdot T(\mathcal{O}(n), n))$ (Benczùr, Karger)
- Goldberg/Rao on sparsified graph (approximative): $\mathcal{O}(mn^{1/2})$

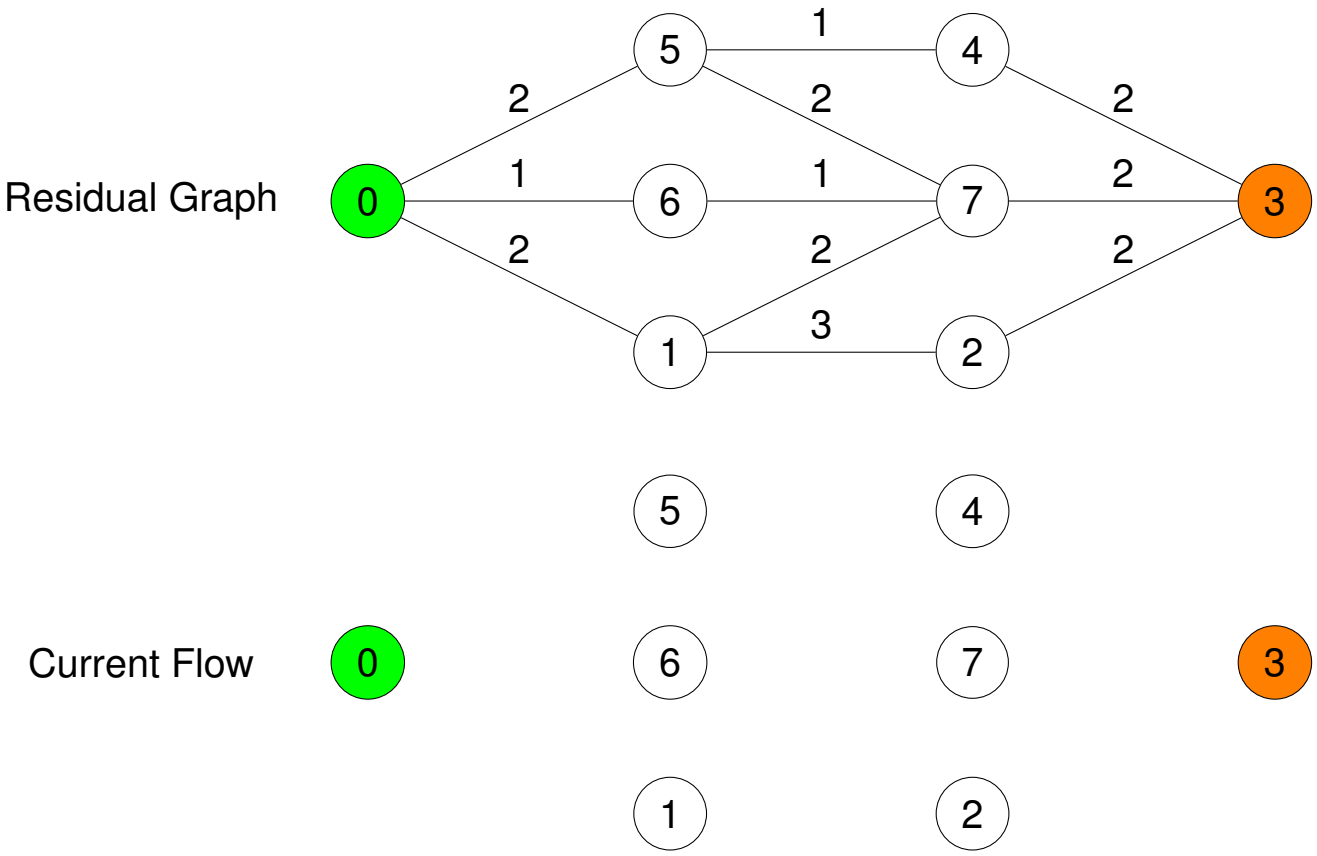
Maxflows: Runtimes

- Directed Graph: $\mathcal{O}(m \cdot \min(m^{1/2}, n^{1/3}))$ (Goldberg, Rao)
- Sparsification: $T(m,n) \rightarrow \mathcal{O}(m) + (m/n) \cdot T(\mathcal{O}(n), n))$ (Benczùr, Karger)
- Goldberg/Rao on sparsified graph (approximative): $\mathcal{O}(mn^{1/2})$
- Christiano *et al.* + Laplacian Solver (Spielman, Teng) (approximative): $\mathcal{O}(mn^{1/3})$

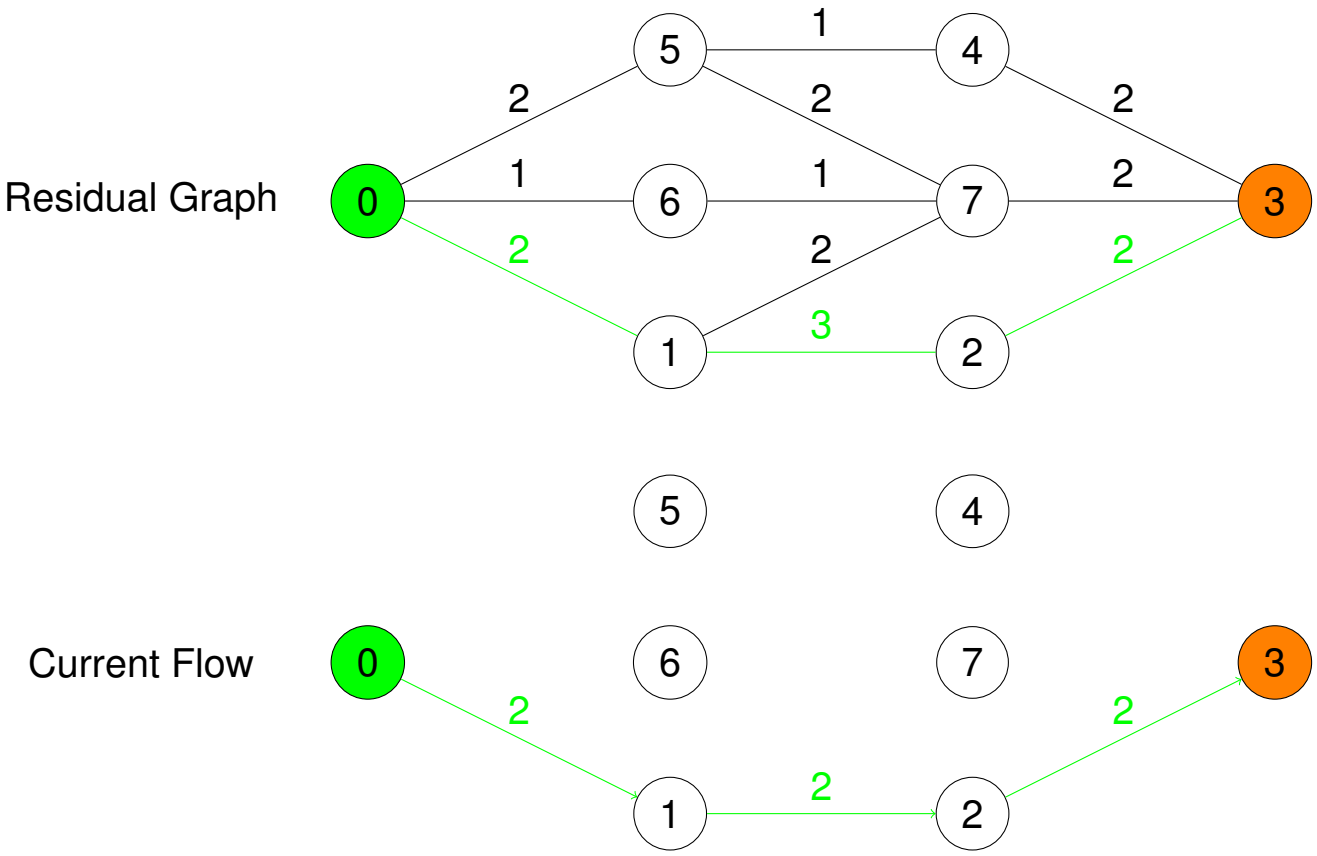
Maxflows: Runtimes

- Directed Graph: $\mathcal{O}(m \cdot \min(m^{1/2}, n^{1/3}))$ (Goldberg, Rao)
- Sparsification: $T(m,n) \rightarrow \mathcal{O}(m) + (m/n) \cdot T(\mathcal{O}(n), n))$ (Benczùr, Karger)
- Goldberg/Rao on sparsified graph (approximative): $\mathcal{O}(mn^{1/2})$
- Christiano *et al.* + Laplacian Solver (Spielman, Teng) (approximative): $\mathcal{O}(mn^{1/3})$
- Sherman: Single-commodity flows (undirected graph, approximative): $(1+\varepsilon)$-optimal in $\mathcal{O}(m^{1+o(1)}\varepsilon^{-2})$

Maxflows

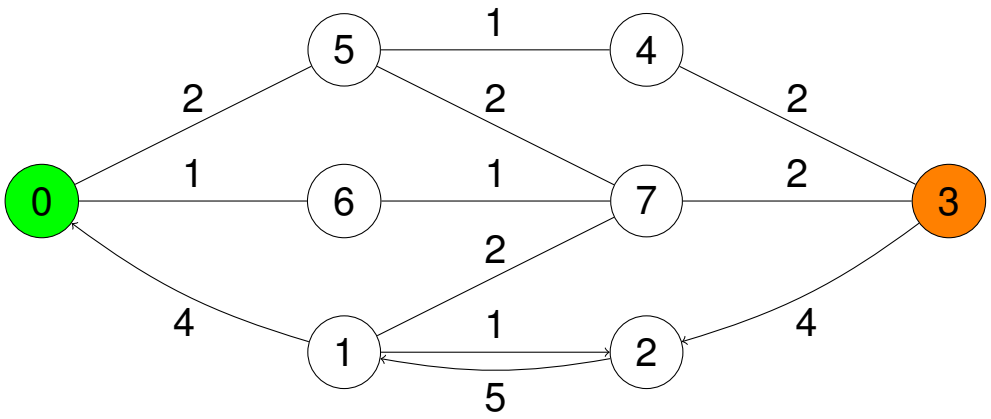

Residual Graph
0
5
6
1
4
7
2
3
1
2
2
1
2
1
2
2
2
3
2
2
Current Flow
0
5
6
1
4
7
2
3

Maxflows

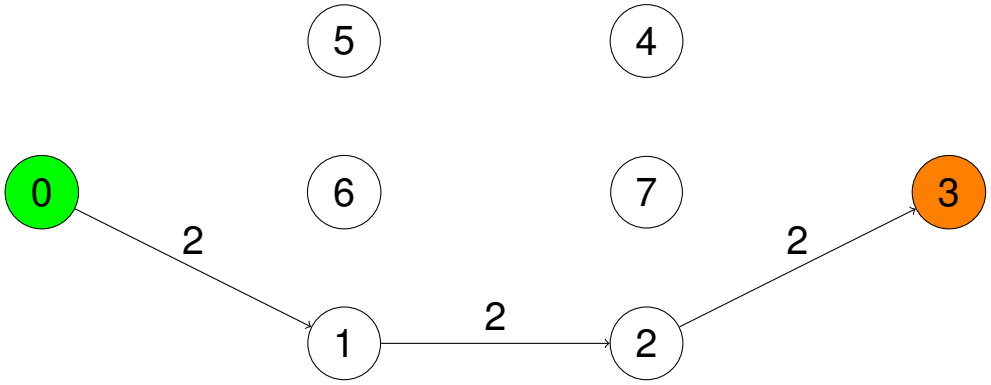

Residual Graph
5
4
0
6
7
3
1
2
1
2
2
1
2
2
2
2
3
Current Flow
5
4
0
6
7
3
1
2
2
2
2

Maxflows

Residual Graph

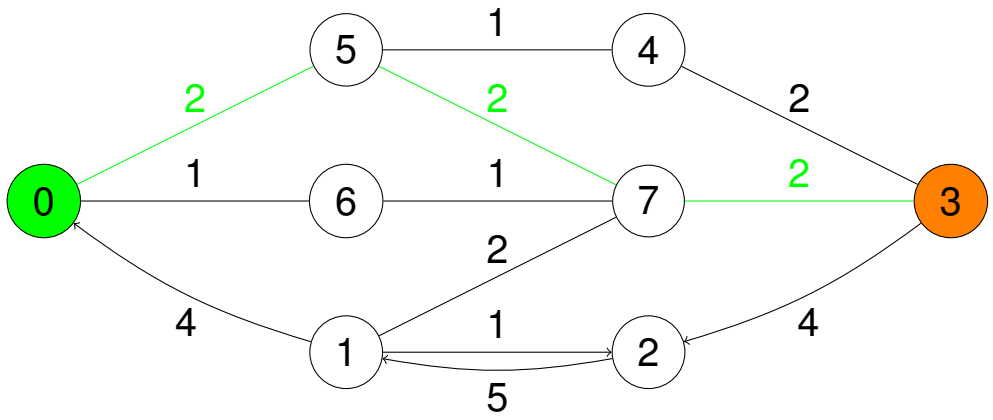


Current Flow

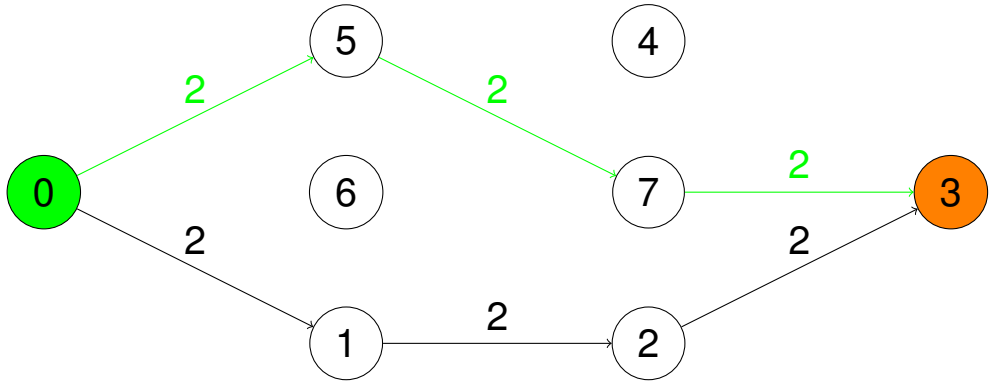


Maxflows

Residual Graph

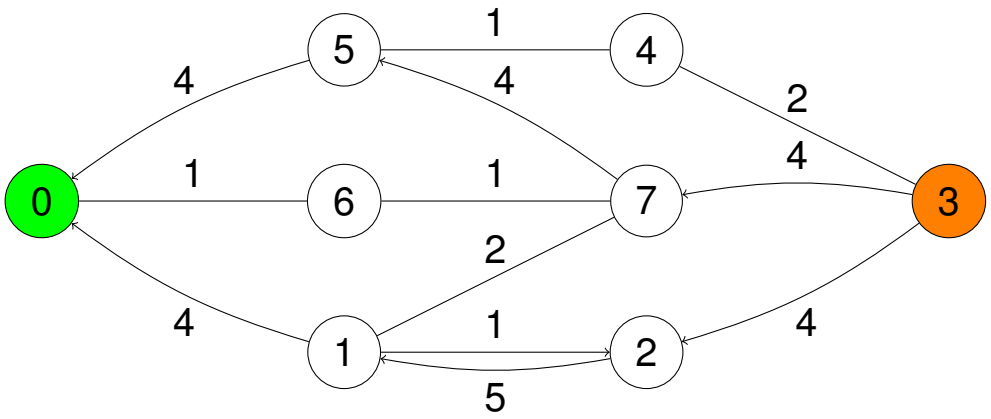


Current Flow

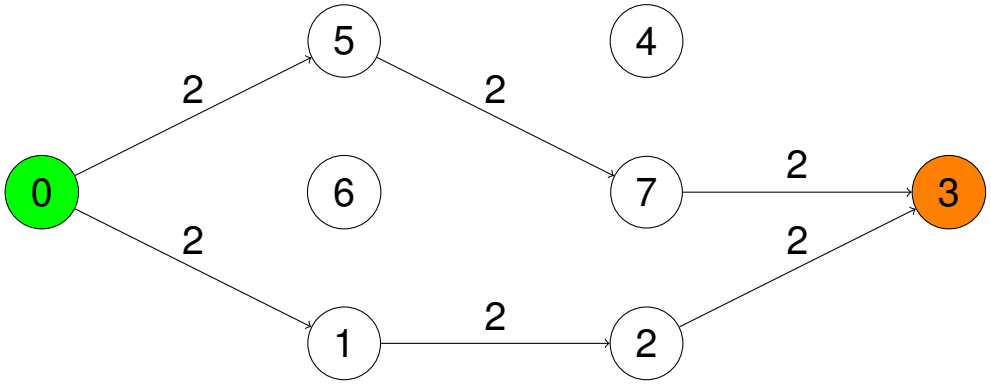


Maxflows

Residual Graph

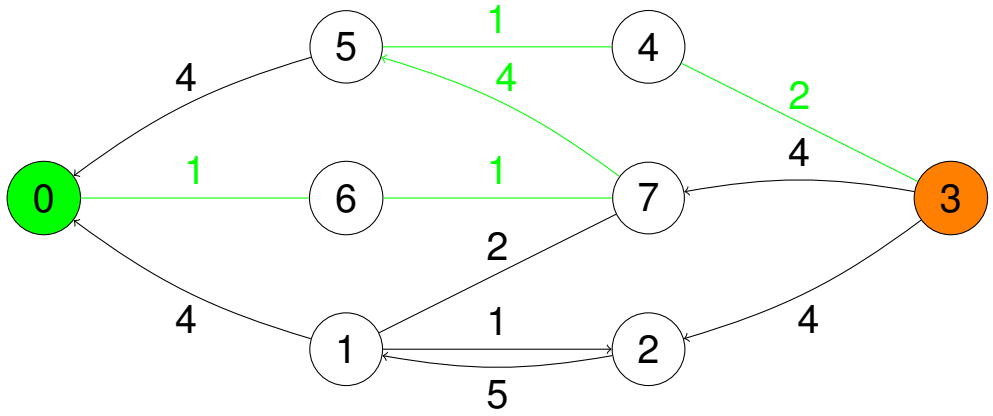


Current Flow

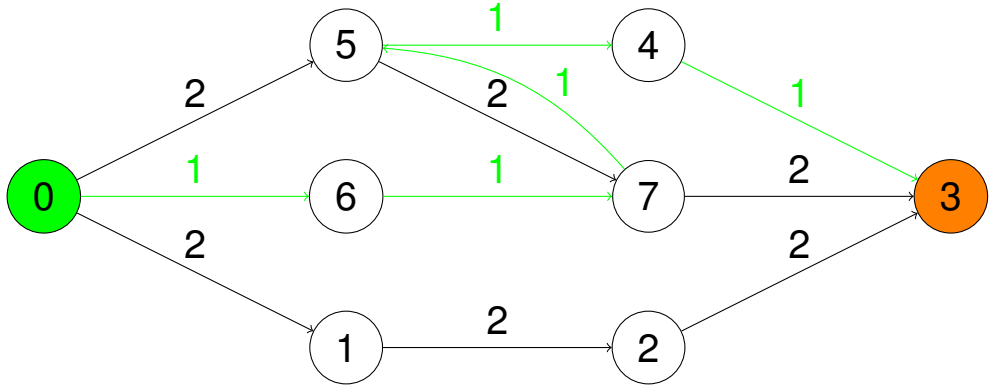


Maxflows

Residual Graph

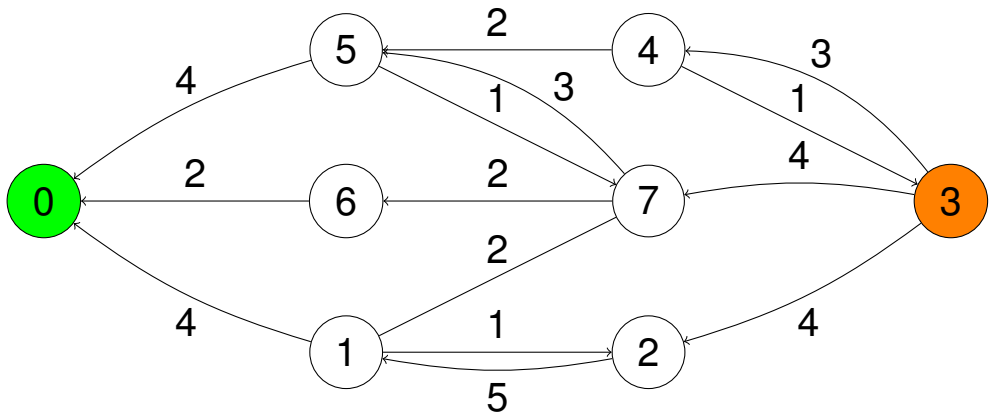


Current Flow

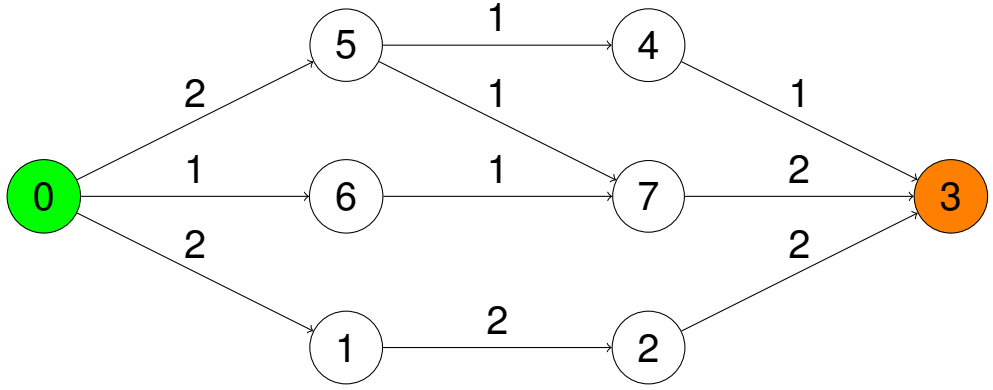


Maxflows

Residual Graph

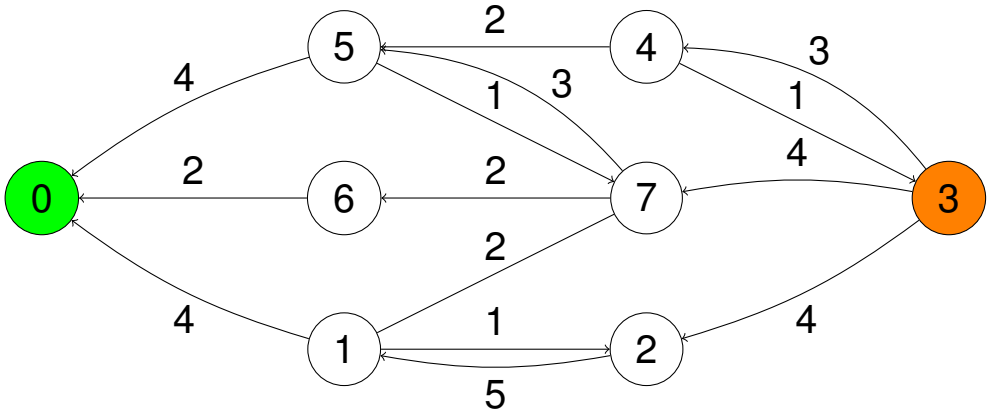


Current Flow

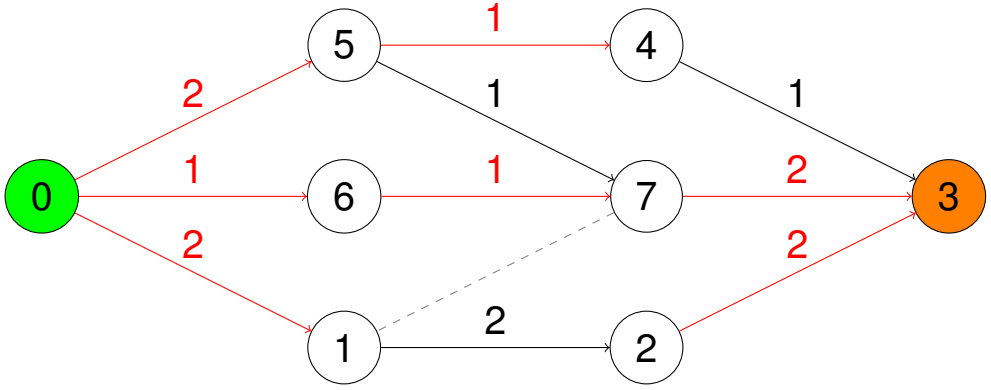


Maxflows

Residual Graph

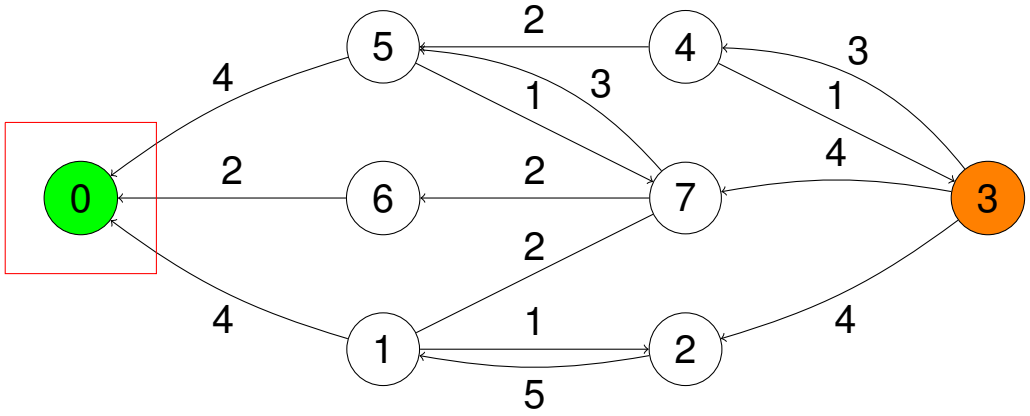


Current Flow

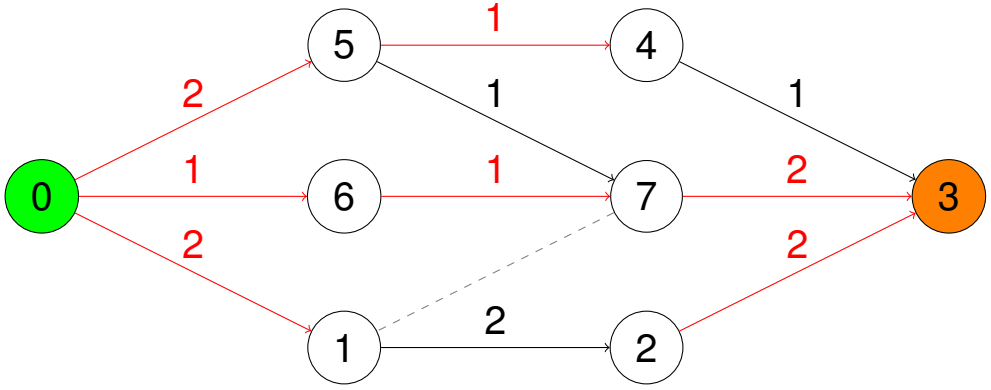


Maxflows

Residual Graph

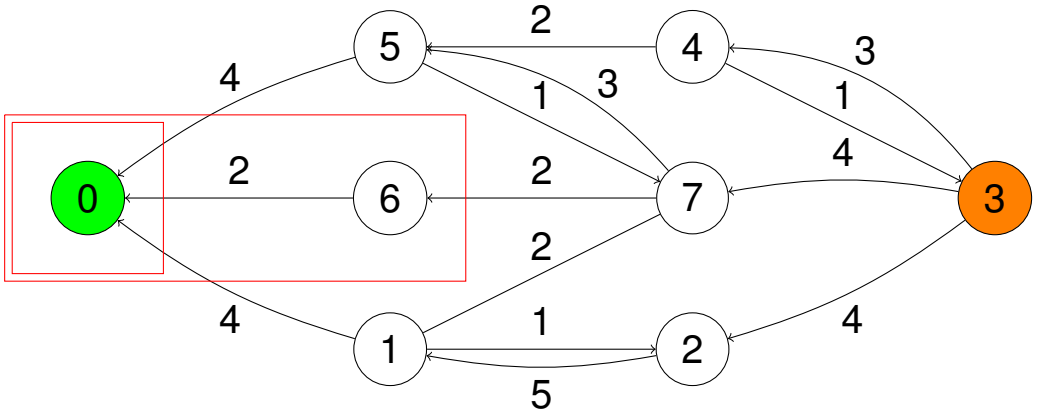


Current Flow

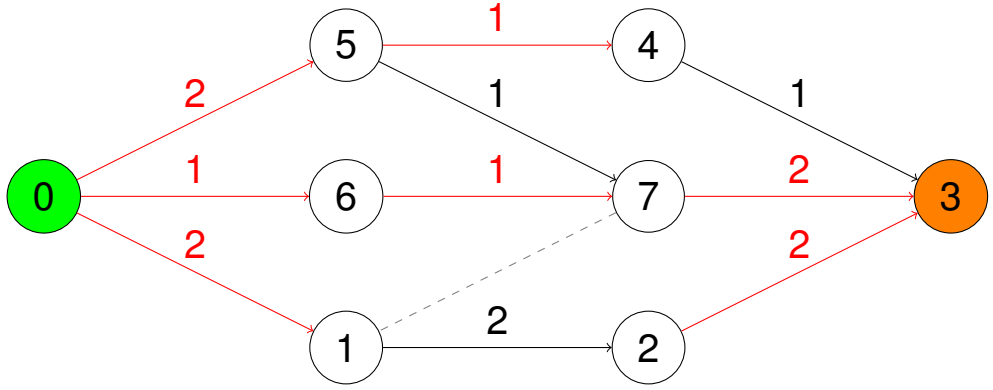


Maxflows

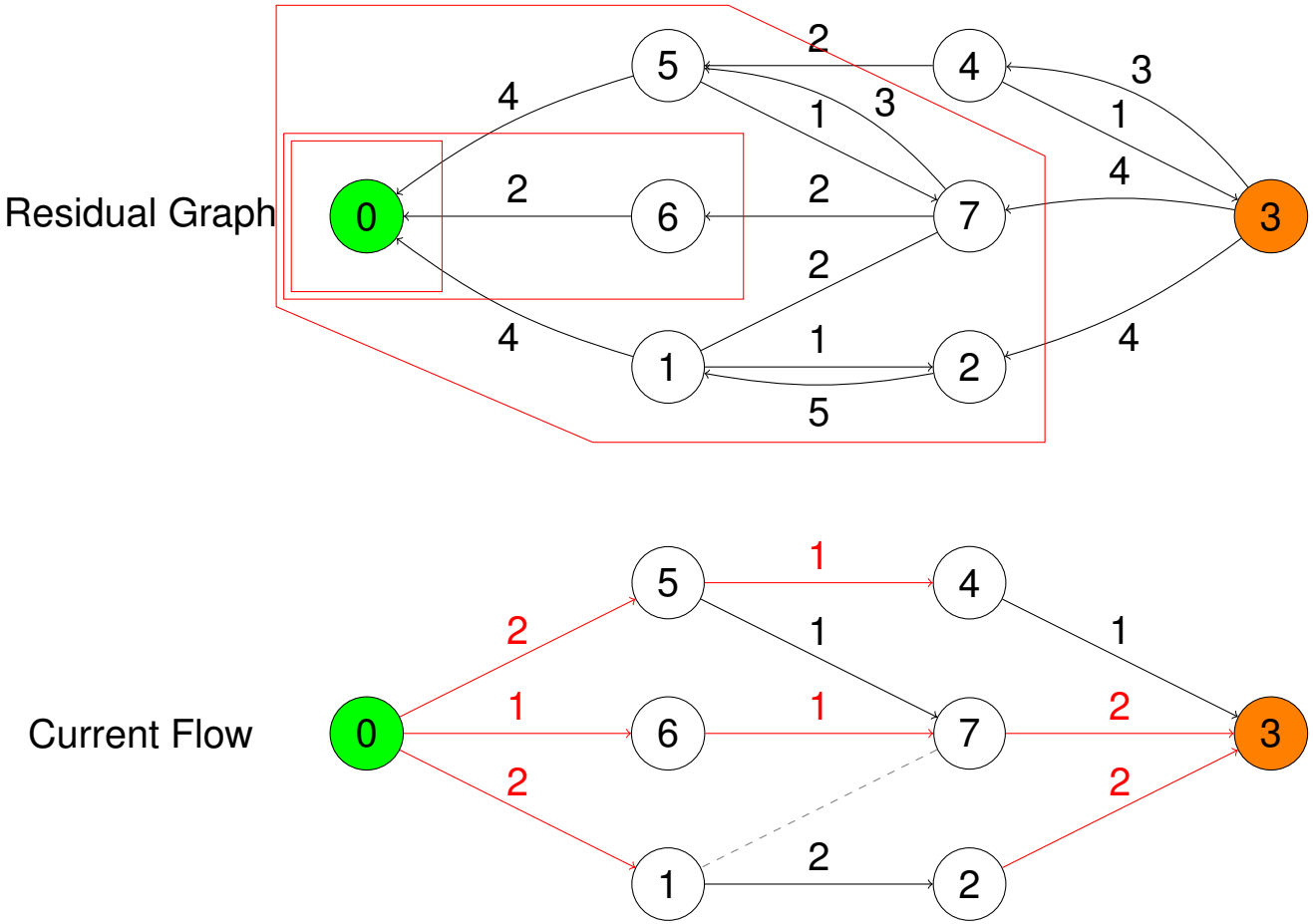
Residual Graph



Current Flow



Maxflows


Residual Graph
5
4
0
6
7
3
1
2
2
4
3
1
3
1
4
2
2
2
4
1
1
4
5
Current Flow
5
4
0
6
7
3
1
2
1
2
1
1
1
2
2
2
2

Minimum Congestion Flow

- Maximum Flow Problem:

Minimum Congestion Flow

- Maximum Flow Problem:
  - s-t-flow of maximal value

Minimum Congestion Flow

- Maximum Flow Problem:
  - s-t-flow of maximal value
  - restricted to s-t-flows

Minimum Congestion Flow

- Maximum Flow Problem:
  - s-t-flow of maximal value
  - restricted to s-t-flows
- Minimum Congestion Flow Problem:

Minimum Congestion Flow

- Maximum Flow Problem:
  - s-t-flow of maximal value
  - restricted to s-t-flows
- Minimum Congestion Flow Problem:
  - flow with minimal value of the *maximum congestion*

Minimum Congestion Flow

- Maximum Flow Problem:
  - s-t-flow of maximal value
  - restricted to s-t-flows
- Minimum Congestion Flow Problem:
  - flow with minimal value of the *maximum congestion*
  - *maximum congestion*: $\max_e \left| \frac{f_e}{c_e} \right|$

Minimum Congestion Flow

- Maximum Flow Problem:
  - s-t-flow of maximal value
  - restricted to s-t-flows
- Minimum Congestion Flow Problem:
  - flow with minimal value of the *maximum congestion*
  - *maximum congestion*: $\max_e \left| \frac{f_e}{c_e} \right|$
  - applicable on both s-t-flows *and* demand-based flows

Minimum Congestion Flow

- Maximum Flow Problem:
  - s-t-flow of maximal value
  - restricted to s-t-flows
- Minimum Congestion Flow Problem:
  - flow with minimal value of the *maximum congestion*
  - *maximum congestion*: $\max_e \left| \frac{f_e}{c_e} \right|$
  - applicable on both s-t-flows *and* demand-based flows
  - maximum flow $= \frac{\text{minimum congestion flow } f}{\text{maximum congestion of } f}$

Congestion Approximator

- Optimal congestion = maximal congested cut

Congestion Approximator

- Optimal congestion = maximal congested cut
- Congestion of 1 cut = 1 entry in congestion approximation vector

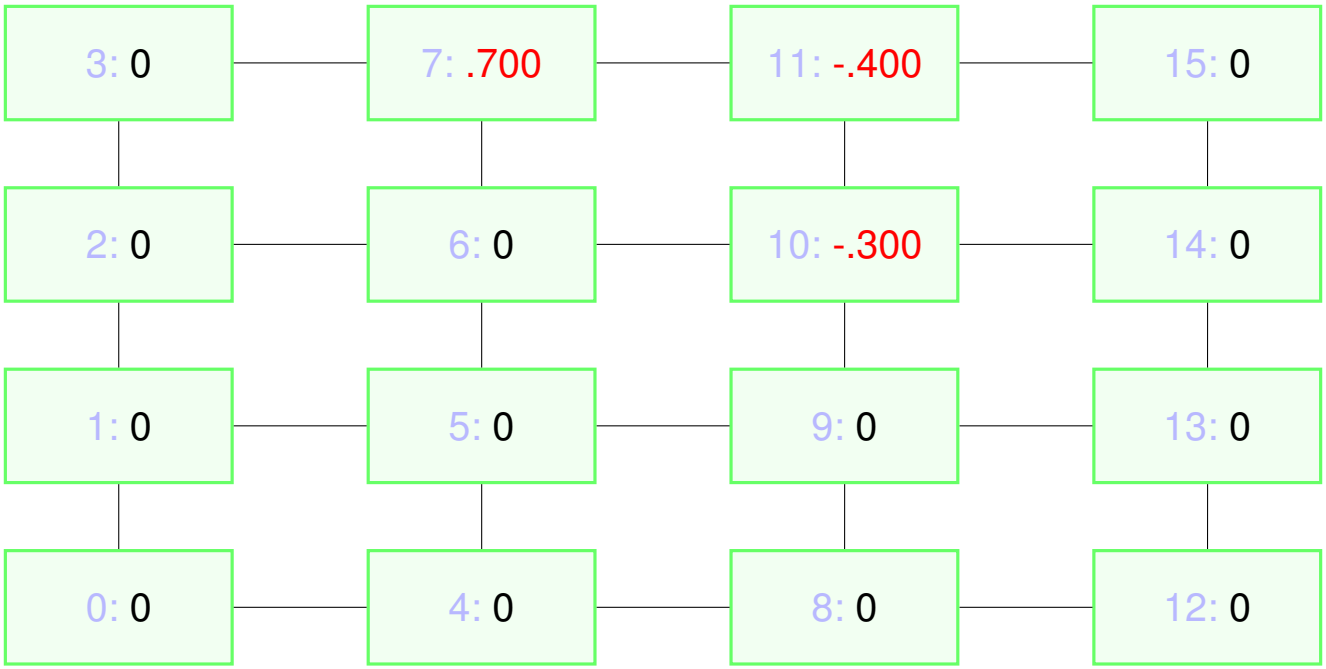
Congestion Approximator

- Optimal congestion = maximal congested cut
- Congestion of 1 cut = 1 entry in congestion approximation vector
- Maximum entry in congestion approximation vector = maximum congestion approximation

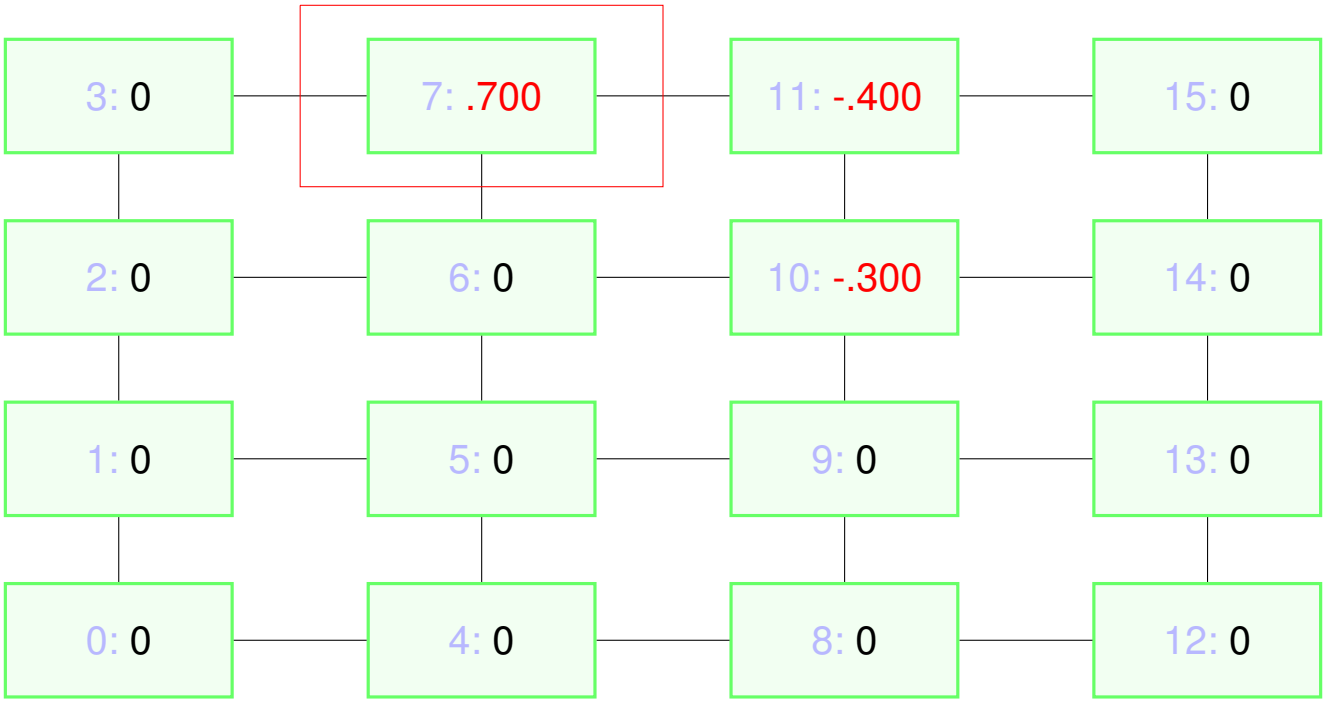
Congestion Approximator

- Optimal congestion = maximal congested cut
- Congestion of 1 cut = 1 entry in congestion approximation vector
- Maximum entry in congestion approximation vector = maximum congestion approximation
- Optimizing goal: minimizing maximum congestion via its approximation

Congestion Approximator

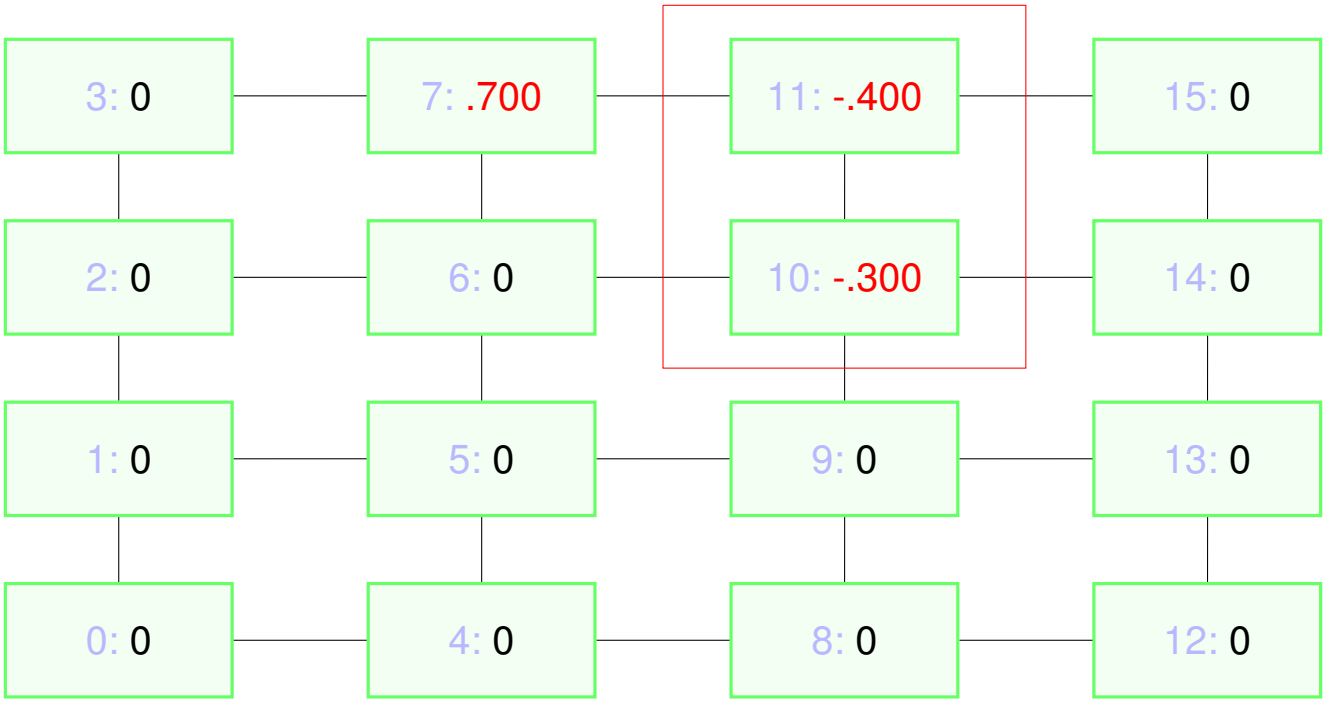


Congestion Approximator



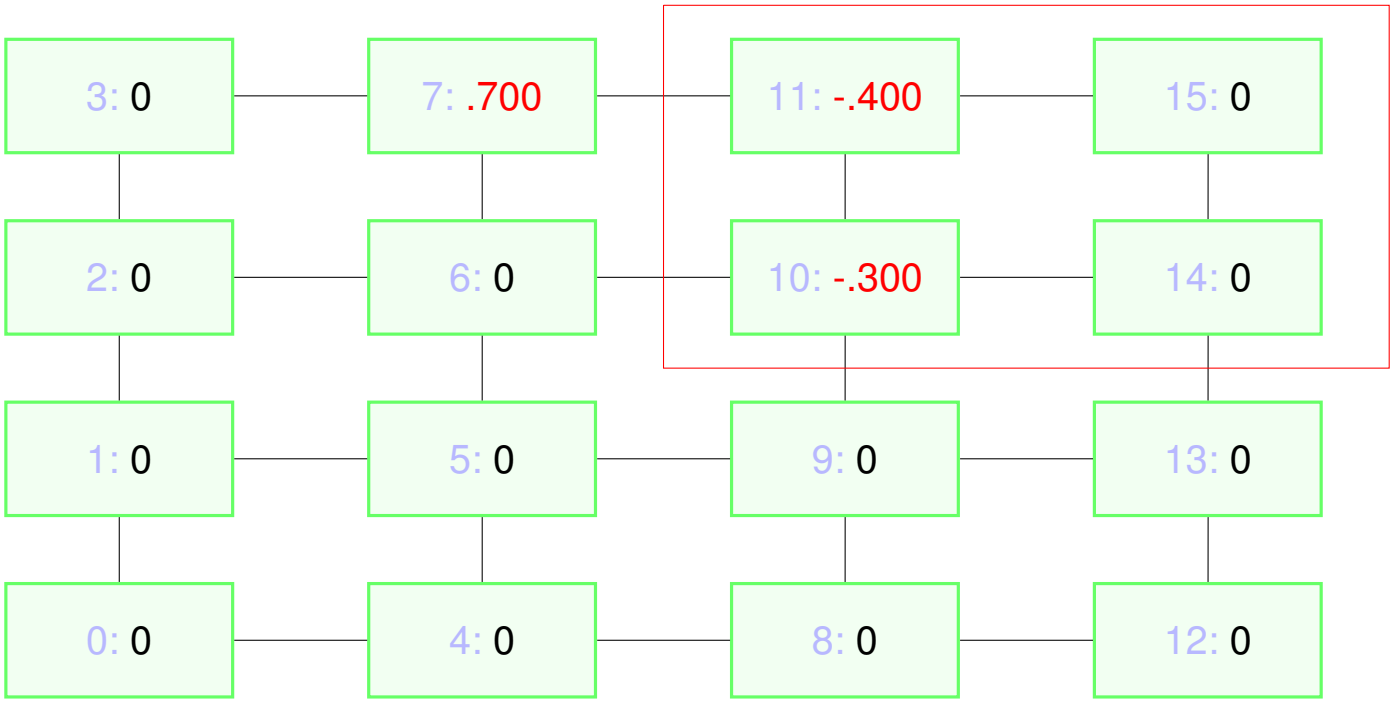
Congestion $\geq 0.7/3$

Congestion Approximator



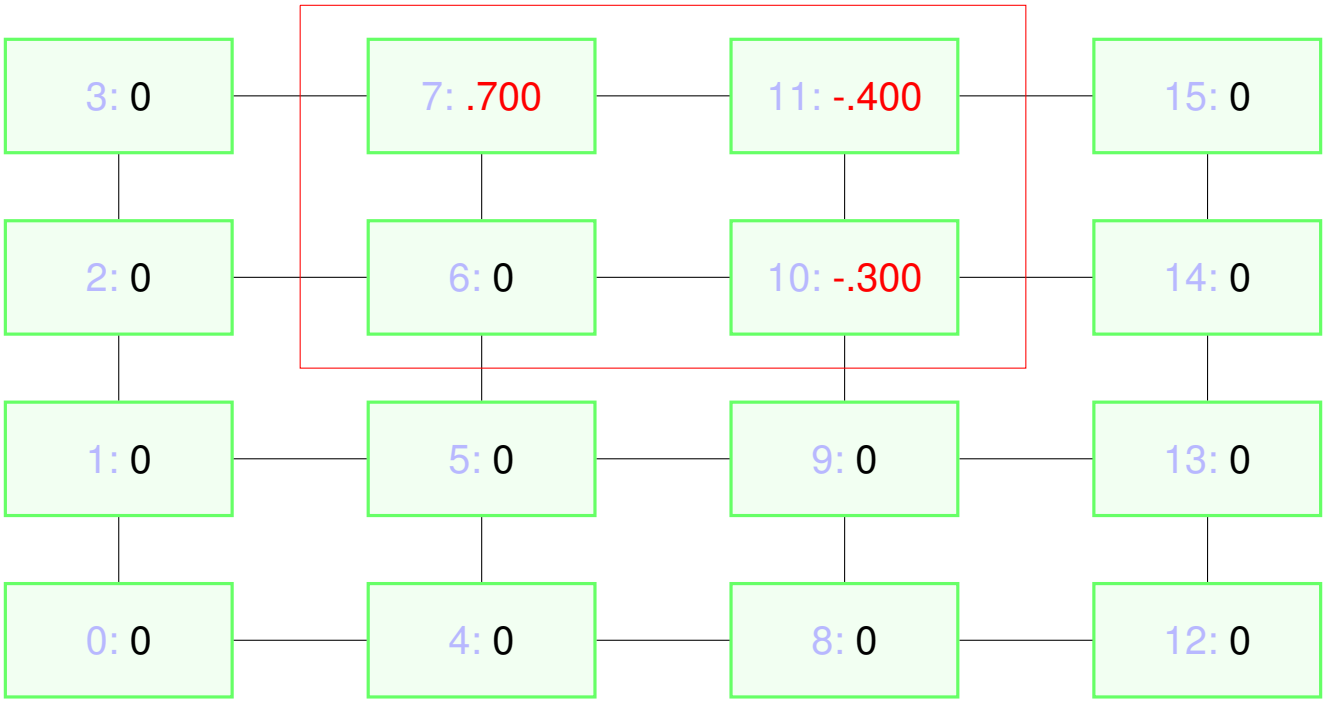
Congestion $\geq 0.7/5$

Congestion Approximator



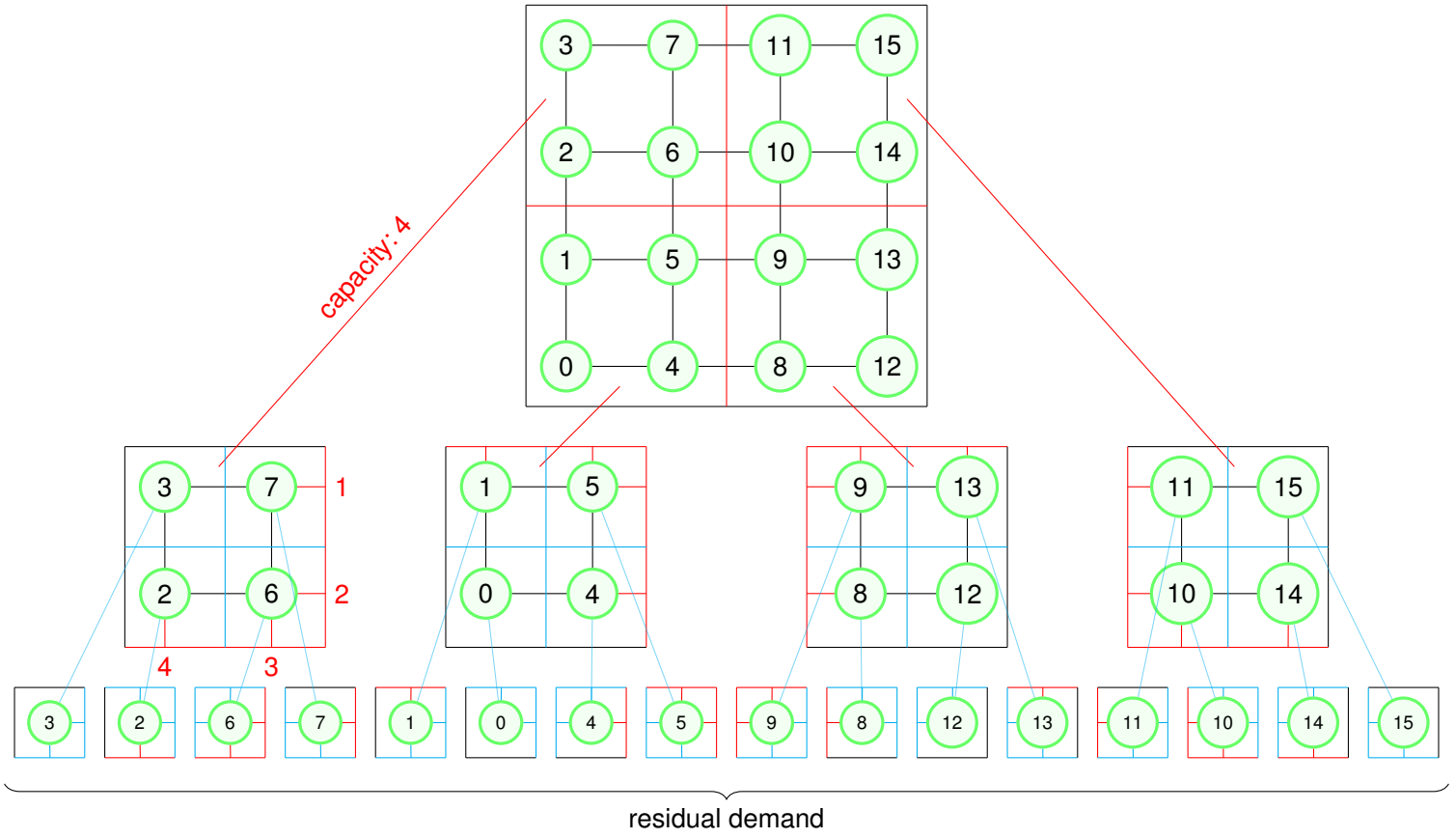
Congestion $\geq 0.7/4$

Congestion Approximator



Congestion ≥ 0

Approximator Scheme



α-Congestion Approximator

Definition:

$$\|Rb\|_\infty \leq \text{opt}(b) \leq \alpha \|Rb\|_\infty, \quad \text{for all demands } b$$

opt(b): Minimum Congestion for b

α-Congestion Approximator

Definition:

$$\|Rb\|_\infty \leq \text{opt}(b) \leq \alpha \|Rb\|_\infty, \quad \text{for all demands } b$$

opt(b): Minimum Congestion for b

Equivalent iff $Rb \neq 0$:

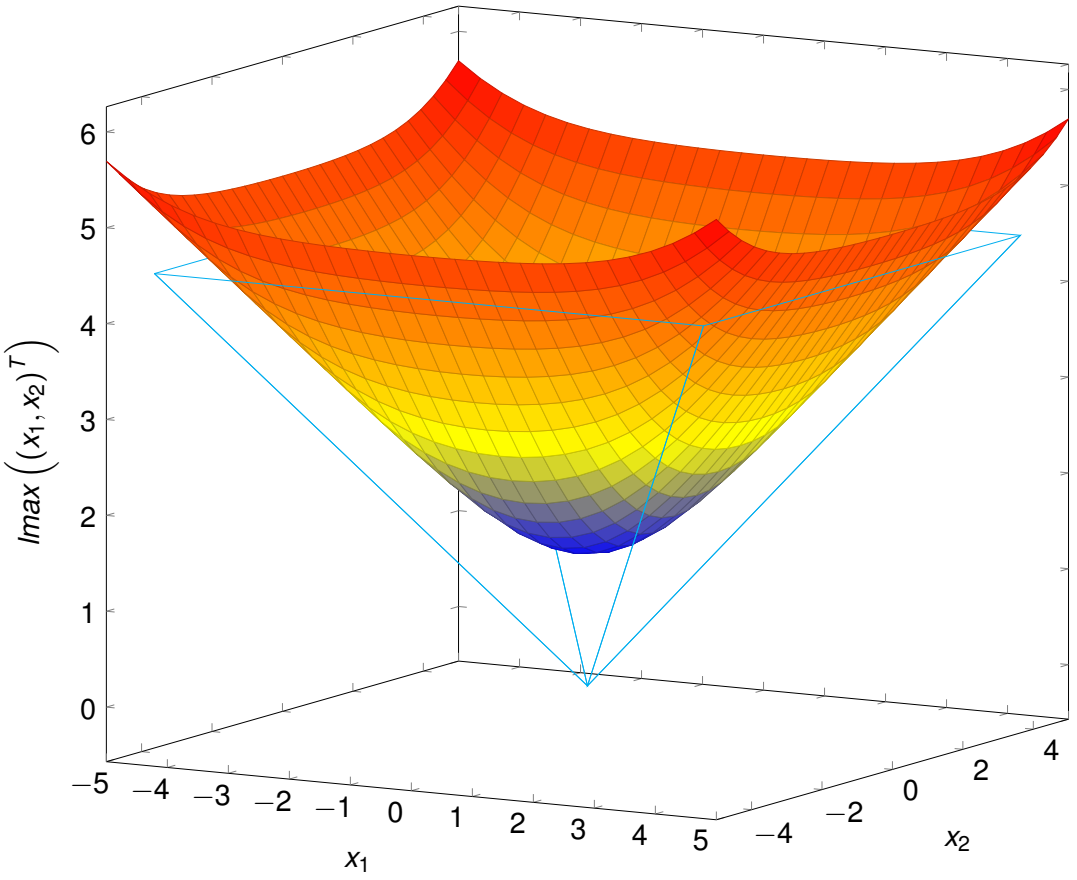
$$1 \leq \frac{\text{opt}(b)}{\|Rb\|_\infty} \leq \alpha, \quad \text{for all demands } b$$

Potential Function

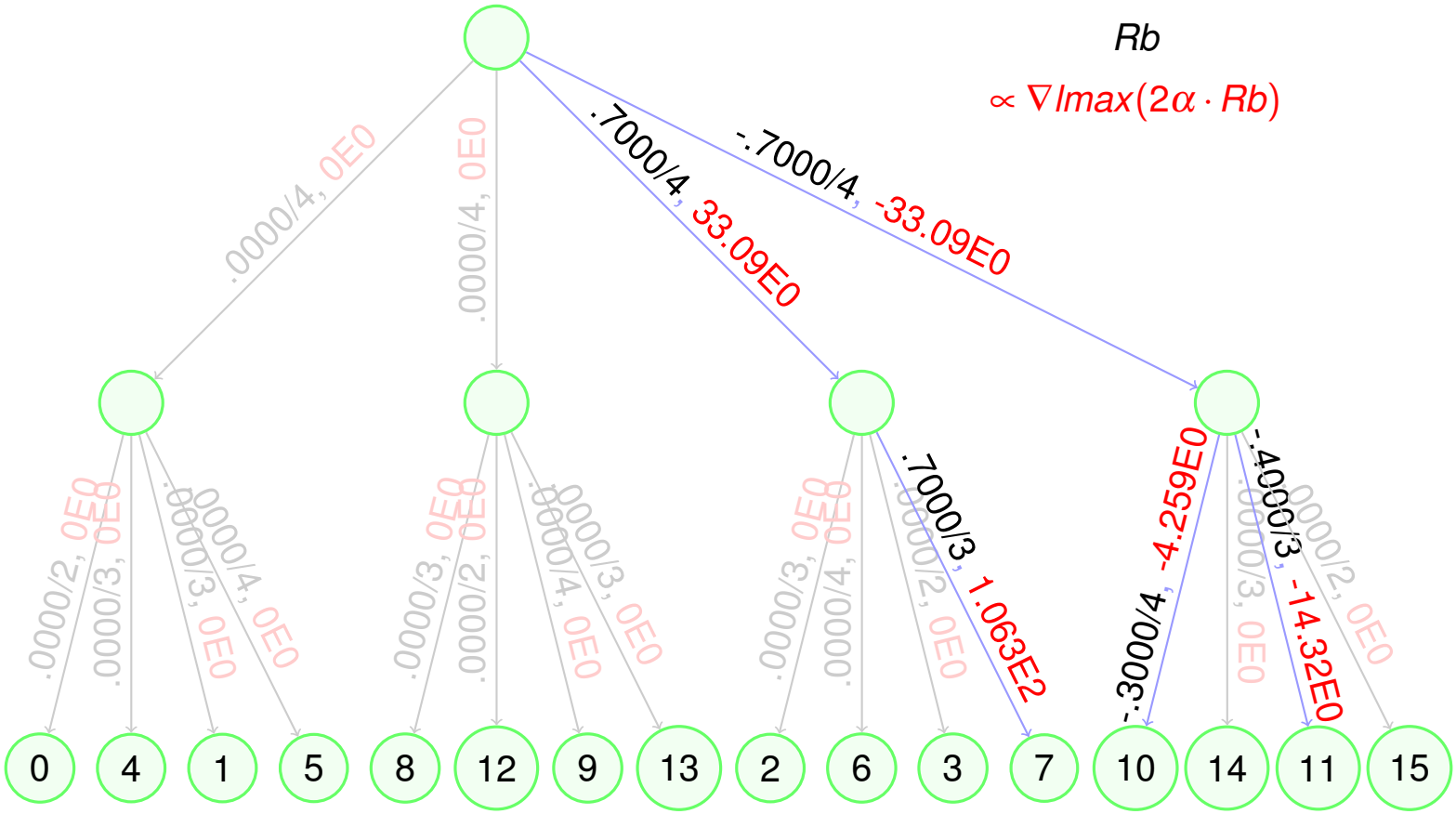
$$\Phi(f) := \left\| C^{-1} f \right\|_\infty + 2\alpha \left\| R(b - Bf) \right\|_\infty$$
$$\phi(f) := lmax\left(C^{-1} f\right) + lmax\left(2\alpha \cdot R(b - Bf)\right)$$
$$\nabla\phi(f) = \left(C^{-1}\right)^T \cdot \nabla lmax\left(C^{-1} f\right) - 2\alpha \cdot B^T R^T \cdot \nabla lmax(2\alpha \cdot R(b - Bf))$$

$$lmax(\vec{x}) := ln\left(\sum_i e^{x_i} + e^{-x_i}\right)$$
$$\left(\nabla lmax(\vec{x})\right)_j = \frac{e^{x_j} - e^{-x_j}}{\sum_i e^{x_i} + e^{-x_i}}$$

Potential Function



Congestion Approximator



Grid Graphs

Grid Graphs

- Undirected

Grid Graphs

- Undirected
- Unit-Capacity

Grid Graphs

- Undirected
- Unit-Capacity
- Multi-Dimensional

Grid Graphs

- Undirected
- Unit-Capacity
- Multi-Dimensional
- $\rightarrow$ can be defined via dimensional node count vector $(n_1, ..., n_d)$

Grid Graphs

Node count:

$$n = \prod_{i=1}^{d} n_i$$

Edge count:

$$m = \left(\sum_{i=1}^{d} \frac{n_i - 1}{n_i} \right) \cdot n$$

$$\rightarrow m \in \Theta(d \cdot n)$$

Grid Graphs

Capacity of hypercube cut $([a_1, b_1], \ldots, [a_d, b_d])$:

$$\sum_{i=1}^{d} \left((\overline{\delta}_{a_i,0} + \overline{\delta}_{b_i,n_i-1}) \cdot \prod_{j=1, j\neq i}^{d} (b_j - a_j + 1) \right)$$

with $\overline{\delta}_{x,y} = 1 - \delta_{x,y}$ (Kronecker-Delta)

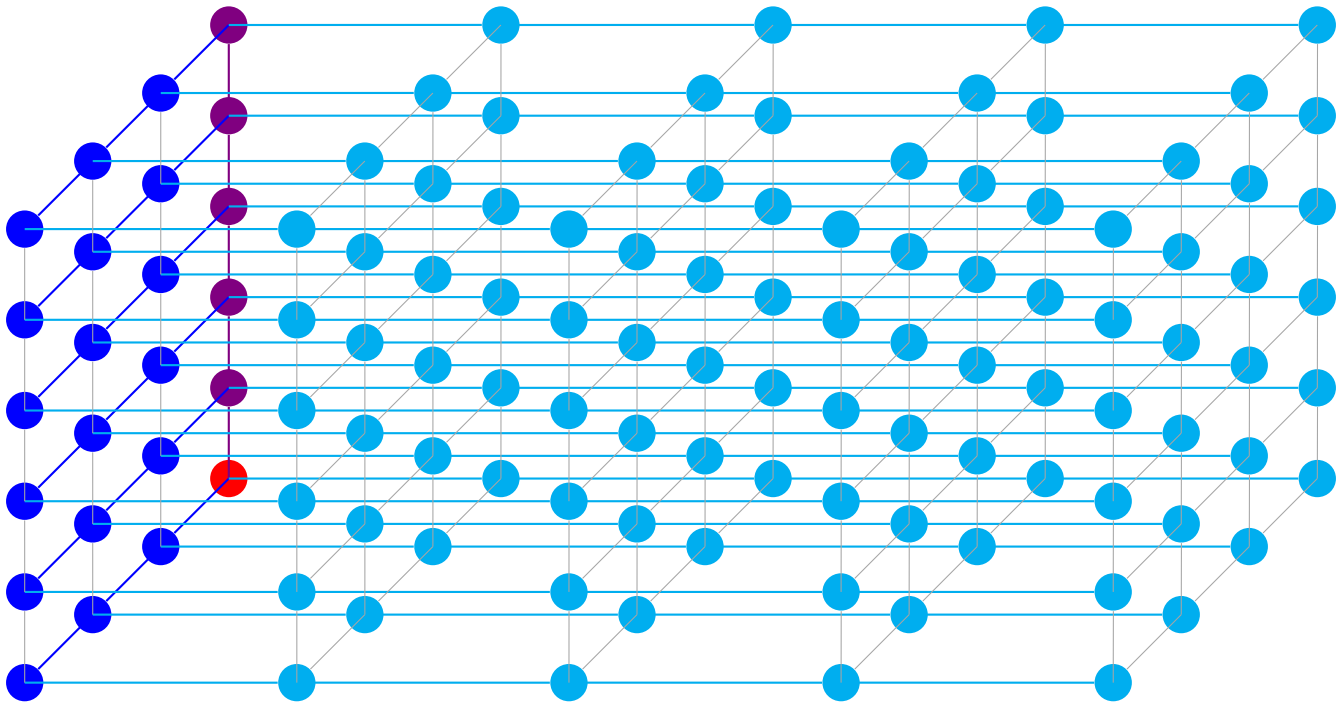
Algorithm

1: **procedure** CompleteRoute(b, ε)
2: $r \leftarrow b$
3: $(f_0, S_0) \leftarrow \texttt{AlmostRoute}(r, \varepsilon)$
4: **for** $i \leftarrow 1, \log_2(2m)$ **do**
5: $r \leftarrow r - Bf_{i-1}$
6: $(f_i, S_i) \leftarrow \texttt{AlmostRoute}(r, 1/2)$
7: **end for**
8: $r \leftarrow r - Bf_i$ ▷ Assuming $i = log(2m)$ after the loop.
9: $f_{i+1} \leftarrow \texttt{GridMST.route}(r)$
10: **return** $\sum_{j=0}^{i+1} f_j$
11: **end procedure**

Algorithm

1: **procedure** ALMOSTROUTE(b, ε)
2: **repeat**
3: **while** $\phi(f) < 16\varepsilon^{-1}\log(n)$ **do**
4: $f \leftarrow \frac{17}{16} \cdot f$
5: $b \leftarrow \frac{17}{16} \cdot b$
6: **end while**
7: $\delta \leftarrow \|C\nabla\phi(f)\|_1$
8: **if** $\delta \geq \varepsilon/4$ **then**
9: $f_e \leftarrow f_e - \frac{\delta}{1+4\alpha^2}\text{sgn}(\nabla\phi(f)_e)c_e$
10: **end if**
11: **until** $\delta < \varepsilon/4$
12: **return** f, $\nabla\phi(f)$
13: **end procedure**

Maximal Spanning Tree



MST Routing

Algorithm 1 Routing Demand b through a (Maximal) Spanning Tree of Grid Graph G

```
1: procedure RouteMST(GridGraph G, GridDemand b)
2:     f ← 0
3:     r ← b                                                    ▷ Residual demand
4:     (n_1, ..., n_d) ← G.nodesPerDimension
5:     for i ← 1, d do
6:         for j ← 1, i − 1 do                                  ▷ Set v_start and v_end properly
7:             (v_start)_j ← 0
8:             (v_end)_j ← 0
9:         end for
10:        (v_start)_i ← n_i − 1
11:        (v_end)_i ← 1
12:        for j ← i + 1, d do
13:            (v_start)_j ← n_j − 1
14:            (v_end)_j ← 0
15:        end for
16:        for v ← v_start, v_end do
17:            u ← v
18:            u_i ← v_i − 1
19:            e ← (u, v)
20:            f_e ← r_v                                        ▷ r_v: Residual demand of leaf v. Every iterated v is a leaf iff the iteration is valid.
21:            r_u ← r_u + r_v
22:            r_v ← 0
23:        end for
24:    end for
25:    return f
26: end procedure
```

MST Routing

Algorithm 2 Simplified Version

```
1:  procedure RouteMST2(GridGraph G, GridDemand b)
2:      f ← 0
3:      r ← b                                          ▷ Residual demand
4:      i ← 1
5:      for k ← n − 1, 1 do
6:          v⃗ ← toVector(k)
7:          while v⃗_i = 0 do
8:              i ← i + 1
9:          end while
10:         v⃗_i ← v⃗_i − 1
11:         u ← toIndex(v⃗)
12:         e ← (u, k)
13:         f_e ← r_k
14:         r_u ← r_u + r_k
15:         r_k ← 0
16:     end for
17:     return f
18: end procedure
```

Implementation

Implementation

- Basic DS & Algorithm(s)

Implementation

- Basic DS & Algorithm(s)
- Step Size Optimization

Implementation

- Basic DS & Algorithm(s)
- Step Size Optimization
- Empiric Search for α

Step Size Optimization

Step Size Optimization

- Default: $f_i \leftarrow f_i - s_i$

Step Size Optimization

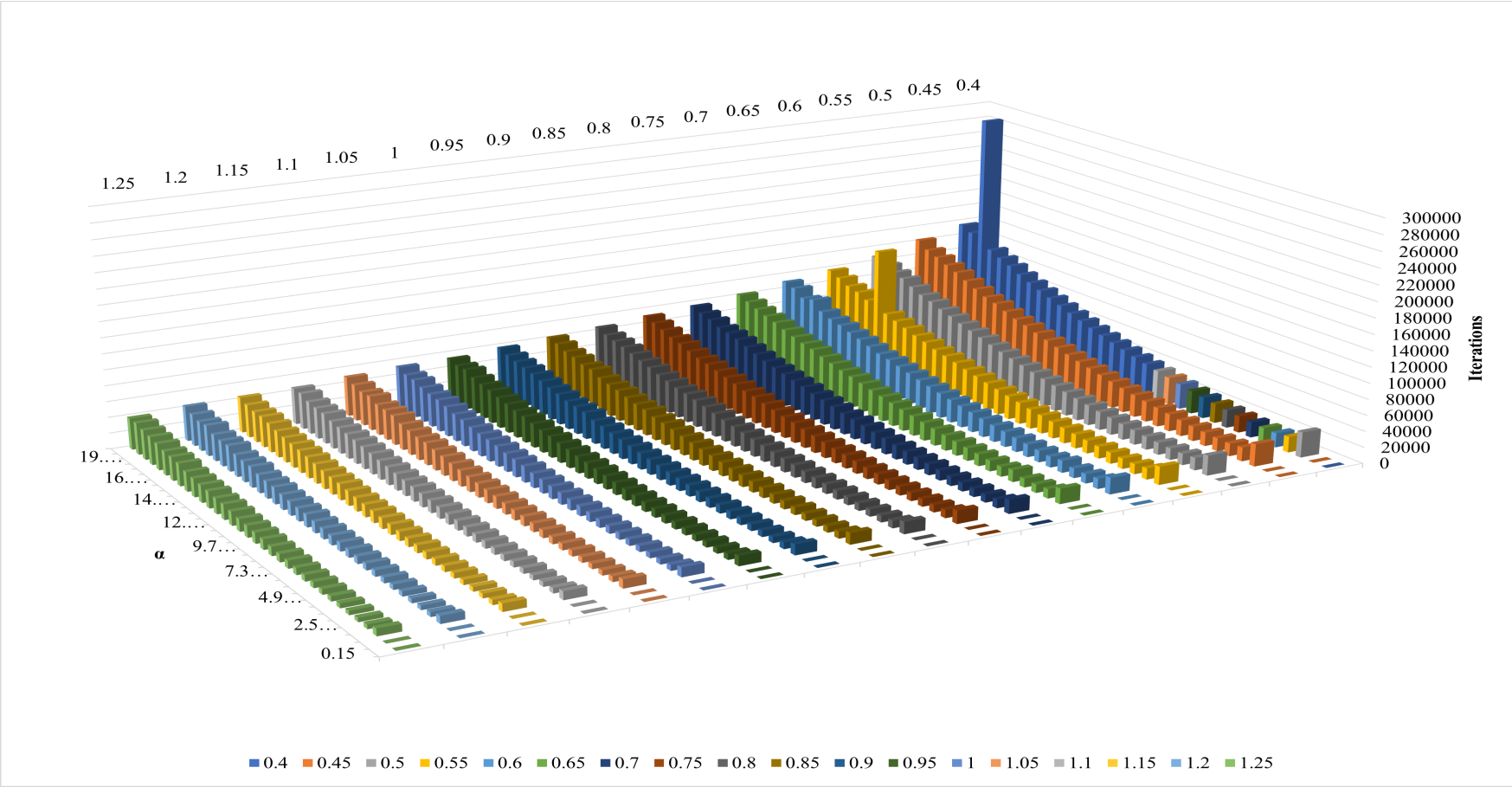
- Default: $f_i \leftarrow f_i - s_i$
- Approach: $f_i \leftarrow f_i - h \cdot s_i$

Step Size Optimization

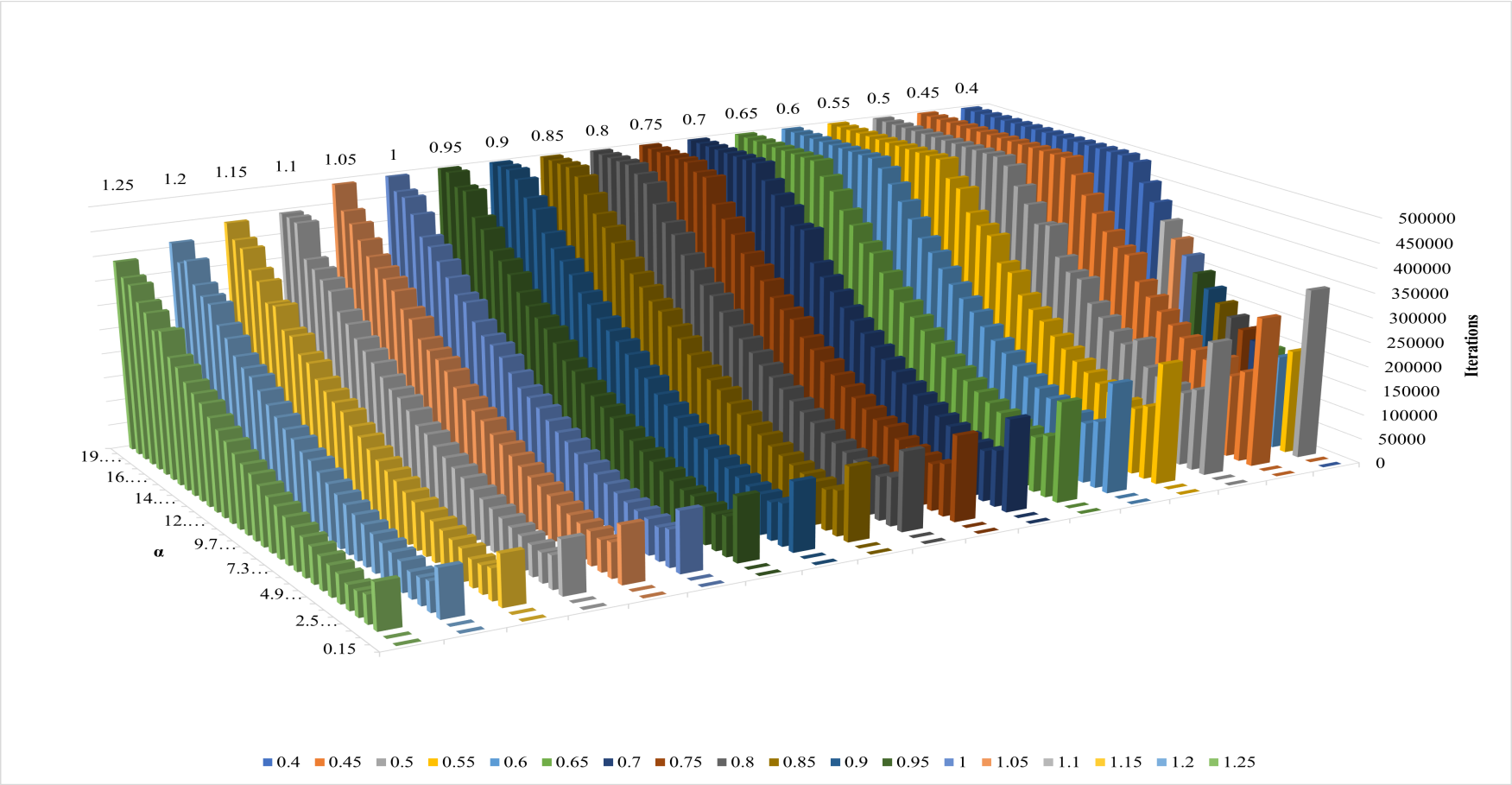
- Default: $f_i \leftarrow f_i - s_i$
- Approach: $f_i \leftarrow f_i - h \cdot s_i$

$$\frac{\vartheta}{\vartheta h}\phi(f - h \cdot s) = (-C^{-1}s)^T \cdot \nabla lmax\left(C^{-1} \cdot (f - h \cdot s)\right) + (2\alpha RBs)^T \cdot \nabla lmax\left(2\alpha R(b - B(f - h \cdot s))\right)$$

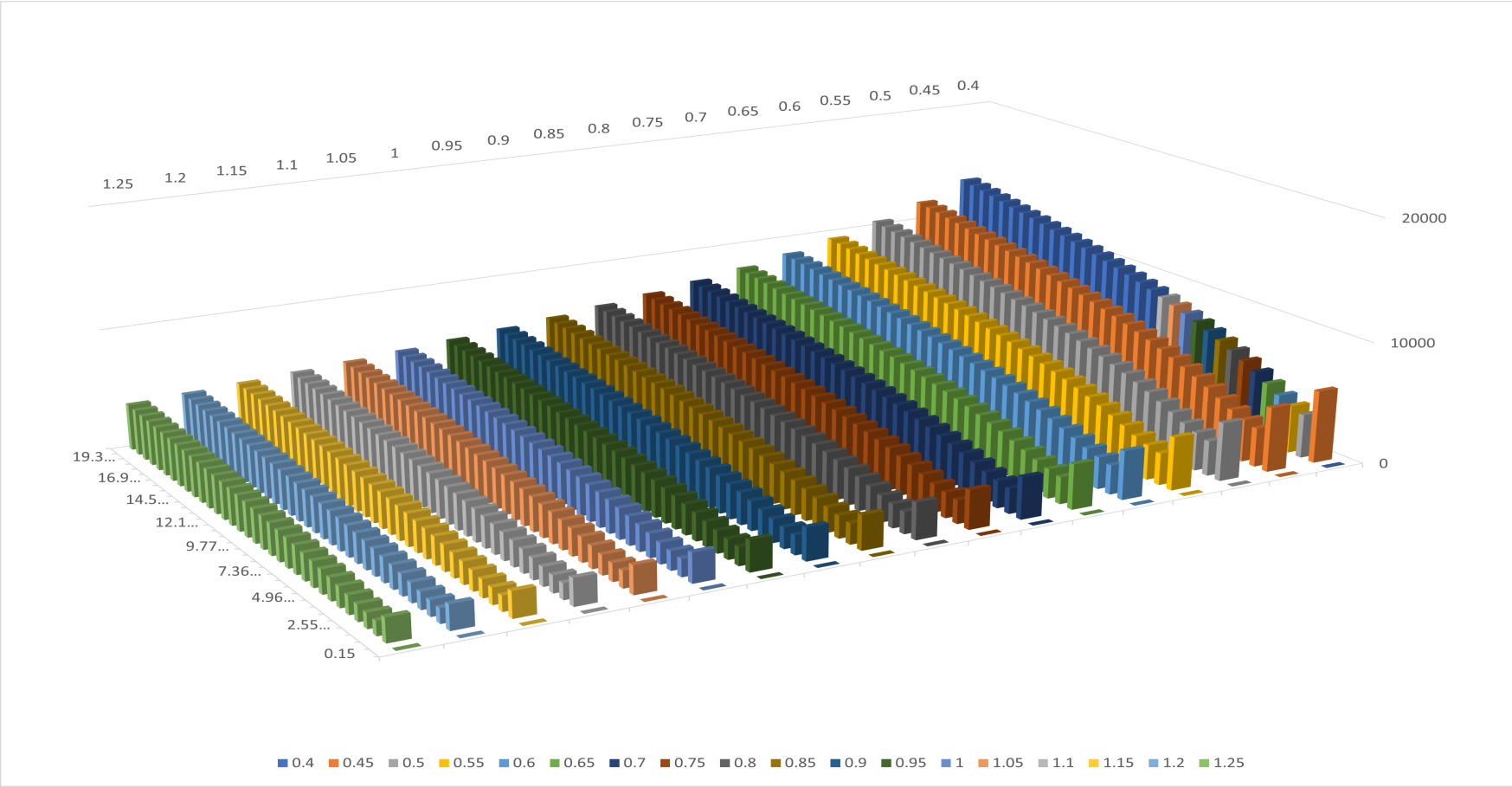
Static Optimization



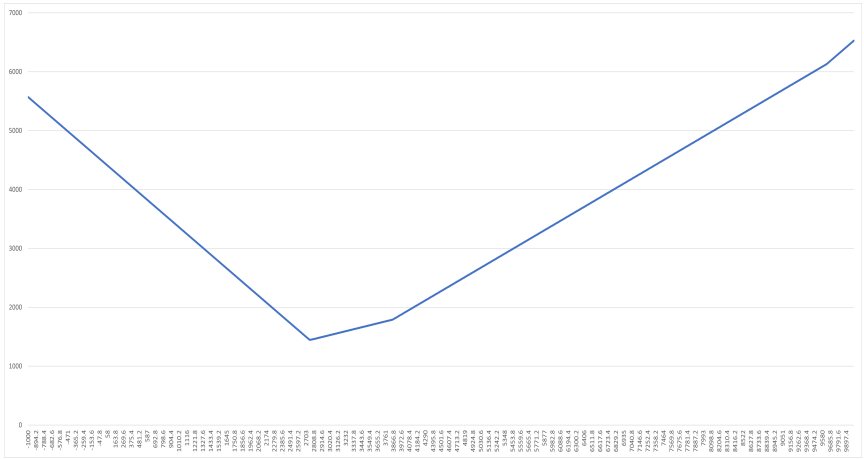
Static Optimization

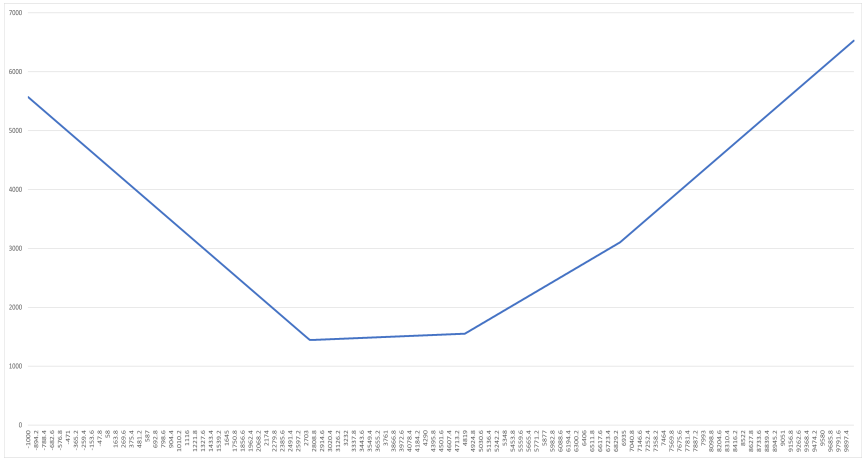


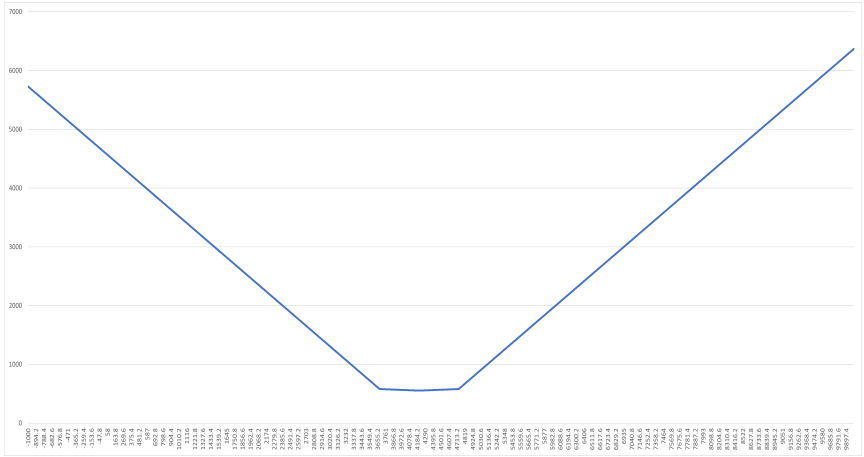
Static Optimization



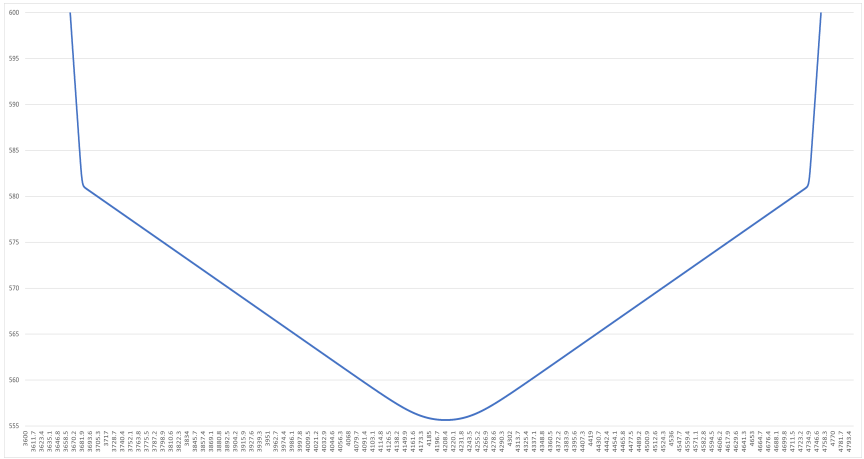
$$\phi(f - h \cdot s)$$

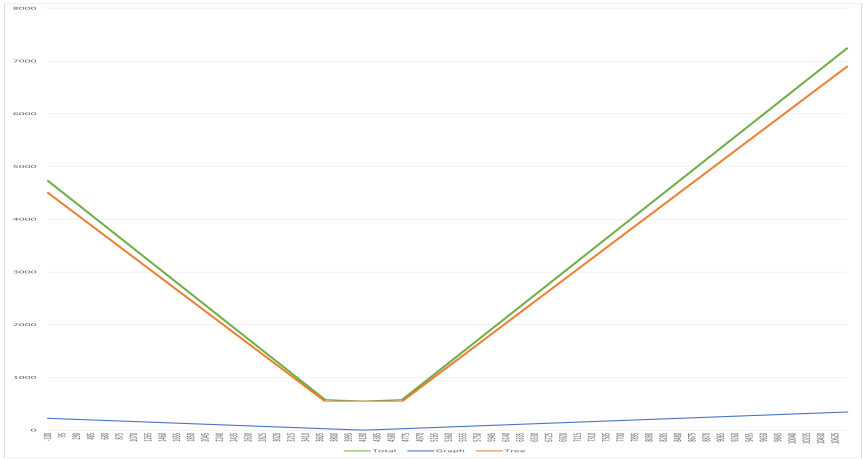


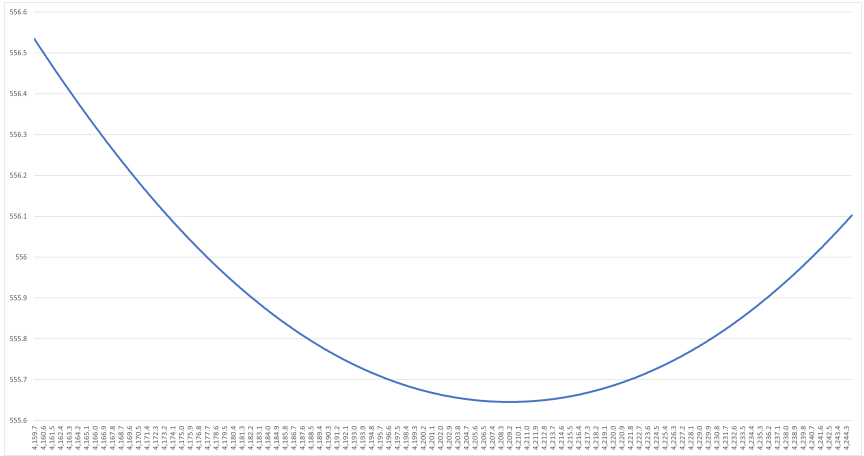


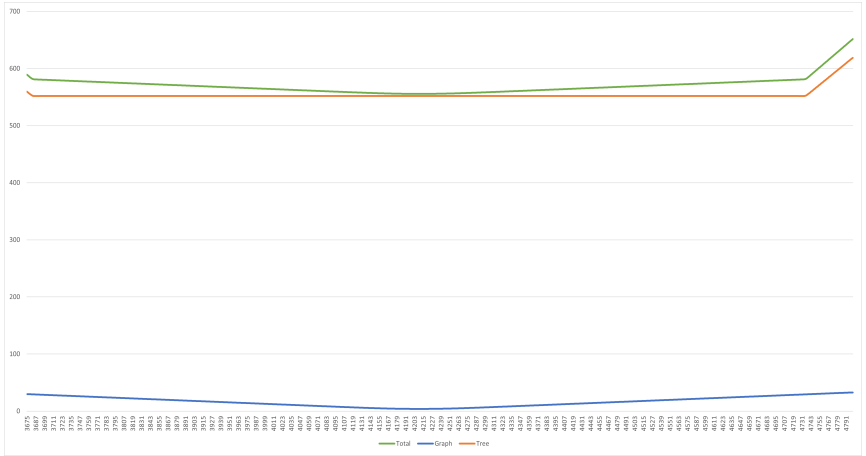


$$\phi(f - h \cdot s)$$

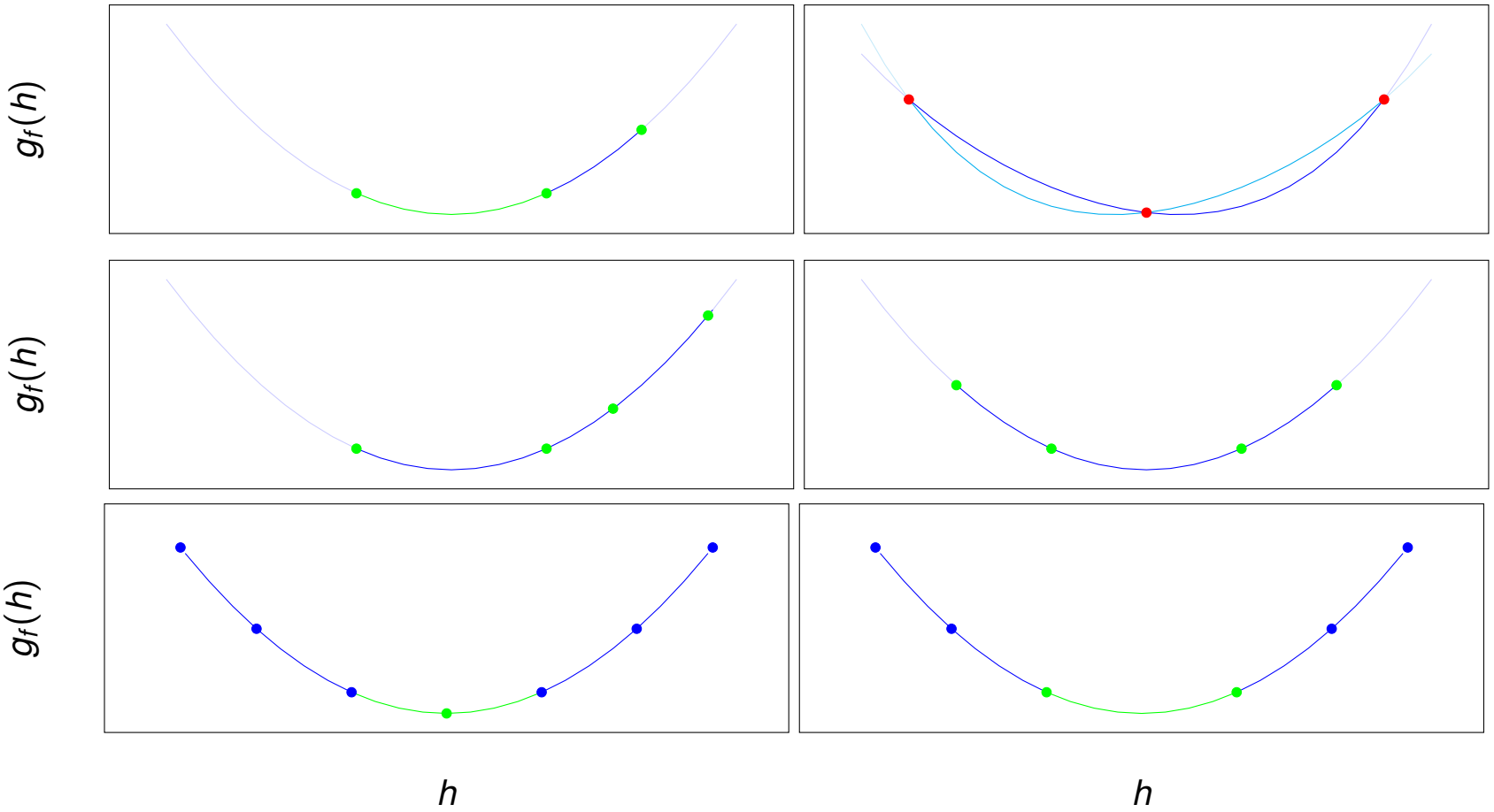

600
595
590
585
580
575
570
565
560
555


8000
7000
6000
5000
4000
3000
2000
1000
0
Total
Graph
Tree

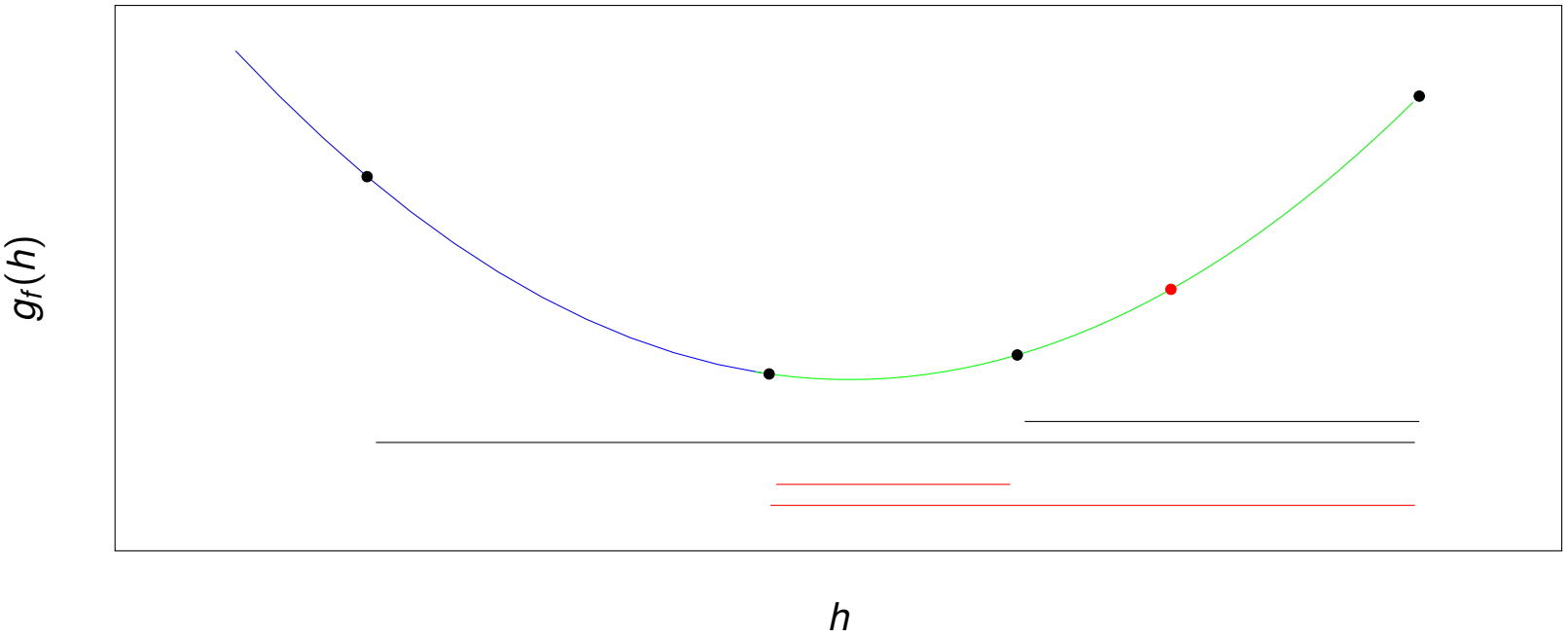

556.6
556.5
556.4
556.3
556.2
556.1
556
555.9
555.8
555.7
555.6


700
600
500
400
300
200
100
0
Total
Graph
Tree

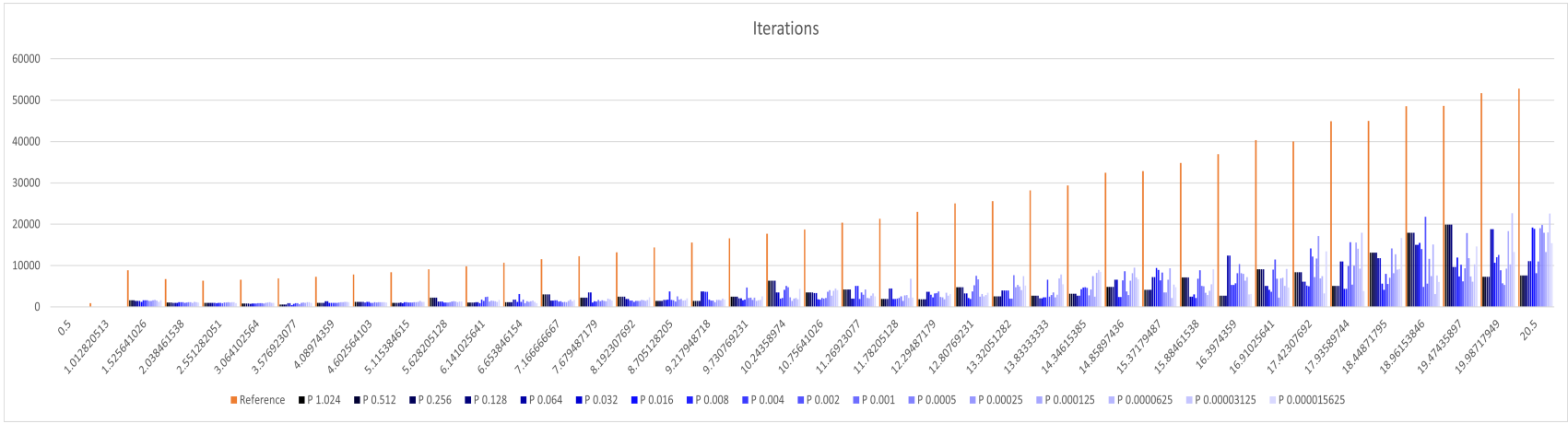
Line Search

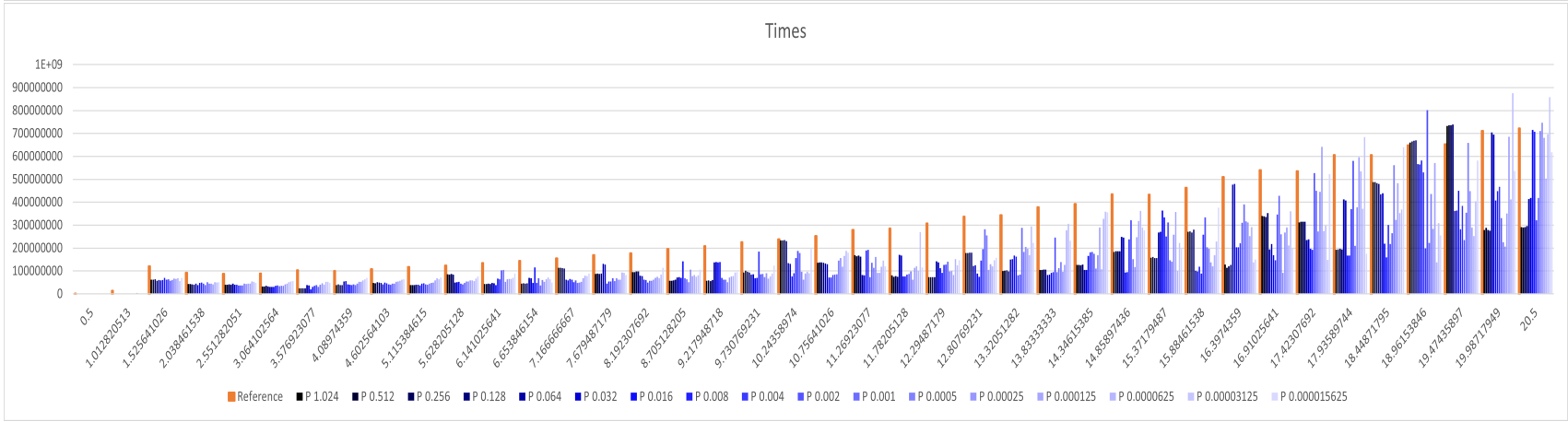


Golden Section Search

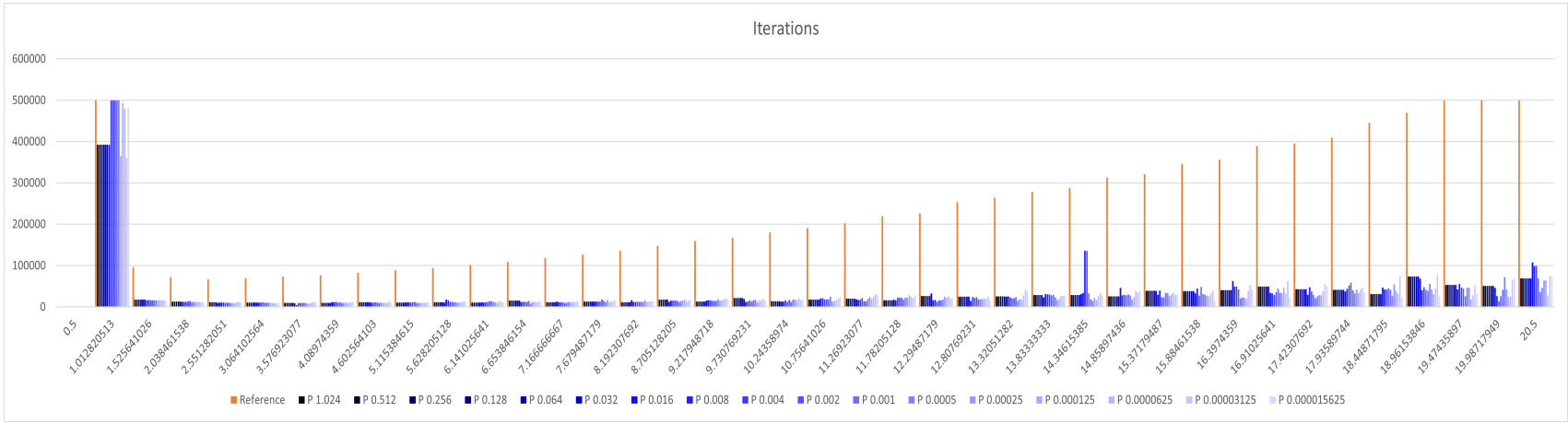


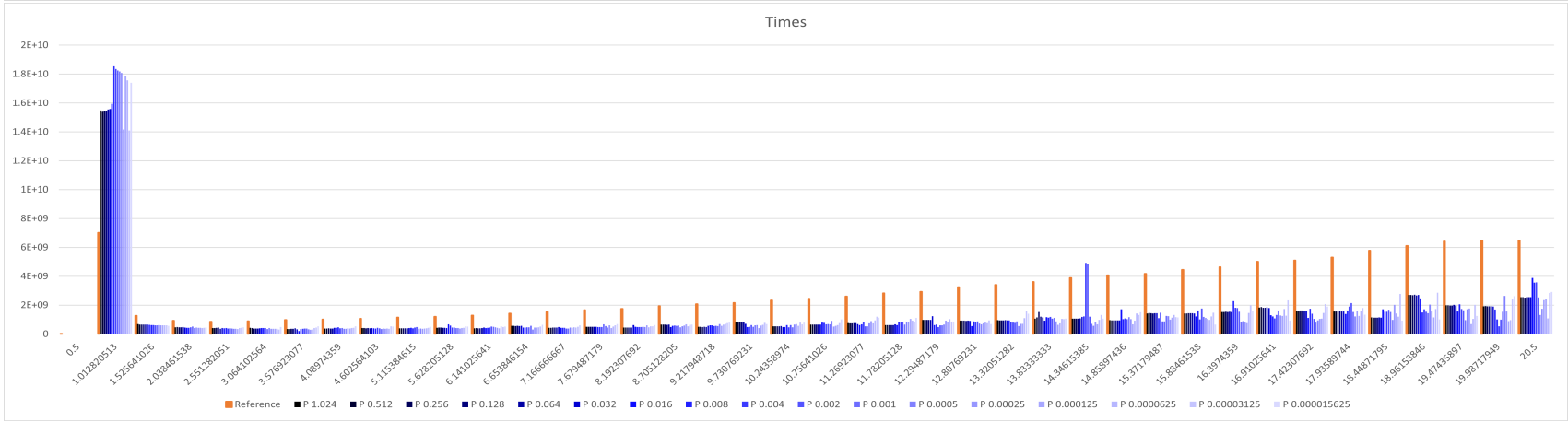
Dynamic Optimization Results



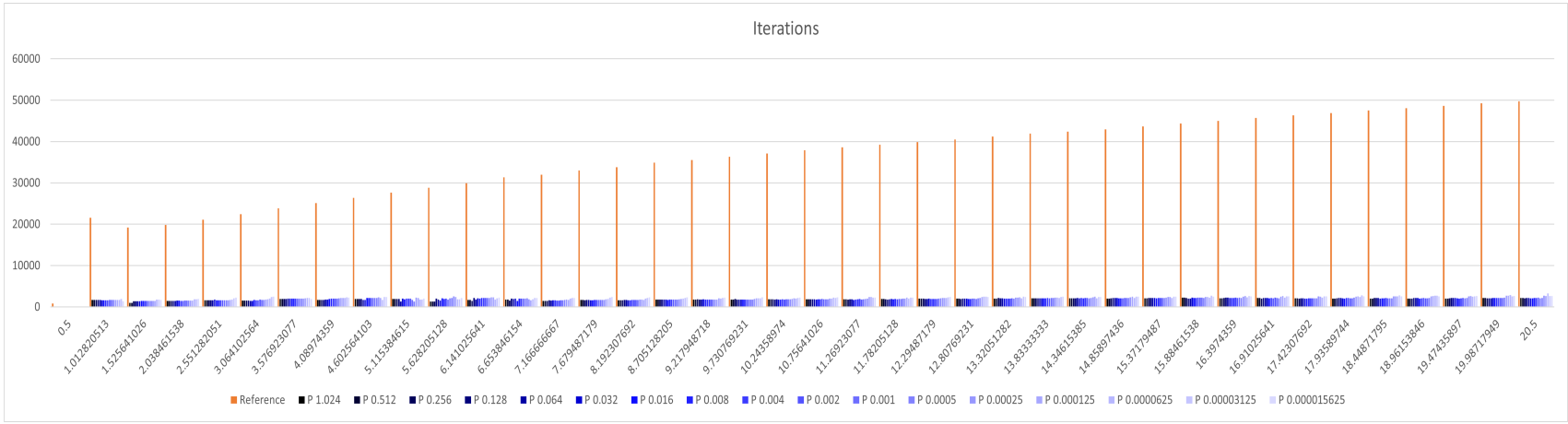


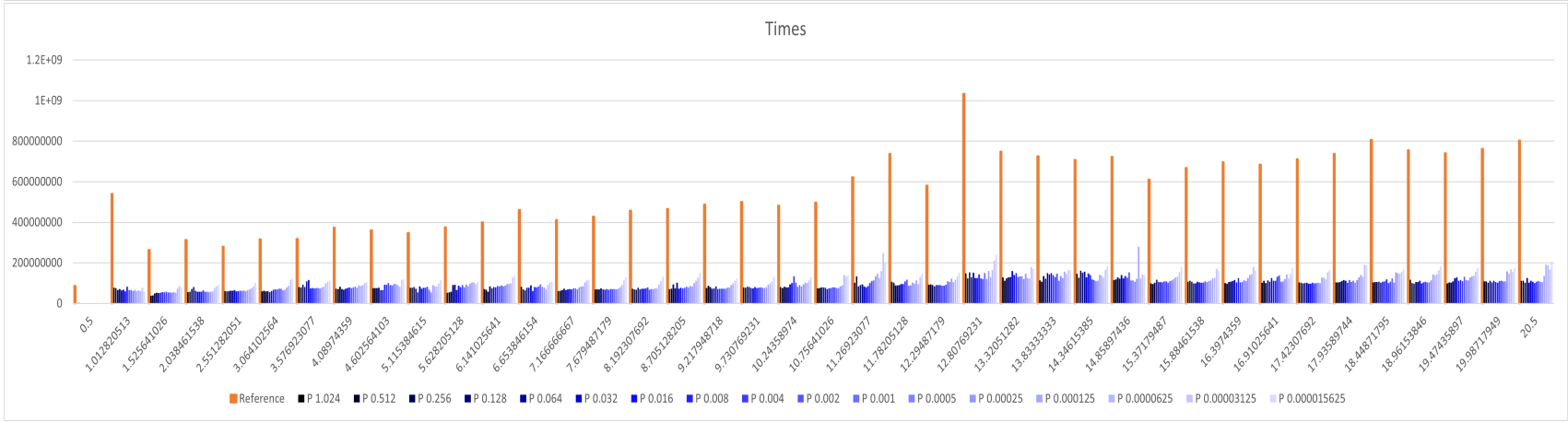
Dynamic Optimization Results





Dynamic Optimization Results





Empiric Search for α

Empiric Search for α

- Random Sampling for 1D grid graphs

Empiric Search for α

- Random Sampling for 1D grid graphs
- 1D-sampling usable in higher dimensions

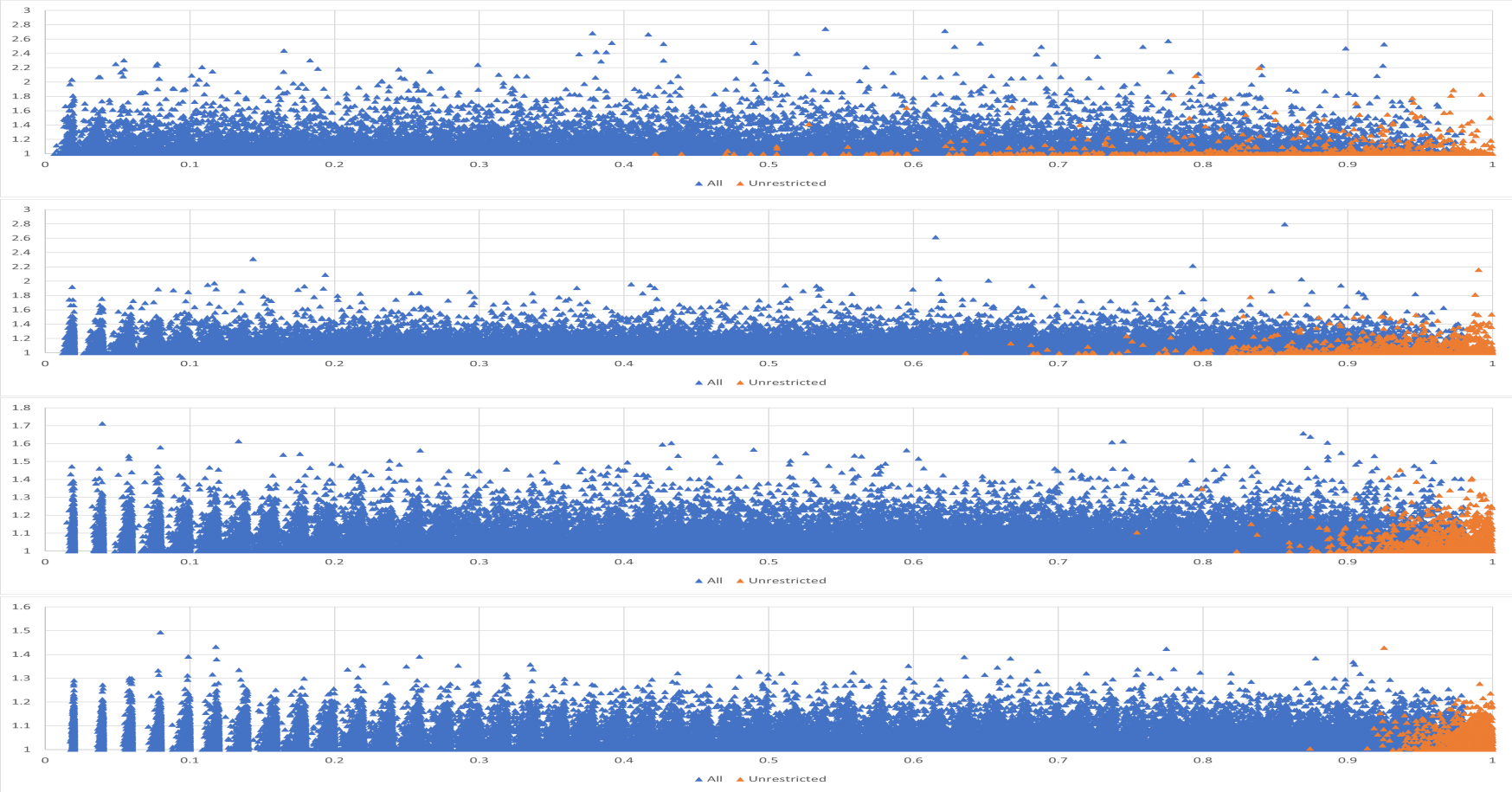
Empiric Search for α

- Random Sampling for 1D grid graphs
- 1D-sampling usable in higher dimensions
- Randomized Cut with Maximum Congestion

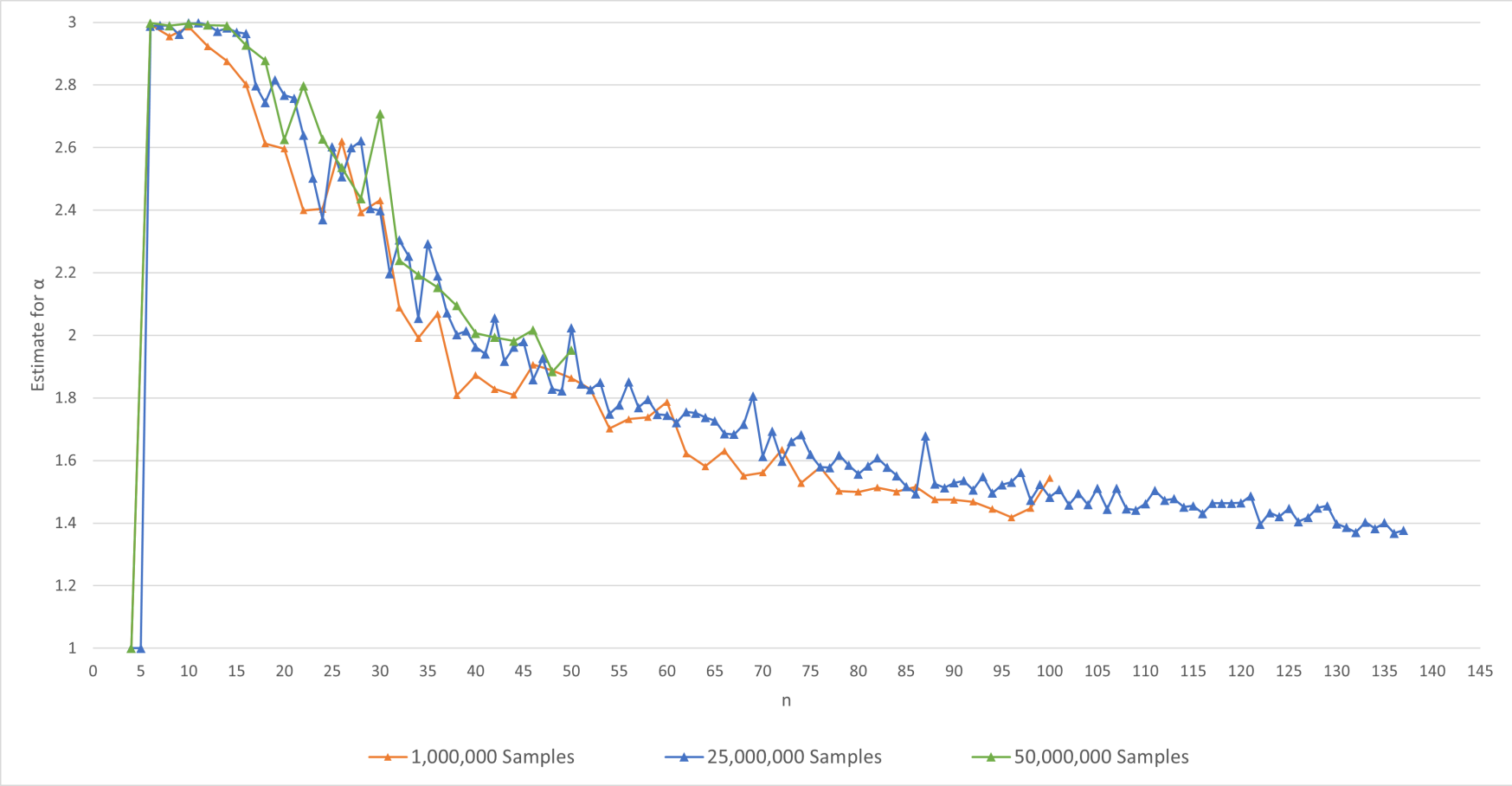
Empiric Search for α

- Random Sampling for 1D grid graphs
- 1D-sampling usable in higher dimensions
- Randomized Cut with Maximum Congestion
- Pre-Calculating Optimal Maximum Congestion

1D-Sampling for α



1D-Sampling for α



References

- GitHub Repository: `https://github.com/js97/approximate-flows`
- Master's Thesis: `https://github.com/js97/approximate-flows/blob/main/Thesis-Paper/Paper.pdf`
- Sherman's Approximative Maxflow Algorithm [She13]: Jonah Sherman. "Nearly Maximum Flows in Nearly Linear Time". In: *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. 2013, pp. 263-269. DOI: 10.1109/FOCS.2013.36.