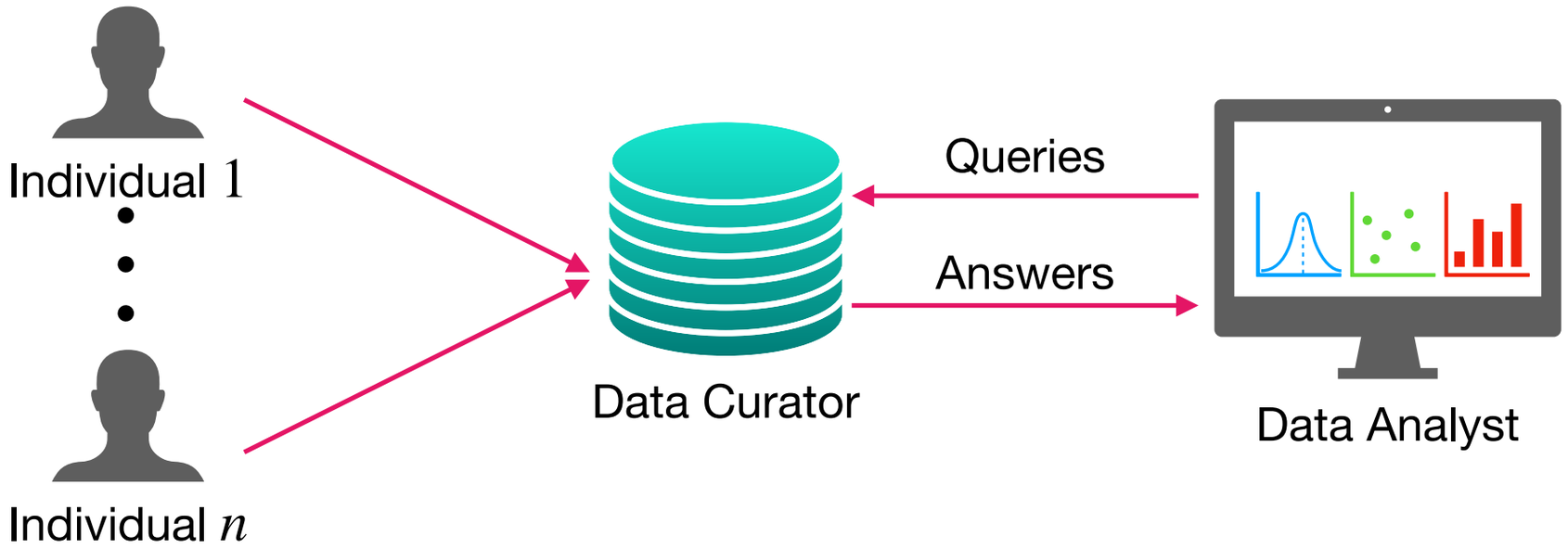
Heterogeneous Differential Privacy via Graphs

Sahel Torkamani
Javad B. Ebrahimi
Parastoo Sadeghi
Rafael G. L. D' Oliveira
Muriel Médard

Sharif University of Technology, Iran
Institute of Science and Technology Austria
Institute for Research in Fundamental Sciences, Iran
University of New South Wales, Australia
Clemson University, USA
Massachusetts Institute of Technology, USA

Private Data Analysis



- Individuals share private data with a curator.
- The Curator manages the release of the data to Analysts.
- Two conflicting goals:
  - **Privacy:** protecting the individuals' privacy.
  - **Utility:** learning useful information about the population.

What is **Differential Privacy?**

- First proposed in 2006 **[Dwork et al. '06]**.
- **Goal:** To learn as little as possible about an individual while learning useful information about a population.
- **Idea:** To add a controlled amount of randomness to the dataset.
- **Highlighted applications:**
  - Google, for sharing historical traffic statistics **[Erlingsson et al. '14]**.
  - Apple's private learning of users' preferences **[Apple DP team '17]**.
  - Microsoft for telemetry in Windows **[Ding et al. ’17]**.
  - The 2020 United States Census.

Some Issues with Traditional Differential Privacy

- A “one-size-fits-all" approach to setting a global privacy level can be damaging to both utility and privacy. **[Jorgensen et al. '15]**

- Some groups might need more protection than others. **[Kohli et al. '18]**

- There may also be statutory mandates, demanding publication of certain datasets with more accuracy. (e.g. US Census)

Main Contributions

- The notion of **Heterogeneous Differential Privacy**.

- Characterizing **optimal** schemes for binary functions.

- Presenting a **low complexity algorithm** for finding such schemes.

Traditional Differential Privacy

- Family of datasets $\mathscr{D}$.

- A symmetric relationship in $\mathscr{D}$ where $d \sim d'$ are said to be neighbors.

Traditional Differential Privacy

- Family of datasets $\mathscr{D}$.
- A symmetric relationship in $\mathscr{D}$ where $d \sim d'$ are said to be neighbors.
- An output space $\mathscr{Q}$.
- True function $T : \mathscr{D} \to \mathscr{Q}$.

Traditional Differential Privacy

- Family of datasets $\mathscr{D}$.
- A symmetric relationship in $\mathscr{D}$ where $d \sim d'$ are said to be neighbors.
- An output space $\mathscr{Q}$.
- True function $T : \mathscr{D} \to \mathscr{Q}$.
- Random function $\mathscr{M} : \mathscr{D} \to \mathscr{Q}$ called random mechanism.

Traditional Differential Privacy

- Family of datasets $\mathscr{D}$.
- A symmetric relationship in $\mathscr{D}$ where $d \sim d'$ are said to be neighbors.
- An output space $\mathscr{Q}$.
- True function $T : \mathscr{D} \to \mathscr{Q}$.
- Random function $\mathscr{M} : \mathscr{D} \to \mathscr{Q}$ called random mechanism.

Definition: Differential Privacy [Dwork et al '06]

$\mathscr{M}$ is ε-differentially private if, for any $d \sim d'$ and $\mathcal{S} \subseteq \mathscr{D}$,

$$\Pr[\mathscr{M}(d) \in \mathcal{S}] \leq e^{\varepsilon} \Pr[\mathscr{M}(d') \in \mathcal{S}]$$

Traditional Differential Privacy

- Family of datasets $\mathscr{D}$.
- A symmetric relationship in $\mathscr{D}$ where $d \sim d'$ are said to be neighbors.
- An output space $\mathscr{Q}$.
- True function $T : \mathscr{D} \to \mathscr{Q}$.
- Random function $\mathscr{M} : \mathscr{D} \to \mathscr{Q}$ called random mechanism.

Definition: Differential Privacy [Dwork et al '06]

$\mathscr{M}$ is ε-differentially private if, for any $d \sim d'$ and $\mathcal{S} \subseteq \mathscr{D}$,

$$\Pr[\mathscr{M}(d) \in \mathcal{S}] \leq e^{\varepsilon} \Pr[\mathscr{M}(d') \in \mathcal{S}]$$

- **Goal:** Approximate the true function T by an ε-DP mechanism $\mathscr{M}$.

Traditional Differential Privacy

- Family of datasets $\mathscr{D}$.
- A symmetric relationship in $\mathscr{D}$ where $d \sim d'$ are said to be neighbors.
- An output space $\mathscr{Q}$. **We consider binary $\mathscr{Q} = \{1,2\}$.**
- True function $T : \mathscr{D} \to \mathscr{Q}$.
- Random function $\mathscr{M} : \mathscr{D} \to \mathscr{Q}$ called random mechanism.

Definition: Differential Privacy [Dwork et al '06]

$\mathscr{M}$ is ε-differentially private if, for any $d \sim d'$ and $\mathcal{S} \subseteq \mathscr{D}$,

$$\Pr[\mathscr{M}(d) \in \mathcal{S}] \leq e^{\varepsilon} \Pr[\mathscr{M}(d') \in \mathcal{S}]$$

- **Goal:** Approximate the true function T by an ε-DP mechanism $\mathscr{M}$.

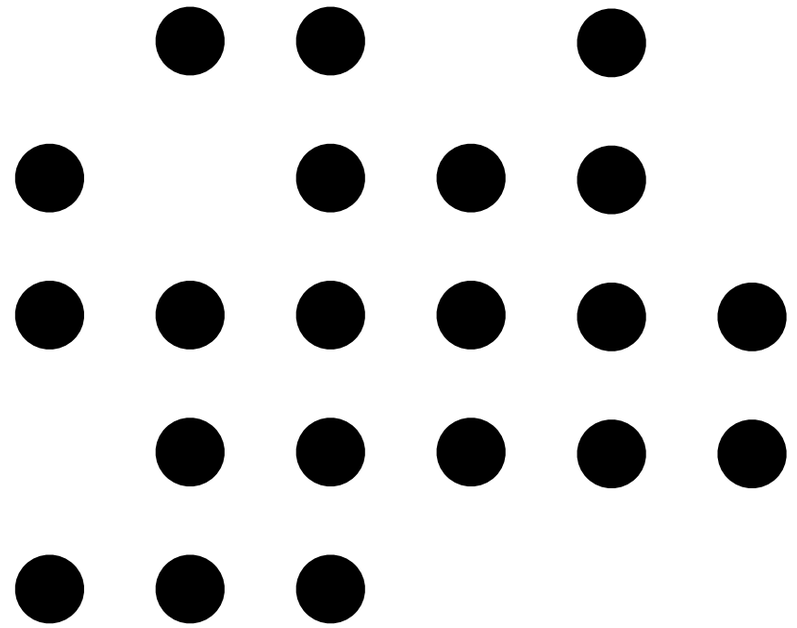
Differential Privacy via Graphs

I. In [D'Oliveira et al. 21'], differential privacy was presented in a graph theoretical framework, where:

Differential Privacy via Graphs

I. In [D'Oliveira et al. 21'], differential privacy was presented in a graph theoretical framework, where:

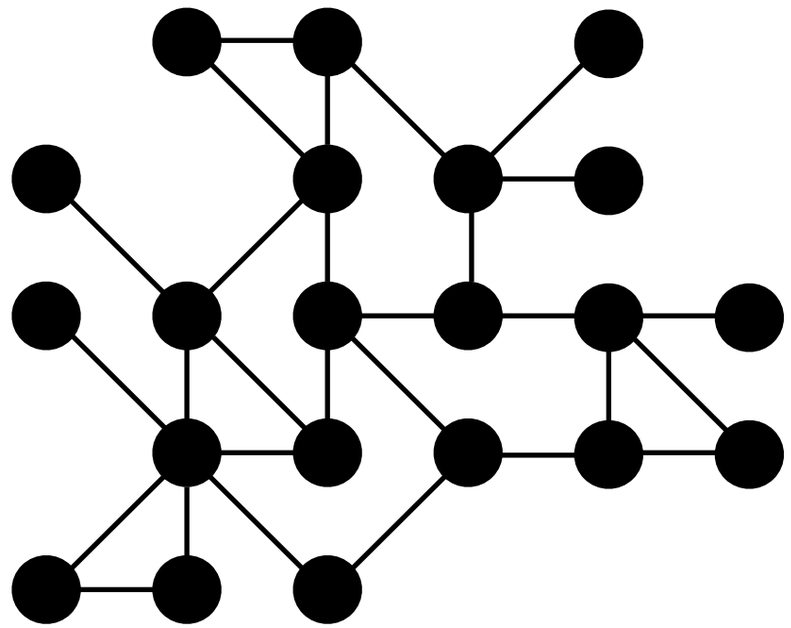
   - Vertices represent datasets.



Differential Privacy via Graphs

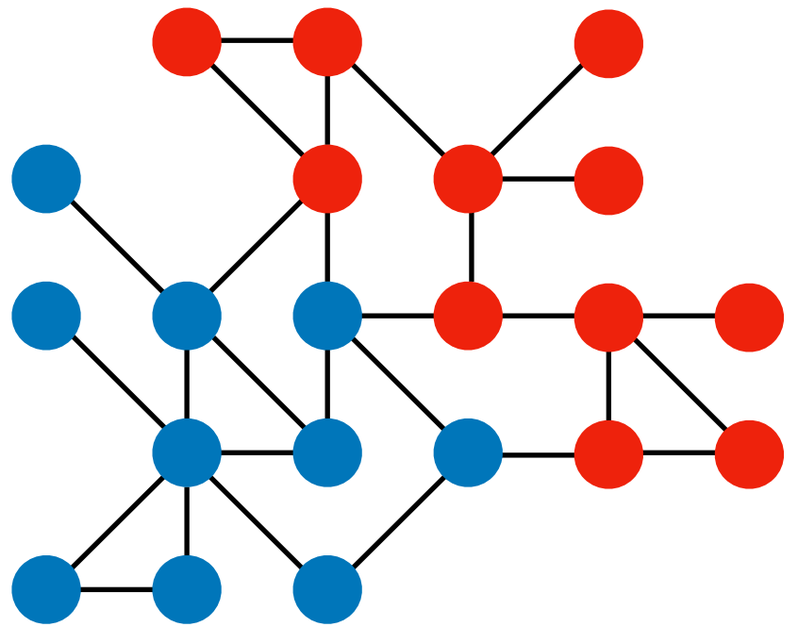
I. In [D'Oliveira et al. 21'], differential privacy was presented in a graph theoretical framework, where:

   - Vertices represent datasets.
   - Edges connect neighboring datasets.



Differential Privacy via Graphs

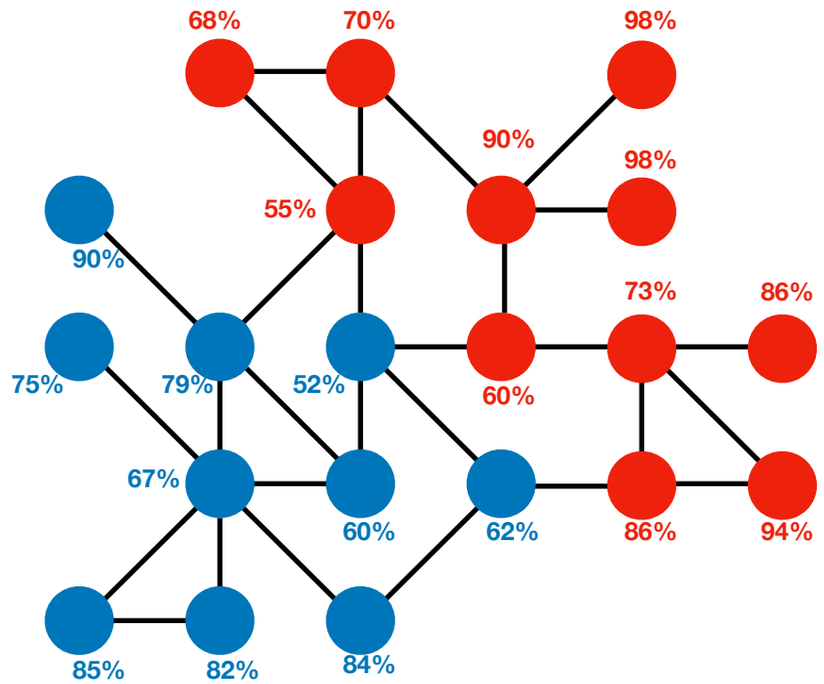
I. In [D'Oliveira et al. 21'], differential privacy was presented in a graph theoretical framework, where:
   - Vertices represent datasets.
   - Edges connect neighboring datasets.
   - Colors represent the output of true function.



Differential Privacy via Graphs

I. In [D'Oliveira et al. 21'], differential privacy was presented in a graph theoretical framework, where:

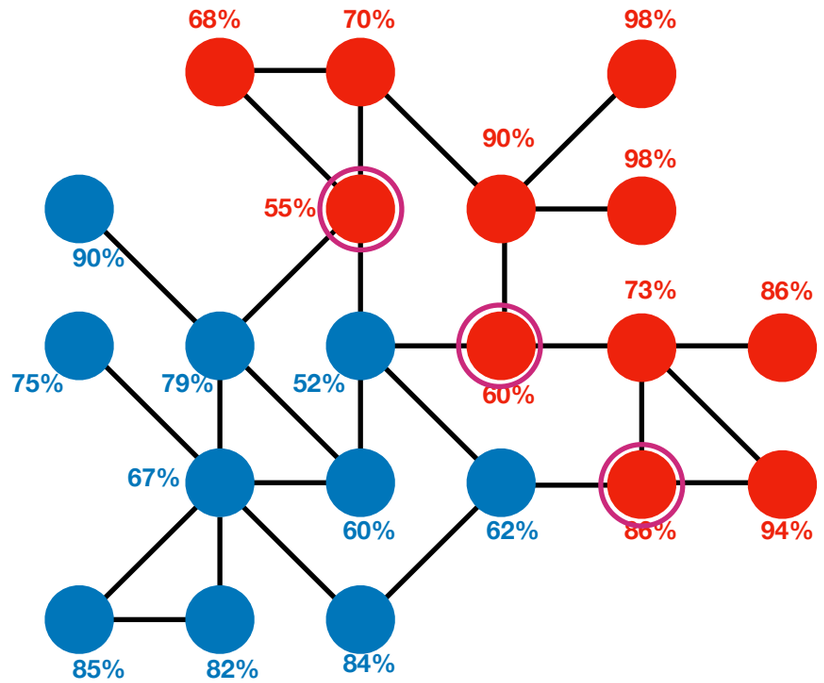
- Vertices represent datasets.
- Edges connect neighboring datasets.
- Colors represent the output of true function.
- A mechanism is a randomized coloring.



Differential Privacy via Graphs

I. In [D'Oliveira et al. 21'], differential privacy was presented in a graph theoretical framework, where:
   - Vertices represent datasets.
   - Edges connect neighboring datasets.
   - Colors represent the output of true function.
   - A mechanism is a randomized coloring.

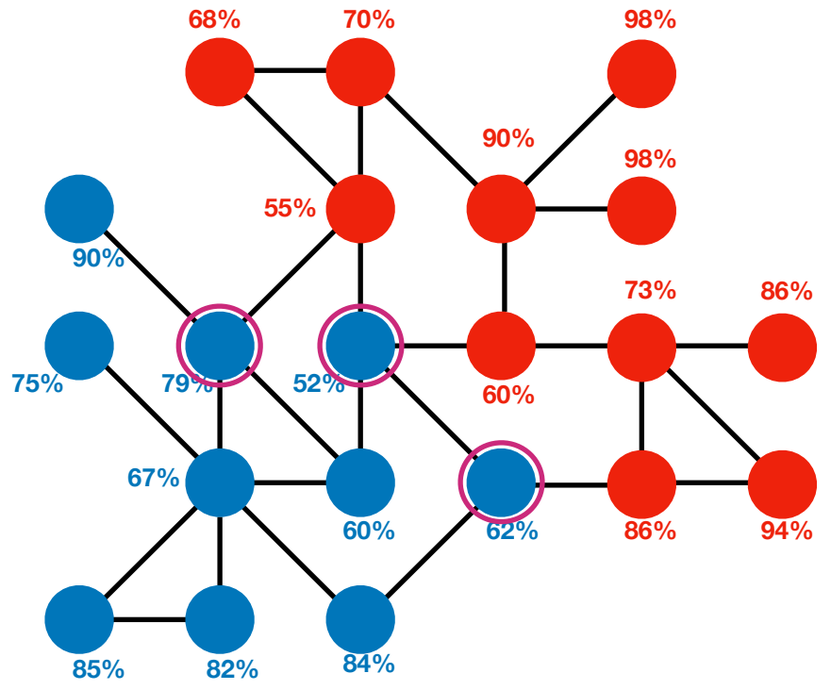
II. The optimal mechanism can be characterized in terms of its value at the boundary.



Differential Privacy via Graphs

I. In [D'Oliveira et al. 21'], differential privacy was presented in a graph theoretical framework, where:
   - Vertices represent datasets.
   - Edges connect neighboring datasets.
   - Colors represent the output of true function.
   - A mechanism is a randomized coloring.

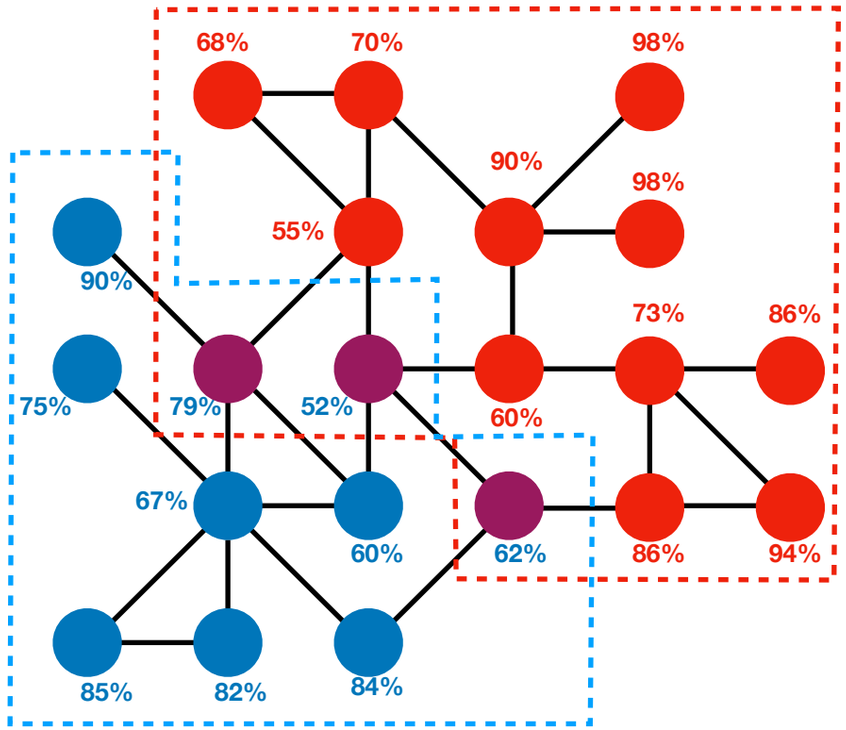
II. The optimal mechanism can be characterized in terms of its value at the boundary.



Differential Privacy via Graphs

I. In [D'Oliveira et al. 21'], differential privacy was presented in a graph theoretical framework, where:
   - Vertices represent datasets.
   - Edges connect neighboring datasets.
   - Colors represent the output of true function.
   - A mechanism is a randomized coloring.

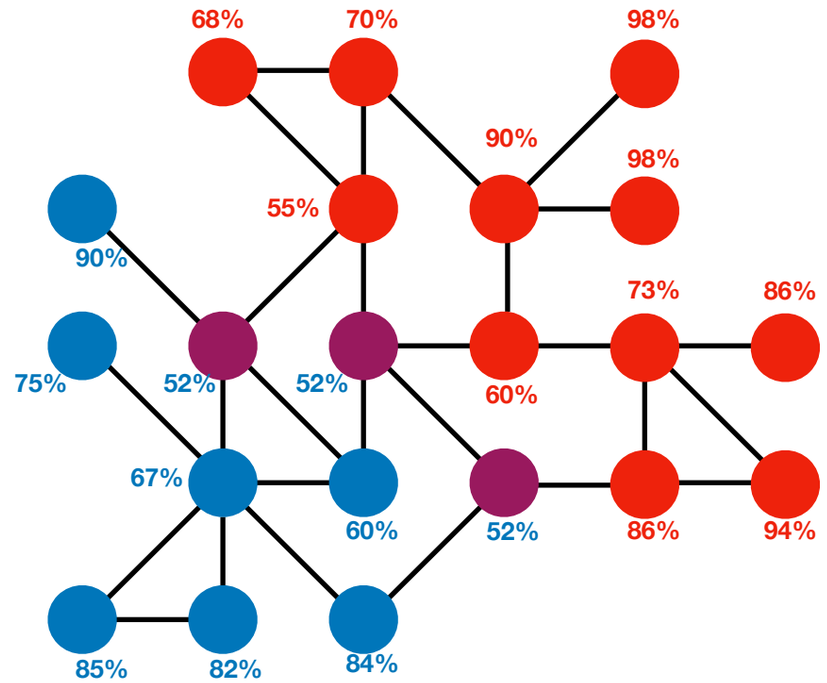
II. The optimal mechanism can be characterized in terms of its value at the boundary.



Differential Privacy via Graphs

I. In [D'Oliveira et al. 21'], differential privacy was presented in a graph theoretical framework, where:
   - Vertices represent datasets.
   - Edges connect neighboring datasets.
   - Colors represent the output of true function.
   - A mechanism is a randomized coloring.

II. The optimal mechanism can be characterized in terms of its value at the boundary.



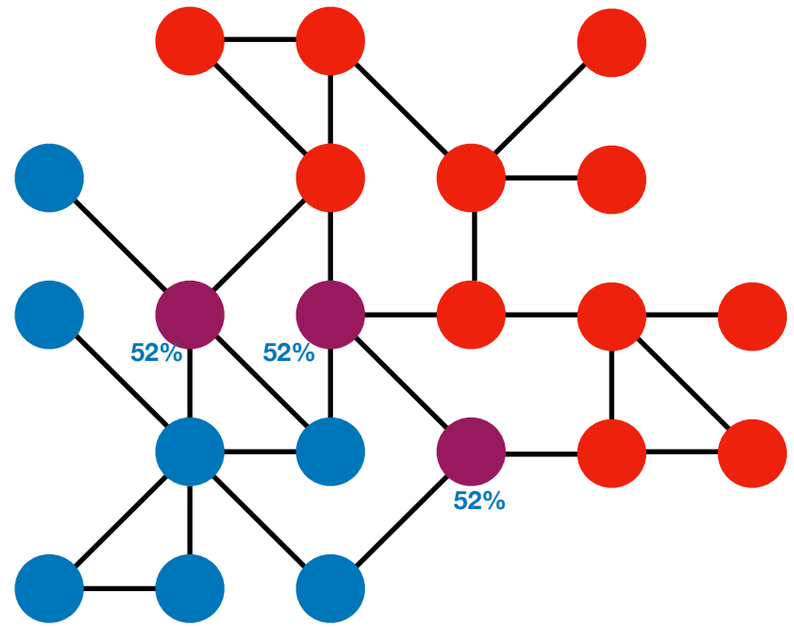
Definition: Boundary Homogeneous

A mechanism $\mathscr{M}$ is boundary homogeneous if the probabilities at the boundary are the same.

Differential Privacy via Graphs

I. In [D'Oliveira et al. 21'], differential privacy was presented in a graph theoretical framework, where:
   - Vertices represent datasets.
   - Edges connect neighboring datasets.
   - Colors represent the output of true function.
   - A mechanism is a randomized coloring.

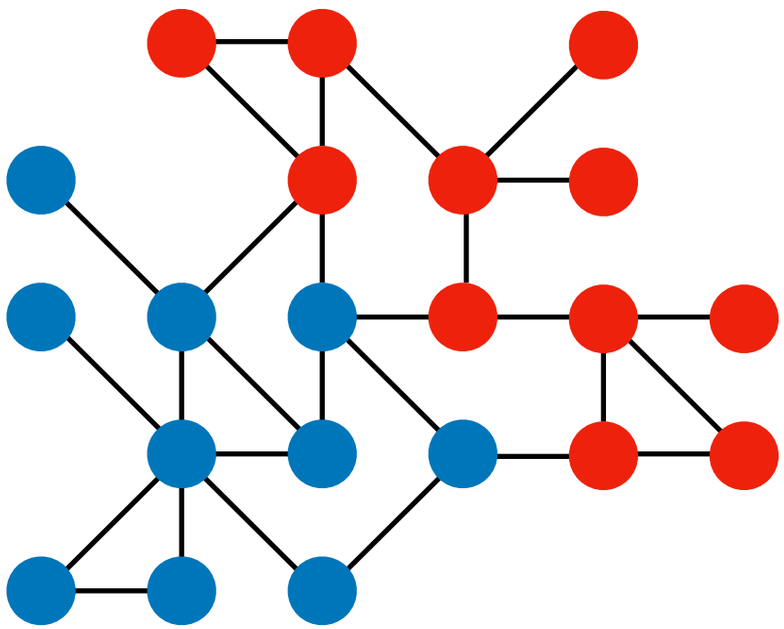
II. The optimal mechanism can be characterized in terms of its value at the boundary.



Theorem [D'Oliveira et al. 21']

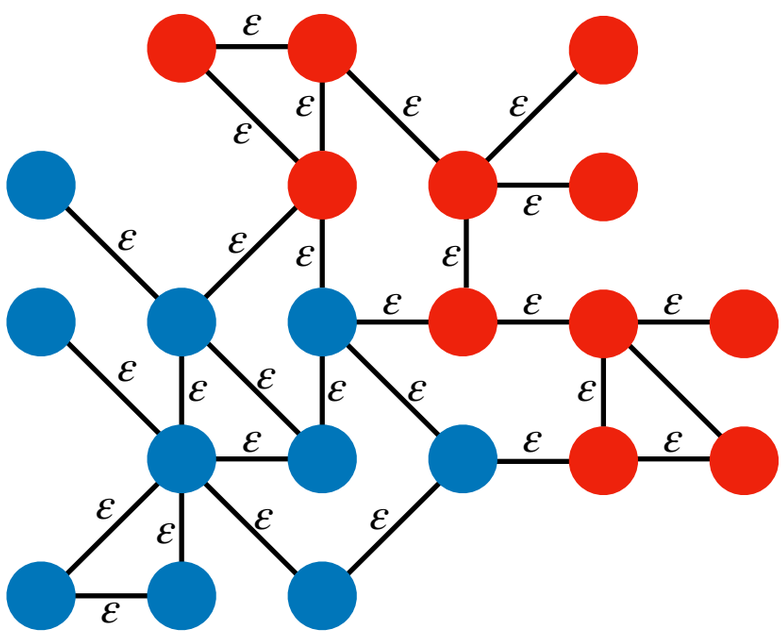
Let the boundary probabilities be equal. Then, there exists at most one optimal ε-DP mechanism. (A closed form is presented in the paper.)

Homogeneous Differential Privacy



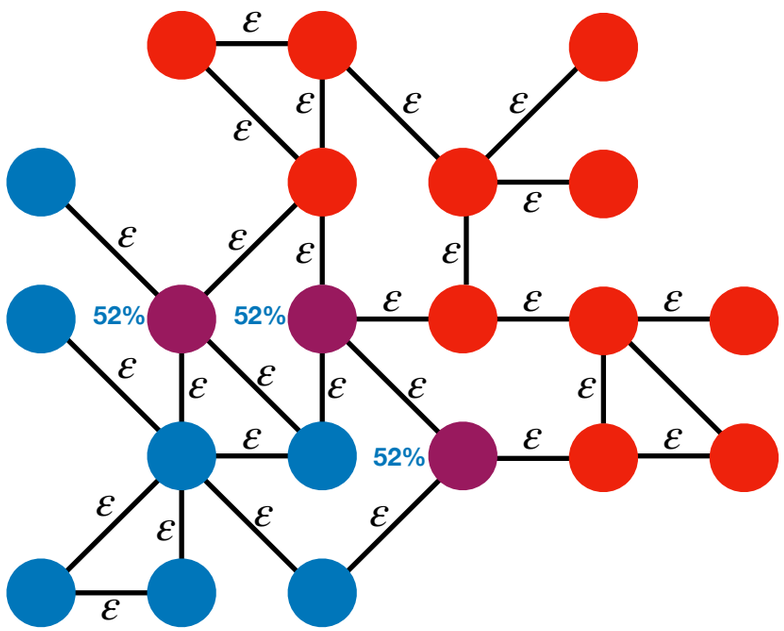
Homogeneous Differential Privacy

- Same level of privacy.



Homogeneous Differential Privacy

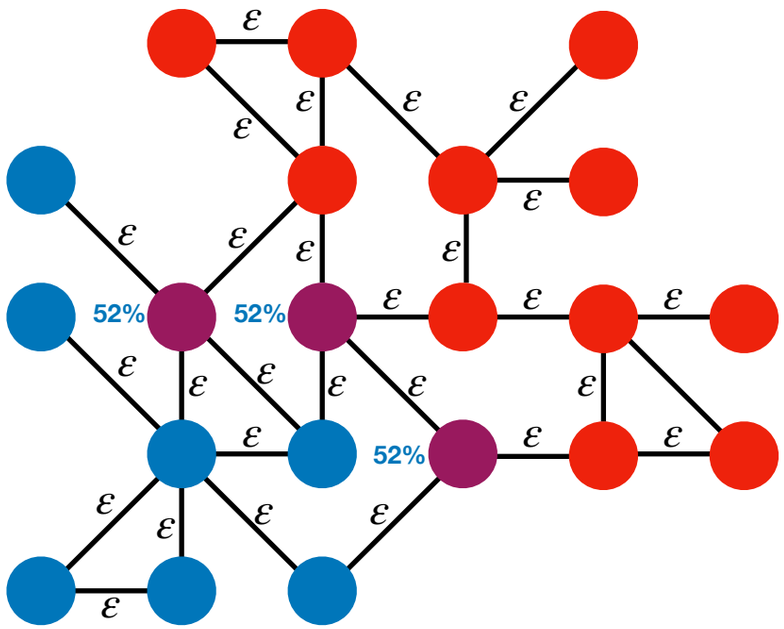
- Same level of privacy.
- Homogeneous boundary.

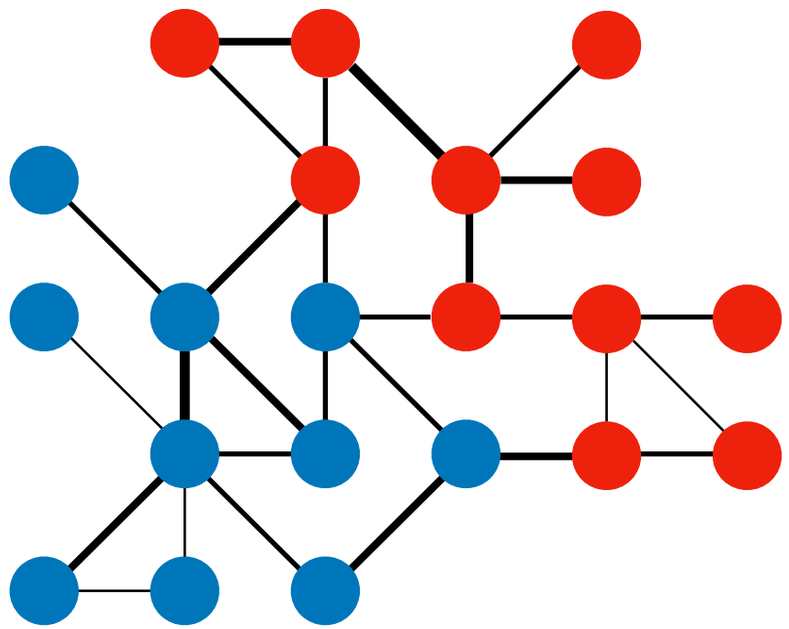


Homogeneous Differential Privacy

- Same level of privacy.
- Homogeneous boundary.

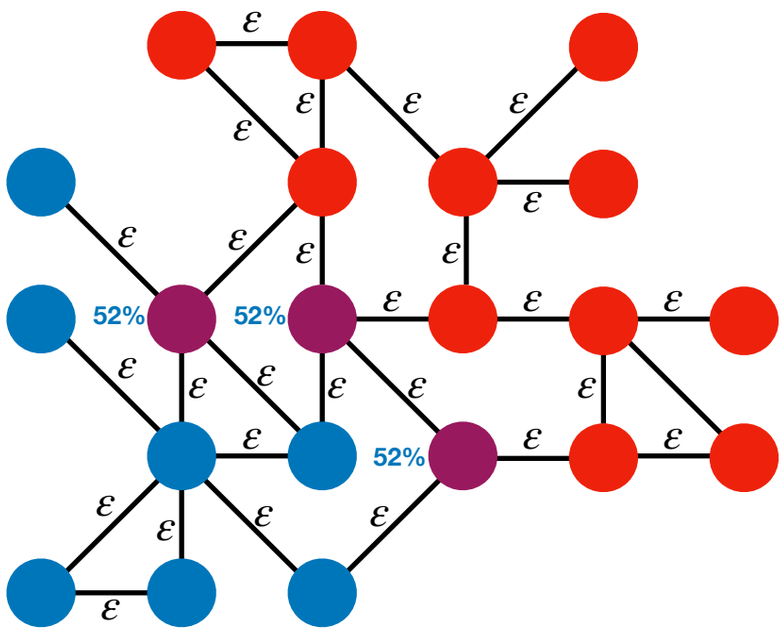
Heterogeneous Differential Privacy





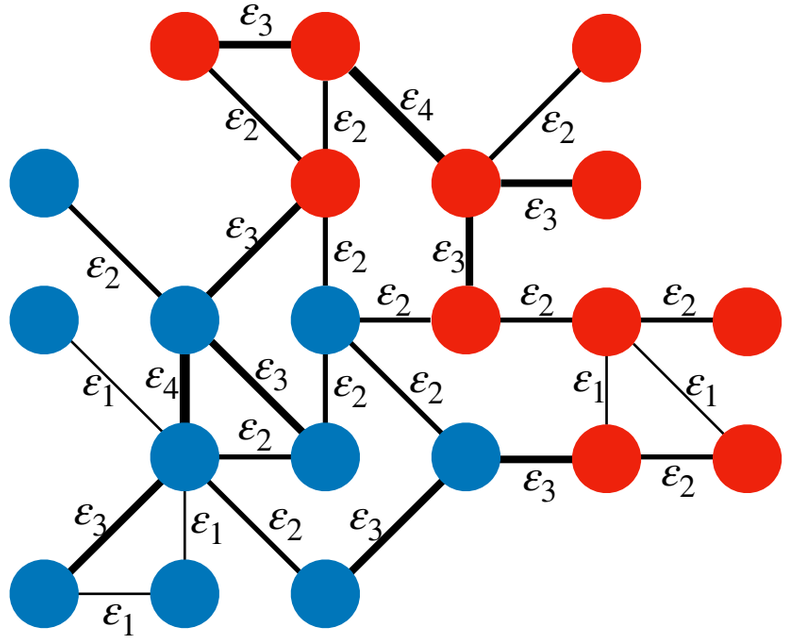
Homogeneous Differential Privacy

- Same level of privacy.
- Homogeneous boundary.



Heterogeneous Differential Privacy

- Different privacy levels.

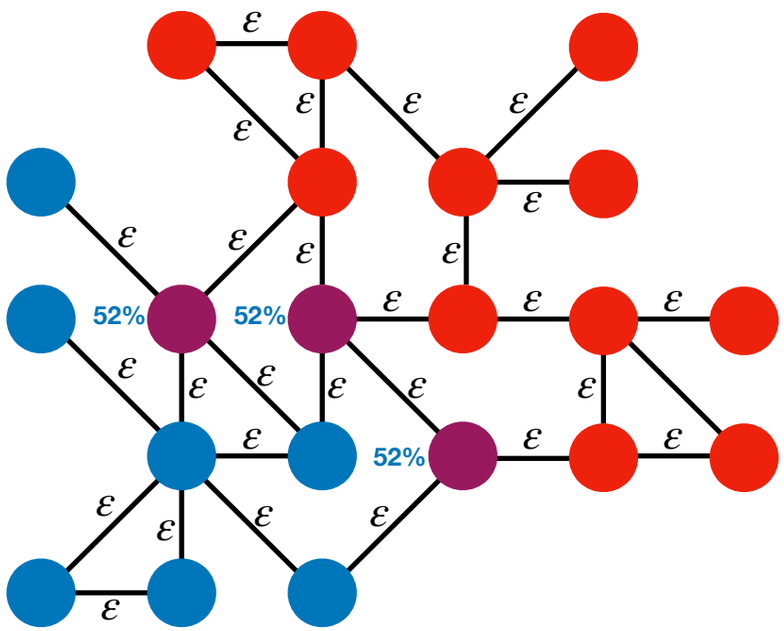


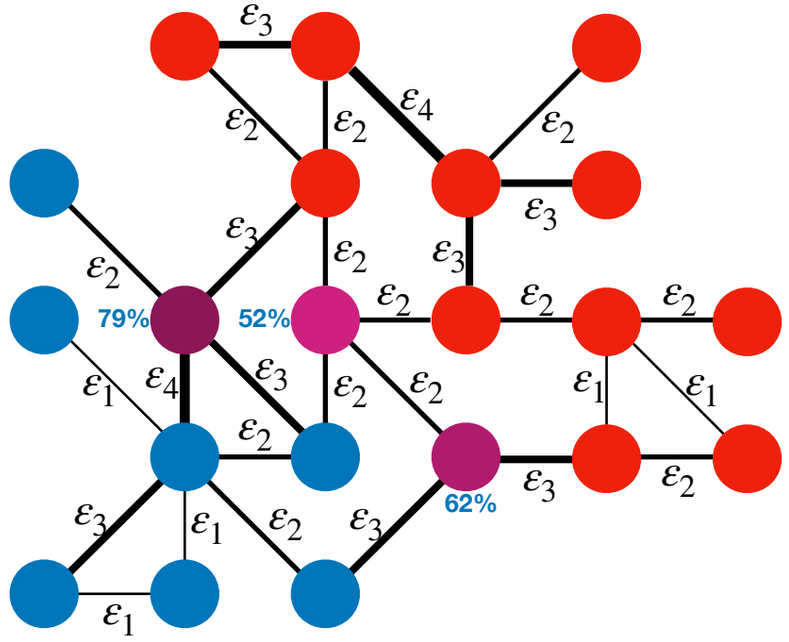
Homogeneous Differential Privacy

- Same level of privacy.
- Homogeneous boundary.

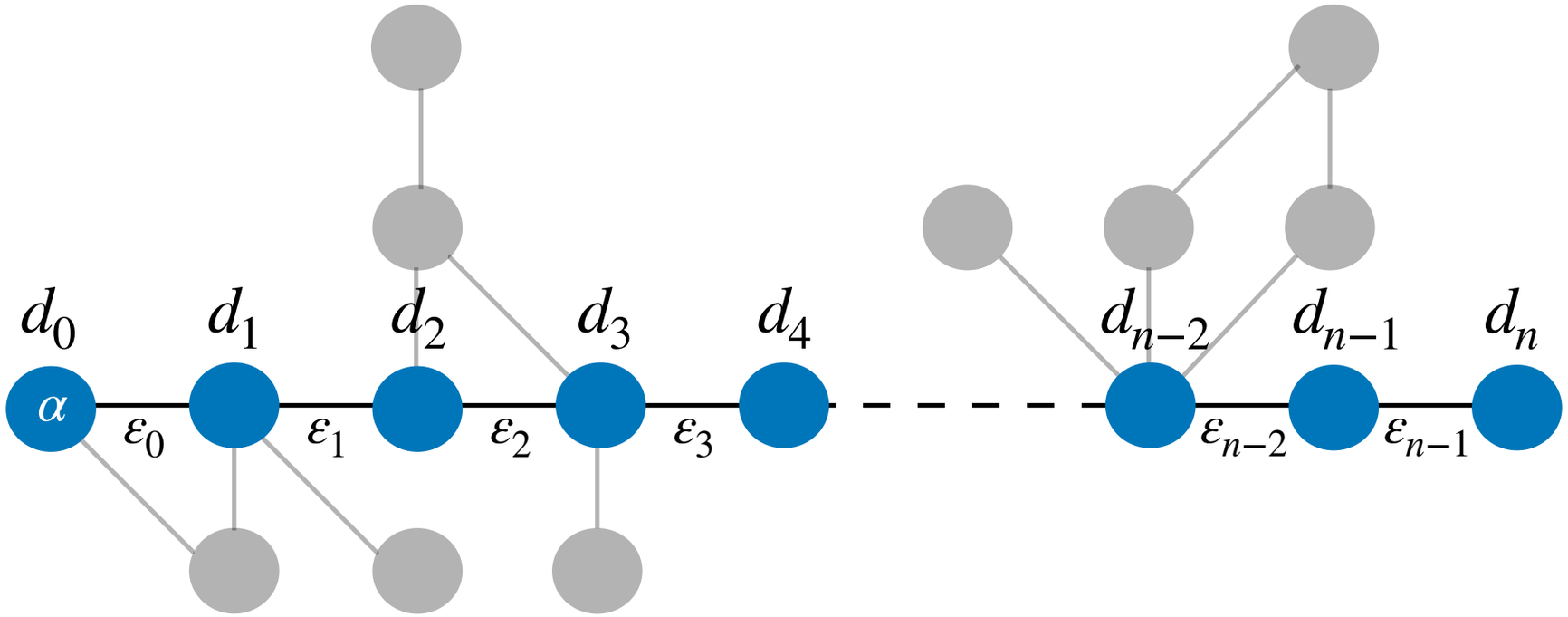
Heterogeneous Differential Privacy

- Different privacy levels.
- Different probability distributions at the boundary

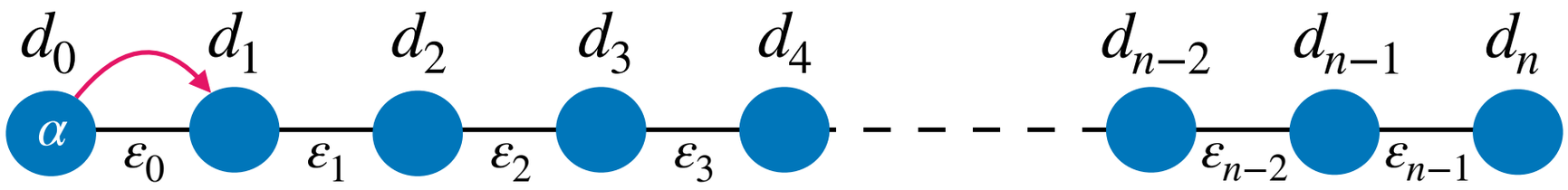




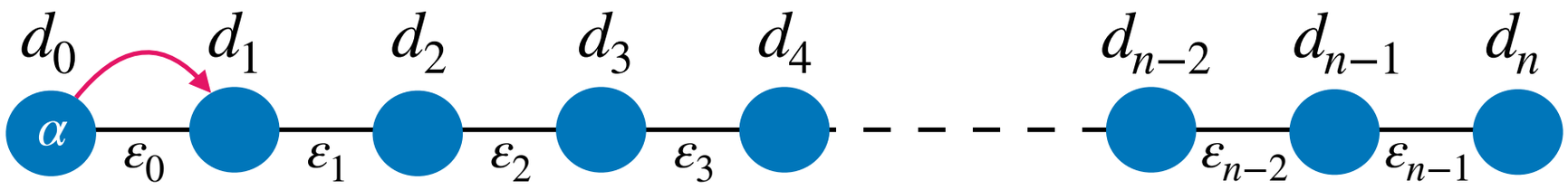
Special Case: Path Graph



Special Case: Path Graph



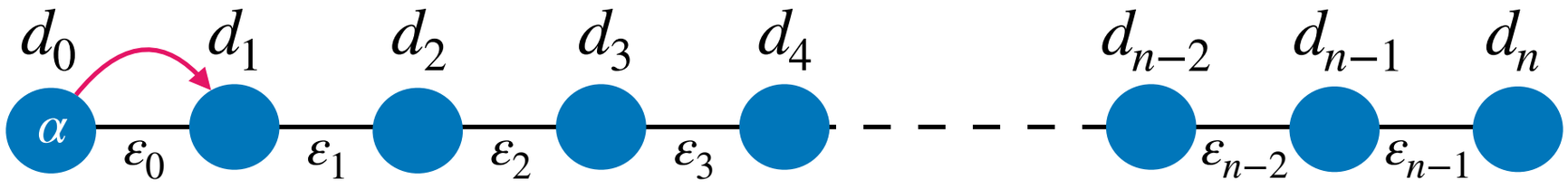
Special Case: Path Graph



Differential Privacy conditions between d_0 and d_1:

- $\Pr[\mathscr{M}(d_1) = 1] \leq e^{\varepsilon} \Pr[\mathscr{M}(d_0) = 1]$
- $(1 - \Pr[\mathscr{M}(d_0) = 1]) \leq e^{\varepsilon}(1 - \Pr[\mathscr{M}(d_1) = 1])$
- $\Pr[\mathscr{M}(d_0) = 1] \leq e^{\varepsilon} \Pr[\mathscr{M}(d_1) = 1]$
- $(1 - \Pr[\mathscr{M}(d_1) = 1]) \leq e^{\varepsilon}(1 - \Pr[\mathscr{M}(d_0) = 1])$

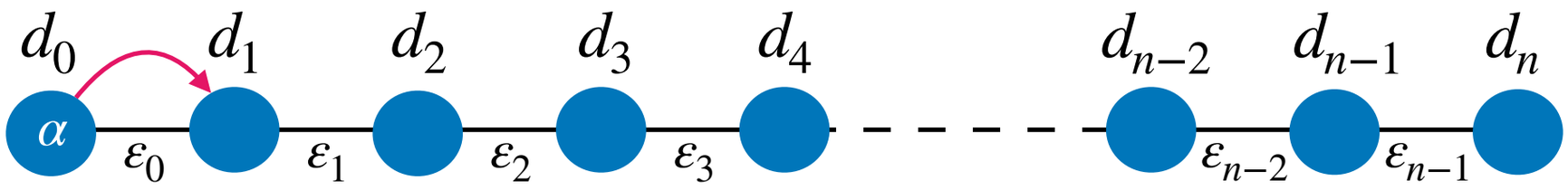
Special Case: Path Graph



Differential Privacy conditions between d_0 and d_1:

- $\Pr[\mathscr{M}(d_1) = 1] \leq e^{\varepsilon} \Pr[\mathscr{M}(d_0) = 1]$
- $(1 - \Pr[\mathscr{M}(d_0) = 1]) \leq e^{\varepsilon}(1 - \Pr[\mathscr{M}(d_1) = 1])$
- $\Pr[\mathscr{M}(d_0) = 1] \leq e^{\varepsilon} \Pr[\mathscr{M}(d_1) = 1]$
- $(1 - \Pr[\mathscr{M}(d_1) = 1]) \leq e^{\varepsilon}(1 - \Pr[\mathscr{M}(d_0) = 1])$

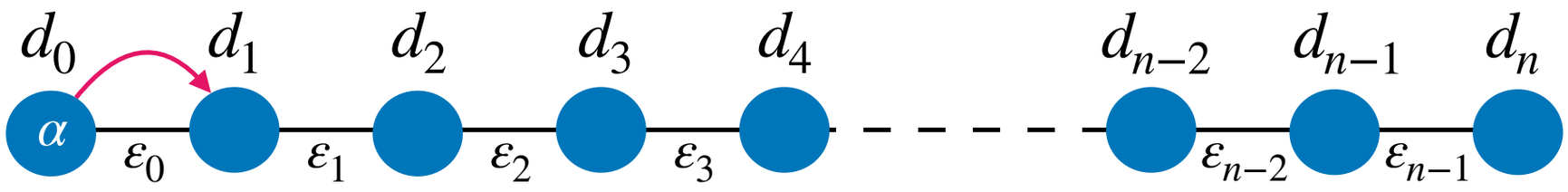
Special Case: Path Graph



Upper bound on d_1 imposed by d_0:

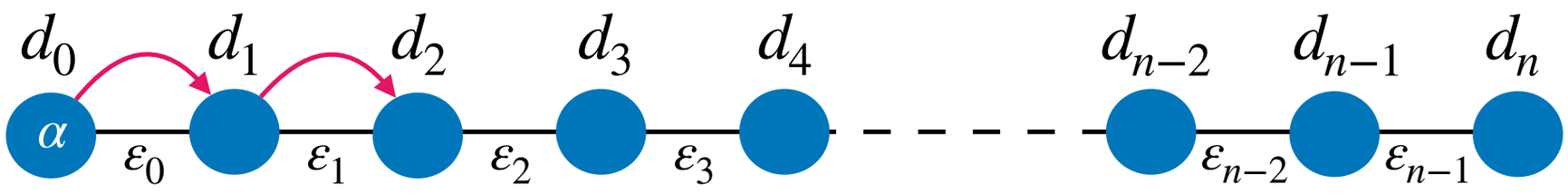
$$\Pr[\mathscr{M}(d_1) = 1] \leq \min\left(e^{\varepsilon}\Pr[\mathscr{M}(d_0) = 1], \frac{e^{\varepsilon} - 1 + \Pr[\mathscr{M}(d_0) = 1]}{e^{\varepsilon}}\right)$$

Special Case: Path Graph



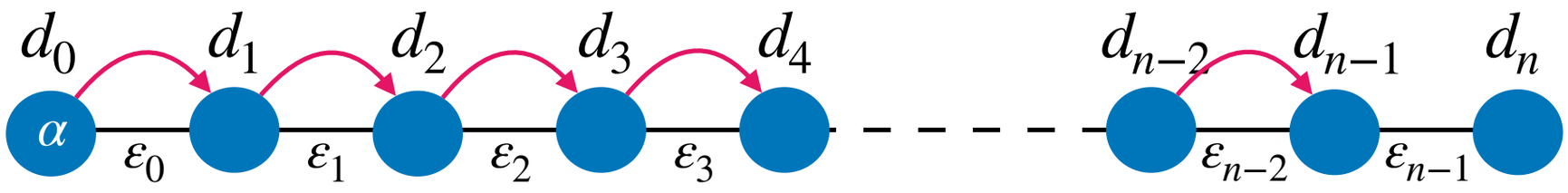
- $\Pr[\mathscr{M}(d_1) = 1] \leq \min\left(e^{\varepsilon}\Pr[\mathscr{M}(d_0) = 1], \dfrac{e^{\varepsilon} - 1 + \Pr[\mathscr{M}(d_0) = 1]}{e^{\varepsilon}}\right)$

Special Case: Path Graph



- $\Pr[\mathscr{M}(d_1) = 1] \leq \min\left(e^{\varepsilon}\Pr[\mathscr{M}(d_0) = 1], \dfrac{e^{\varepsilon} - 1 + \Pr[\mathscr{M}(d_0) = 1]}{e^{\varepsilon}}\right)$
- $\Pr[\mathscr{M}(d_2) = 1] \leq \min\left(e^{\varepsilon}\Pr[\mathscr{M}(d_1) = 1], \dfrac{e^{\varepsilon} - 1 + \Pr[\mathscr{M}(d_1) = 1]}{e^{\varepsilon}}\right)$

Special Case: Path Graph

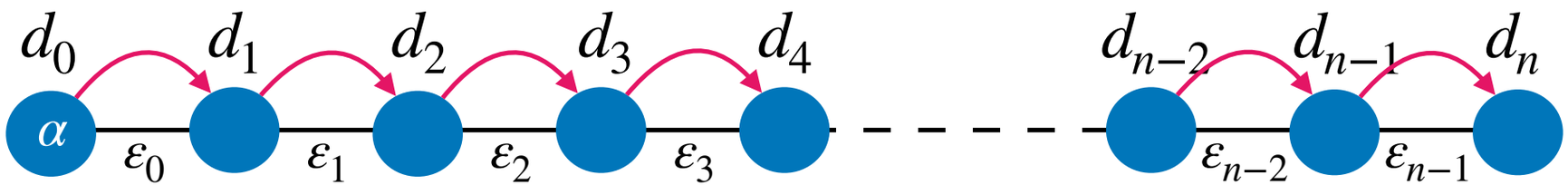


- $\Pr[\mathscr{M}(d_1) = 1] \leq \min\left(e^{\varepsilon}\Pr[\mathscr{M}(d_0) = 1], \dfrac{e^{\varepsilon} - 1 + \Pr[\mathscr{M}(d_0) = 1]}{e^{\varepsilon}}\right)$
- $\Pr[\mathscr{M}(d_2) = 1] \leq \min\left(e^{\varepsilon}\Pr[\mathscr{M}(d_1) = 1], \dfrac{e^{\varepsilon} - 1 + \Pr[\mathscr{M}(d_1) = 1]}{e^{\varepsilon}}\right)$

$\vdots$

- $\Pr[\mathscr{M}(d_{n-1}) = 1] \leq \min\left(e^{\varepsilon}\Pr[\mathscr{M}(d_{n-2}) = 1], \dfrac{e^{\varepsilon} - 1 + \Pr[\mathscr{M}(d_{n-2}) = 1]}{e^{\varepsilon}}\right)$

Special Case: Path Graph

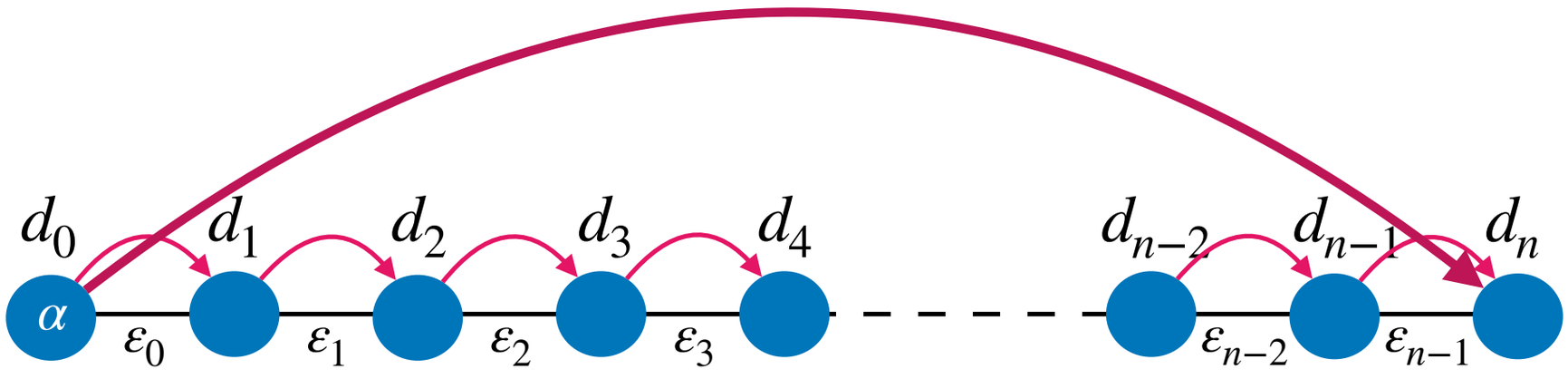


- $\Pr[\mathscr{M}(d_1) = 1] \leq \min\left(e^{\varepsilon}\Pr[\mathscr{M}(d_0) = 1], \dfrac{e^{\varepsilon} - 1 + \Pr[\mathscr{M}(d_0) = 1]}{e^{\varepsilon}}\right)$
- $\Pr[\mathscr{M}(d_2) = 1] \leq \min\left(e^{\varepsilon}\Pr[\mathscr{M}(d_1) = 1], \dfrac{e^{\varepsilon} - 1 + \Pr[\mathscr{M}(d_1) = 1]}{e^{\varepsilon}}\right)$

$\vdots$

- $\Pr[\mathscr{M}(d_{n-1}) = 1] \leq \min\left(e^{\varepsilon}\Pr[\mathscr{M}(d_{n-2}) = 1], \dfrac{e^{\varepsilon} - 1 + \Pr[\mathscr{M}(d_{n-2}) = 1]}{e^{\varepsilon}}\right)$
- $\Pr[\mathscr{M}(d_n) = 1] \leq \min\left(e^{\varepsilon}\Pr[\mathscr{M}(d_{n-1}) = 1], \dfrac{e^{\varepsilon} - 1 + \Pr[\mathscr{M}(d_{n-1}) = 1]}{e^{\varepsilon}}\right)$

Special Case: Path Graph

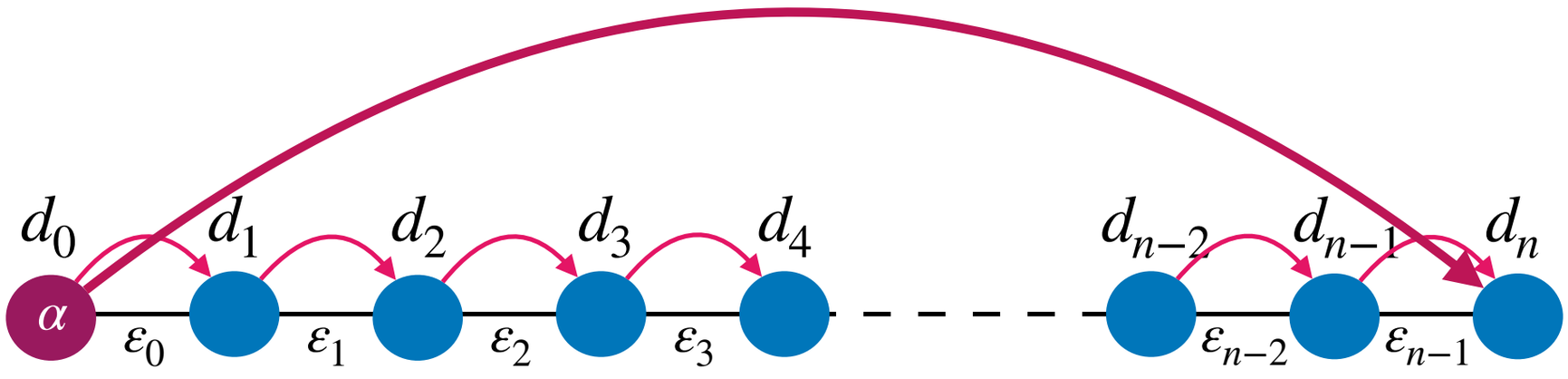


- $\Pr[\mathscr{M}(d_1) = 1] \leq \min\left(e^{\varepsilon}\Pr[\mathscr{M}(d_0) = 1], \dfrac{e^{\varepsilon} - 1 + \Pr[\mathscr{M}(d_0) = 1]}{e^{\varepsilon}}\right)$
- $\Pr[\mathscr{M}(d_2) = 1] \leq \min\left(e^{\varepsilon}\Pr[\mathscr{M}(d_1) = 1], \dfrac{e^{\varepsilon} - 1 + \Pr[\mathscr{M}(d_1) = 1]}{e^{\varepsilon}}\right)$

$\vdots$

- $\Pr[\mathscr{M}(d_{n-1}) = 1] \leq \min\left(e^{\varepsilon}\Pr[\mathscr{M}(d_{n-2}) = 1], \dfrac{e^{\varepsilon} - 1 + \Pr[\mathscr{M}(d_{n-2}) = 1]}{e^{\varepsilon}}\right)$
- $\Pr[\mathscr{M}(d_n) = 1] \leq \min\left(e^{\varepsilon}\Pr[\mathscr{M}(d_{n-1}) = 1], \dfrac{e^{\varepsilon} - 1 + \Pr[\mathscr{M}(d_{n-1}) = 1]}{e^{\varepsilon}}\right)$

Special Case: Path Graph



Lemma: Differential Privacy for the Path Graph

The optimal binary-valued heterogeneous differentially private mechanism $\mathscr{M}^*$ is given by

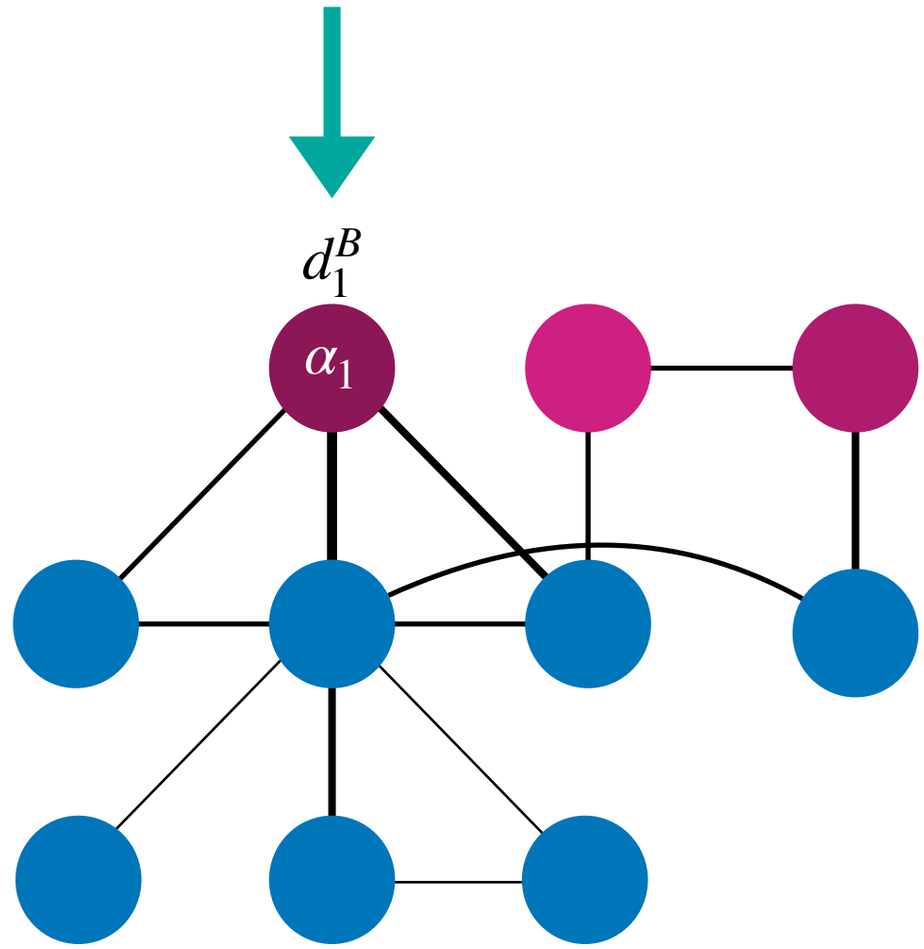
$$\Pr[\mathscr{M}^*(v_i) = 1] = \begin{cases} e^{\varepsilon_{i-1}+\ldots+\varepsilon_1+\varepsilon_0}\,\alpha & i \le \tau \\ e^{-\varepsilon_{i-1}-\ldots-\varepsilon_1-\varepsilon_\tau+\varepsilon_{\tau-1}+\ldots+\varepsilon_1+\varepsilon_0}\,\alpha + 1 - e^{-\varepsilon_{i-1}-\ldots-\varepsilon_1-\varepsilon_\tau} & i > \tau \end{cases}$$

where

$$\tau = \arg\min_{i\in[n]} \{\frac{1}{\alpha} \le e^{\varepsilon_{i-1}+\ldots+\varepsilon_1+\varepsilon_0}(e^{\varepsilon_i}+1)\}\,.$$

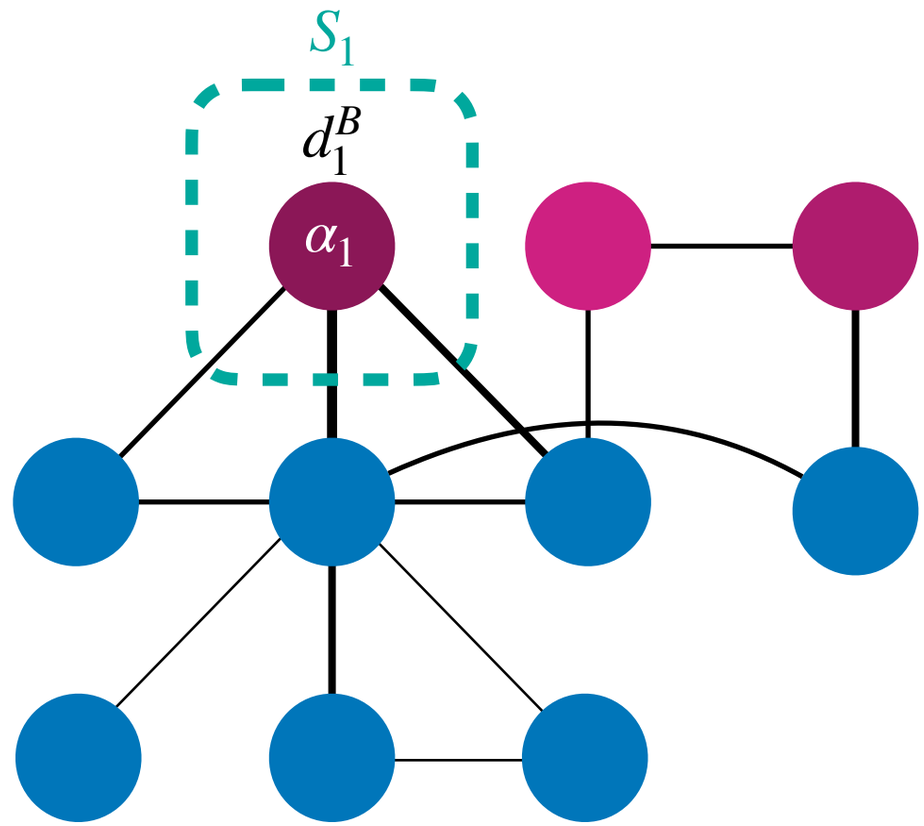
Algorithm

- For each boundary vertex d_i^B with a fixed value α_i, the upper bound on the other vertices imposed by d_i^B is calculated in the following steps.



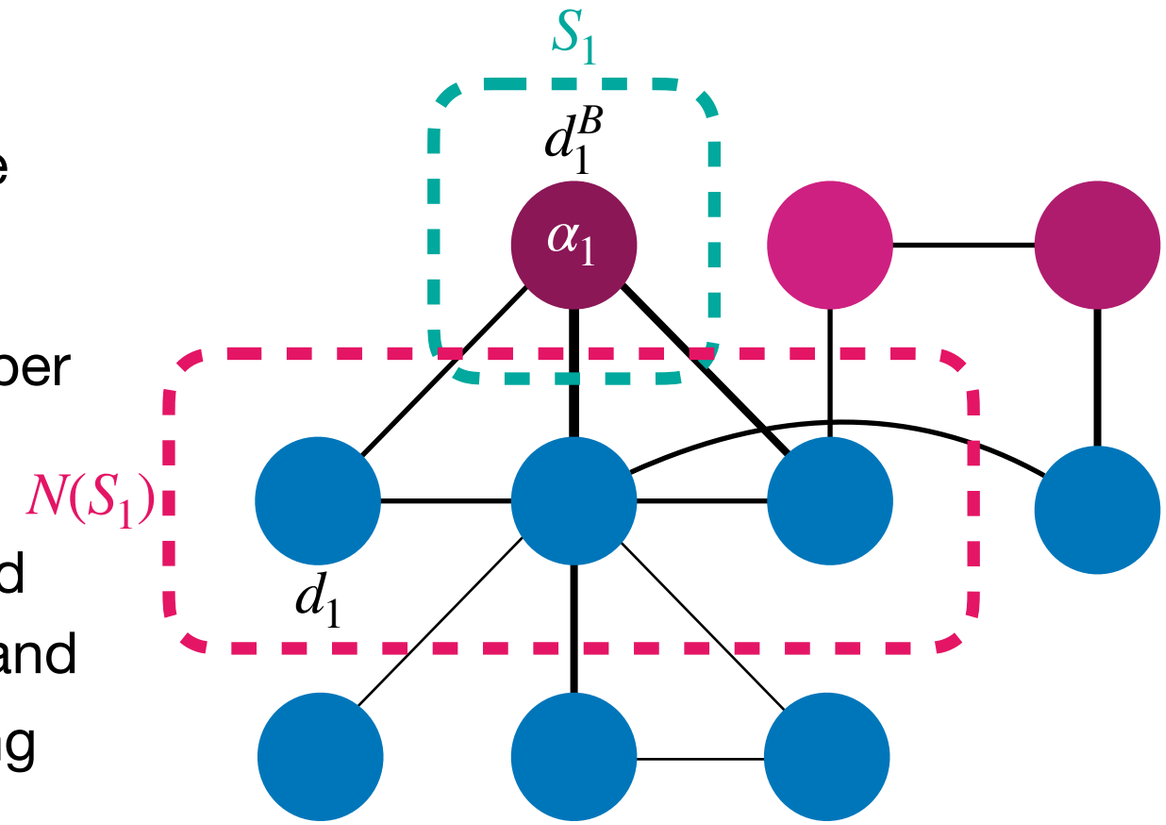
Algorithm

- For each boundary vertex d_i^B with a fixed value α_i, the upper bound on the other vertices imposed by d_i^B is calculated in the following steps.
- Let S_i be the set of vertices which their upper bound (imposed by d_i^B) is calculated.



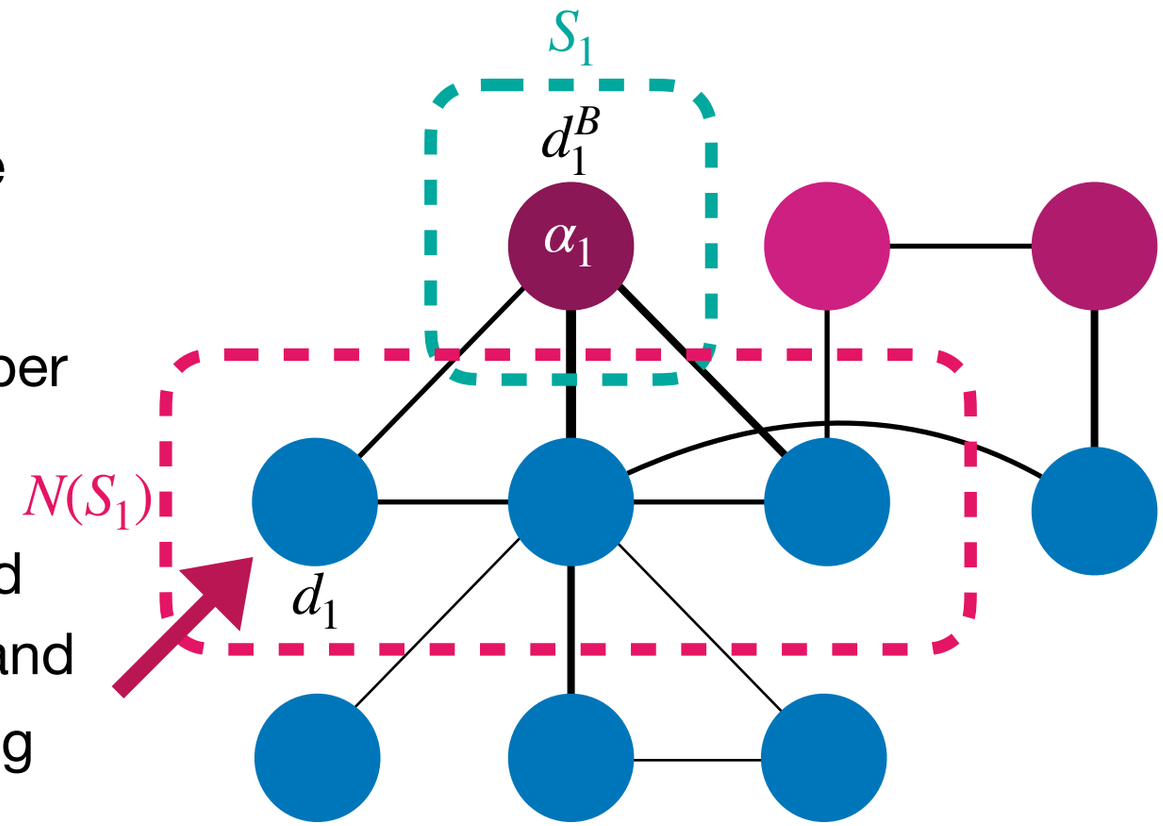
Algorithm

- For each boundary vertex d_i^B with a fixed value α_i, the upper bound on the other vertices imposed by d_i^B is calculated in the following steps.
- Let S_i be the set of vertices which their upper bound (imposed by d_i^B) is calculated.
- In each step, we find the least upper bound among the neighbors of S_i imposed by S_i and S_{i+1} is defined by adding the corresponding vertex to S_i.



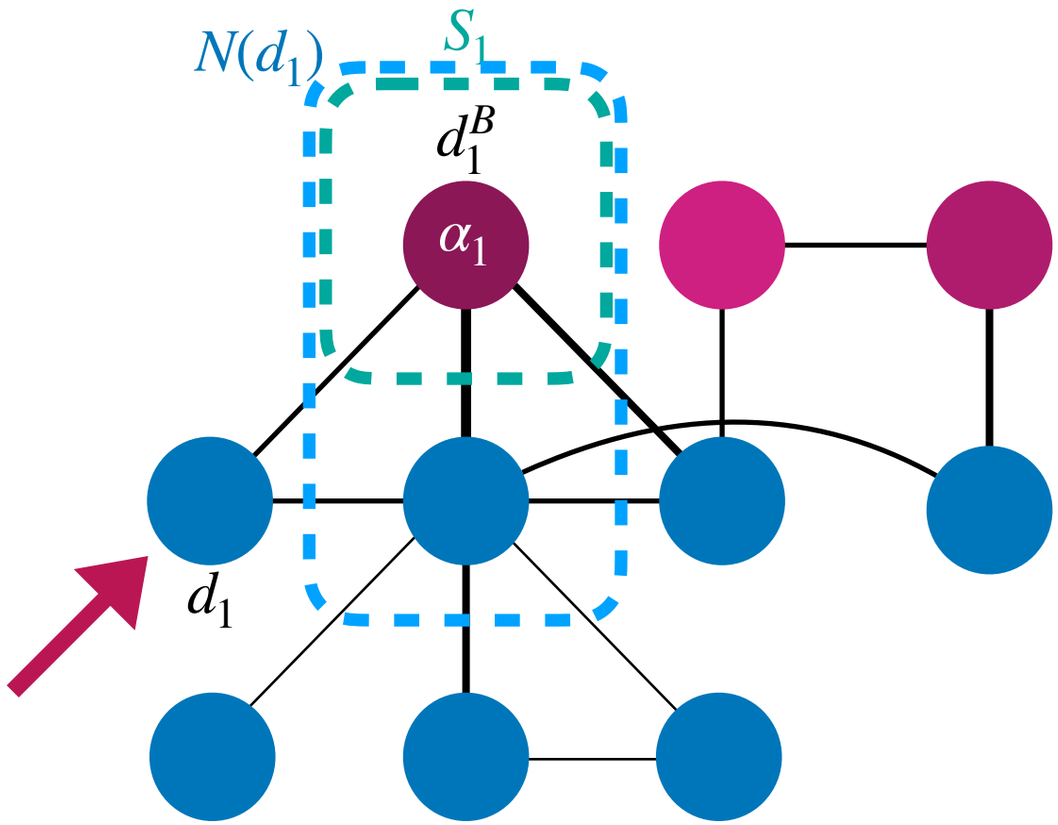
Algorithm

- For each boundary vertex d_i^B with a fixed value α_i, the upper bound on the other vertices imposed by d_i^B is calculated in the following steps.
- Let S_i be the set of vertices which their upper bound (imposed by d_i^B) is calculated.
- In each step, we find the least upper bound among the neighbors of S_i imposed by S_i and S_{i+1} is defined by adding the corresponding vertex to S_i.



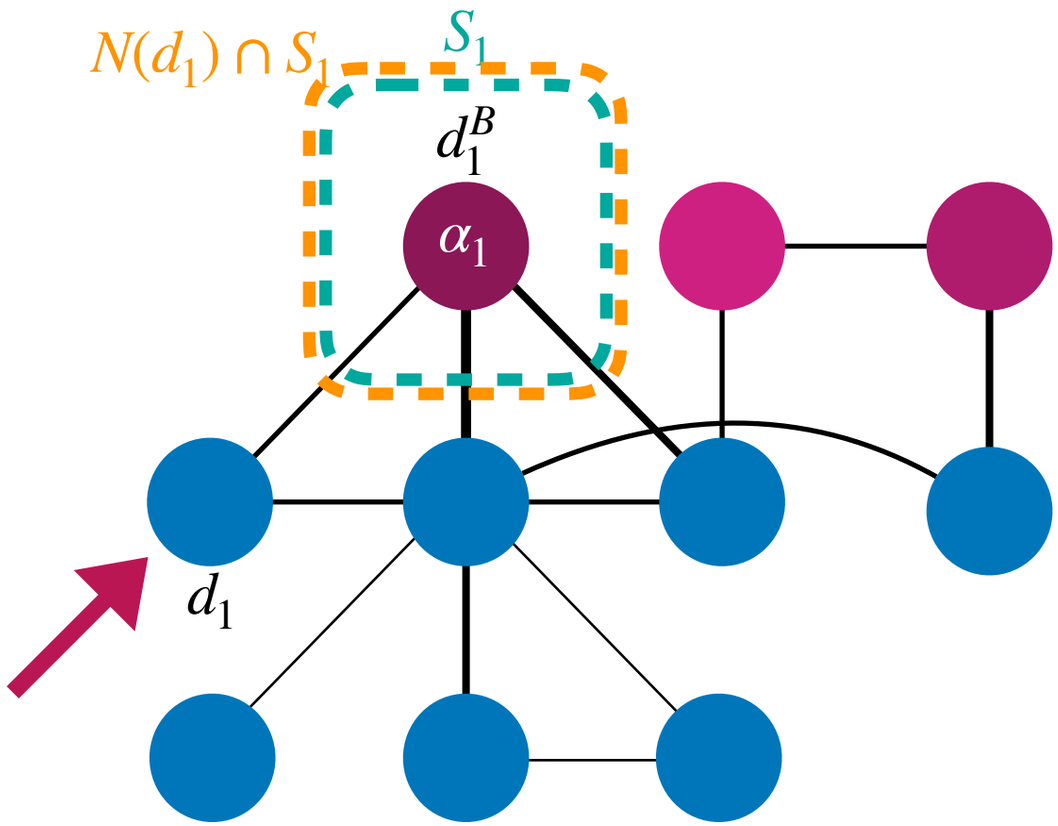
Algorithm

- For each boundary vertex d_i^B with a fixed value α_i, the upper bound on the other vertices imposed by d_i^B is calculated in the following steps.
- Let S_i be the set of vertices which their upper bound (imposed by d_i^B) is calculated.
- In each step, we find the least upper bound among the neighbors of S_i imposed by S_i and S_{i+1} is defined by adding the corresponding vertex to S_i.



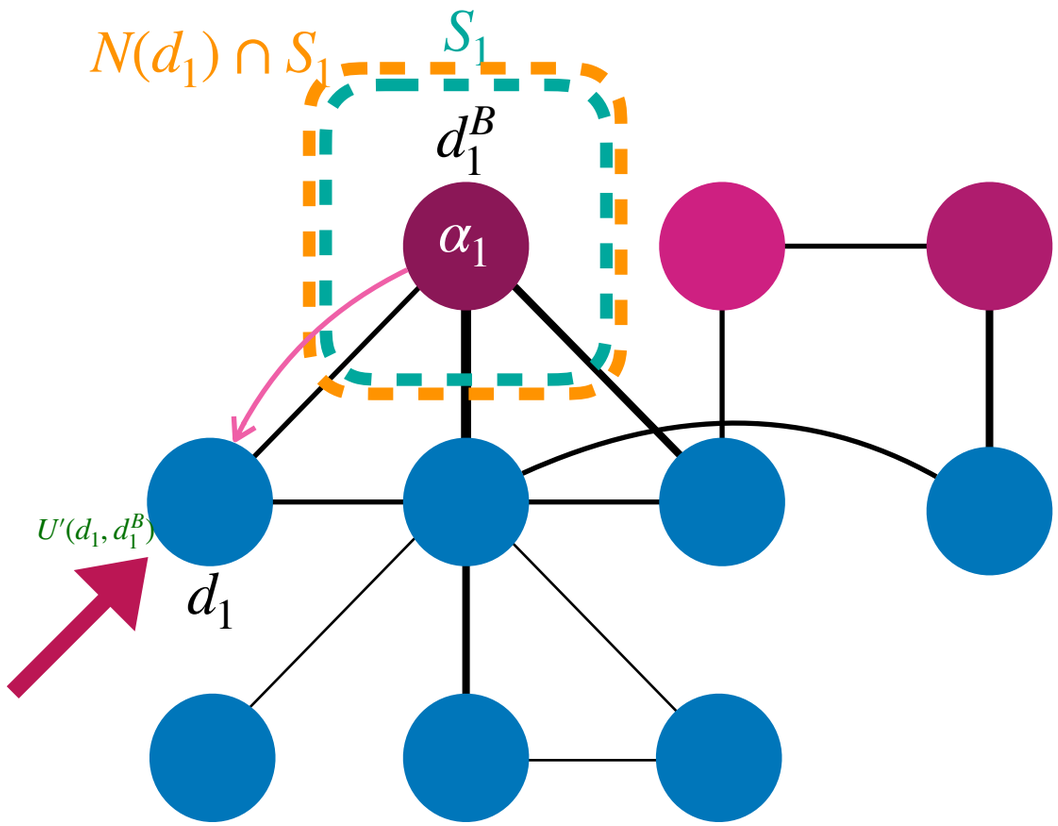
Algorithm

- For each boundary vertex d_i^B with a fixed value α_i, the upper bound on the other vertices imposed by d_i^B is calculated in the following steps.
- Let S_i be the set of vertices which their upper bound (imposed by d_i^B) is calculated.
- In each step, we find the least upper bound among the neighbors of S_i imposed by S_i and S_{i+1} is defined by adding the corresponding vertex to S_i.



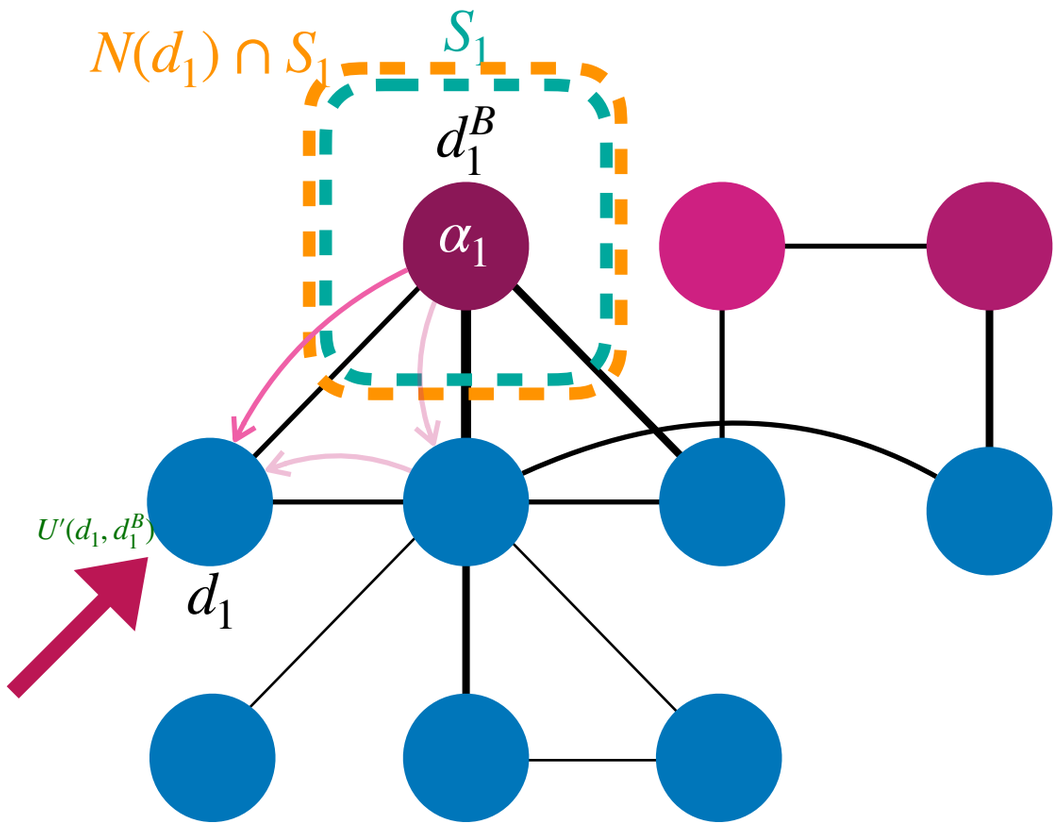
Algorithm

- For each boundary vertex d_i^B with a fixed value α_i, the upper bound on the other vertices imposed by d_i^B is calculated in the following steps.
- Let S_i be the set of vertices which their upper bound (imposed by d_i^B) is calculated.
- In each step, we find the least upper bound among the neighbors of S_i imposed by S_i and S_{i+1} is defined by adding the corresponding vertex to S_i.



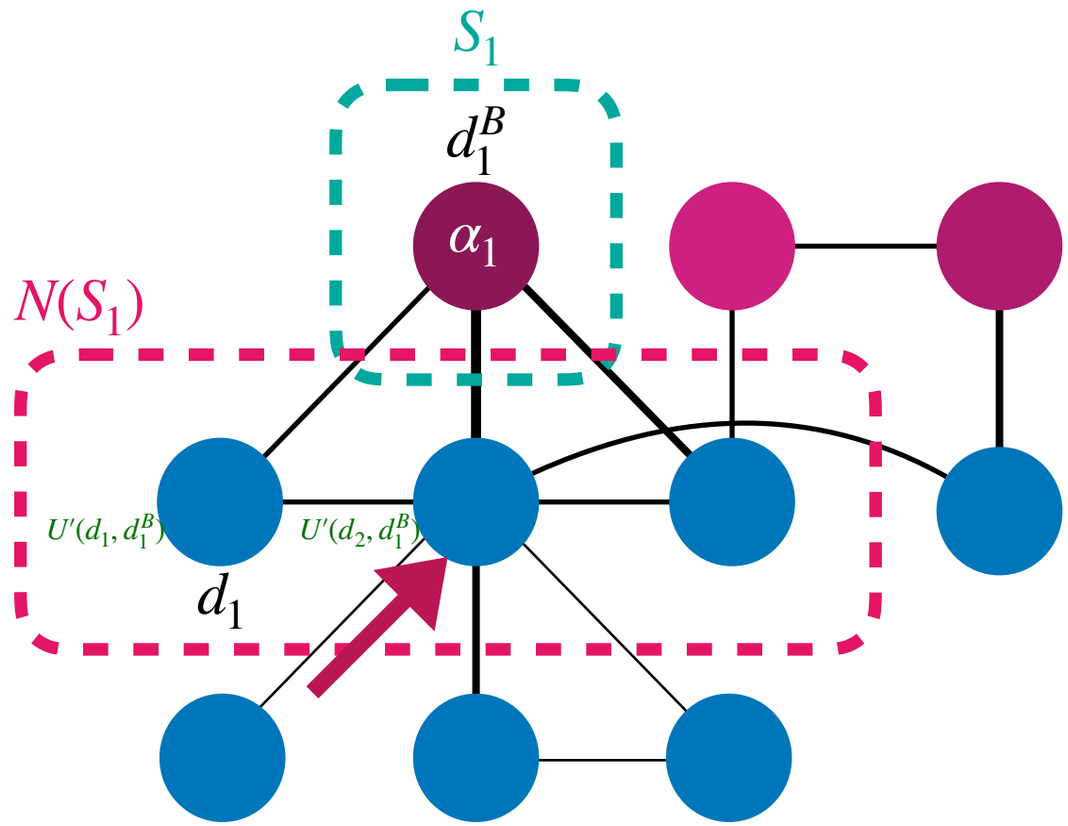
Algorithm

- For each boundary vertex d_i^B with a fixed value α_i, the upper bound on the other vertices imposed by d_i^B is calculated in the following steps.
- Let S_i be the set of vertices which their upper bound (imposed by d_i^B) is calculated.
- In each step, we find the least upper bound among the neighbors of S_i imposed by S_i and S_{i+1} is defined by adding the corresponding vertex to S_i.



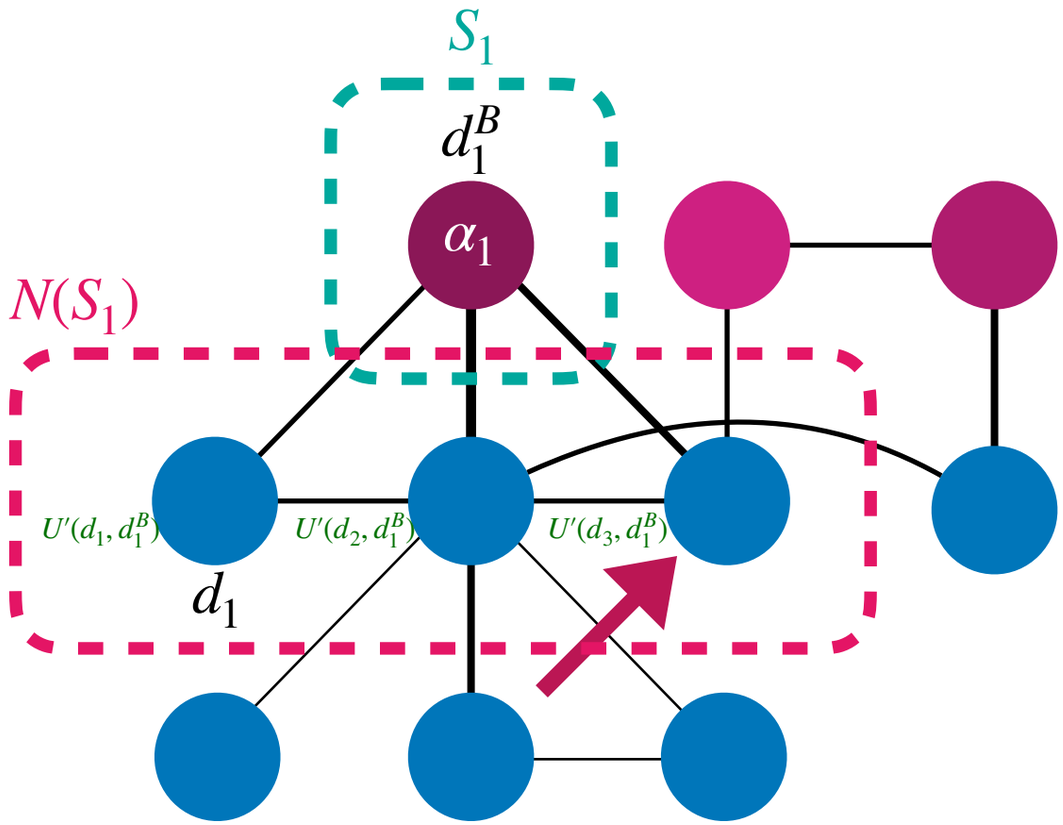
Algorithm

- For each boundary vertex d_i^B with a fixed value α_i, the upper bound on the other vertices imposed by d_i^B is calculated in the following steps.
- Let S_i be the set of vertices which their upper bound (imposed by d_i^B) is calculated.
- In each step, we find the least upper bound among the neighbors of S_i imposed by S_i and S_{i+1} is defined by adding the corresponding vertex to S_i.



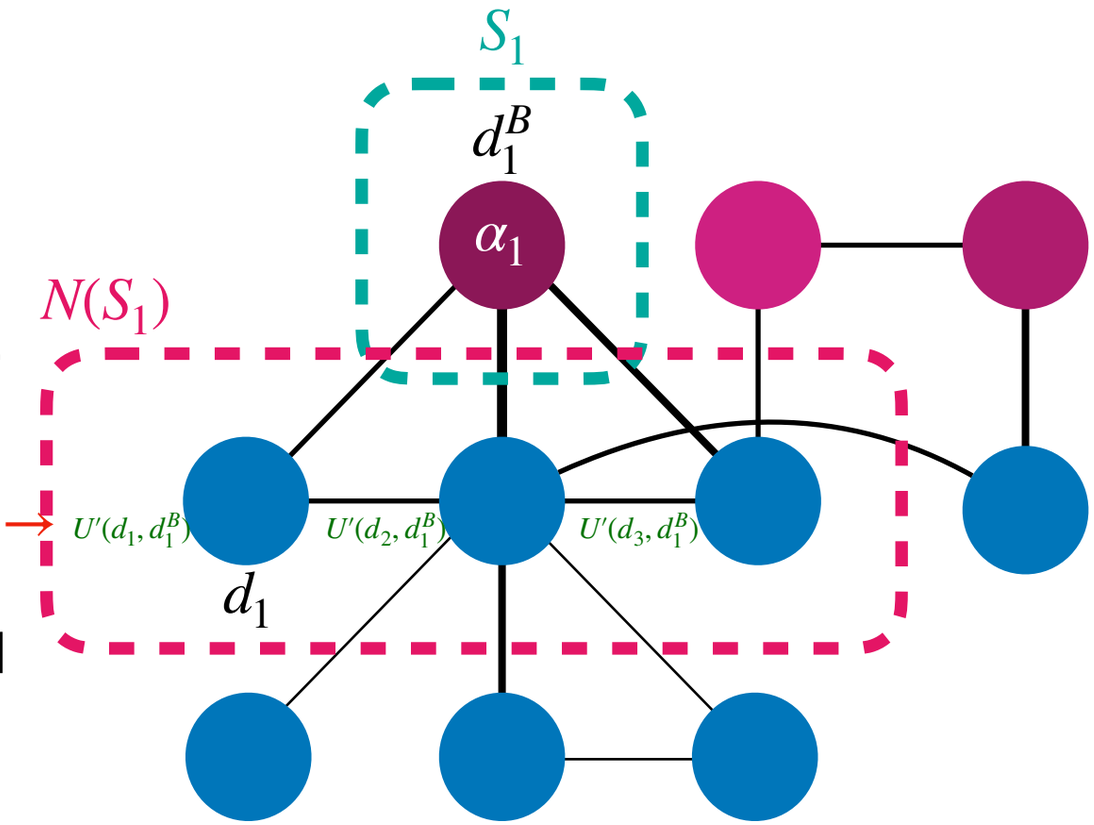
Algorithm

- For each boundary vertex d_i^B with a fixed value α_i, the upper bound on the other vertices imposed by d_i^B is calculated in the following steps.
- Let S_i be the set of vertices which their upper bound (imposed by d_i^B) is calculated.
- In each step, we find the least upper bound among the neighbors of S_i imposed by S_i and S_{i+1} is defined by adding the corresponding vertex to S_i.



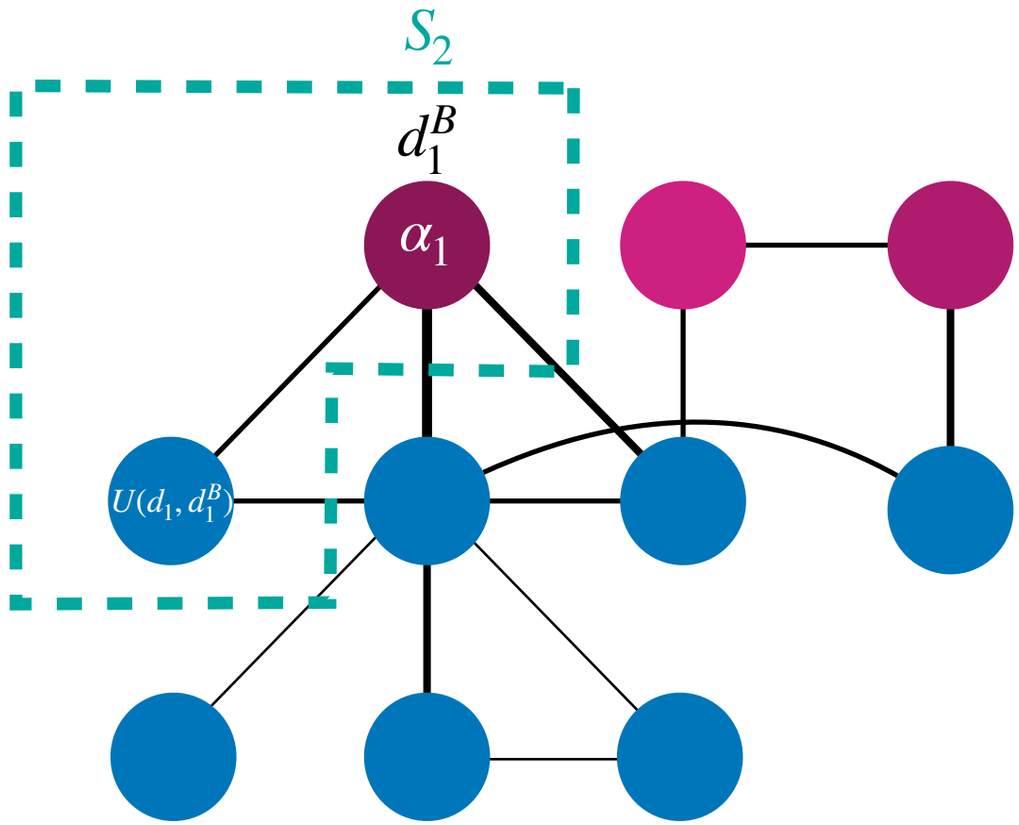
Algorithm

- For each boundary vertex d_i^B with a fixed value α_i, the upper bound on the other vertices imposed by d_i^B is calculated in the following steps.
- Let S_i be the set of vertices which their upper bound (imposed by d_i^B) is calculated.
- In each step, we find the least upper bound among the neighbors of S_i imposed by S_i and S_{i+1} is defined by adding the corresponding vertex to S_i.



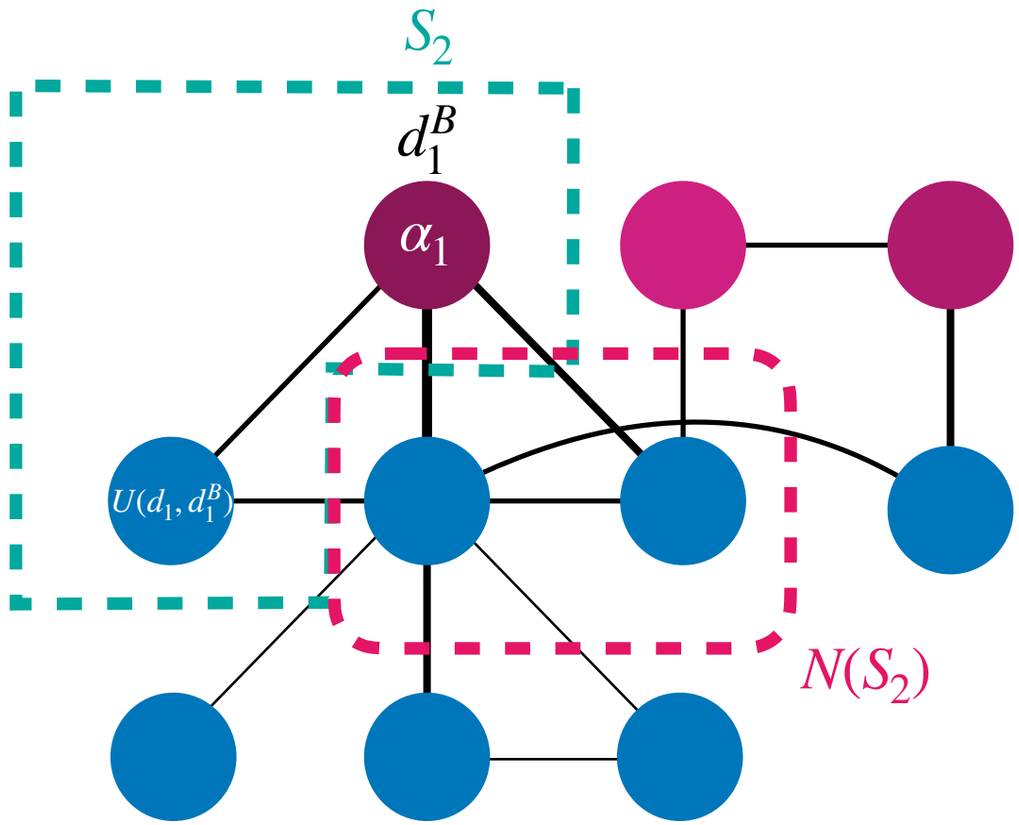
Algorithm

- For each boundary vertex d_i^B with a fixed value α_i, the upper bound on the other vertices imposed by d_i^B is calculated in the following steps.
- Let S_i be the set of vertices which their upper bound (imposed by d_i^B) is calculated.
- In each step, we find the least upper bound among the neighbors of S_i imposed by S_i and S_{i+1} is defined by adding the corresponding vertex to S_i.
- Let $U(d, d')$ be the lowest upper bound on d imposed by d'.



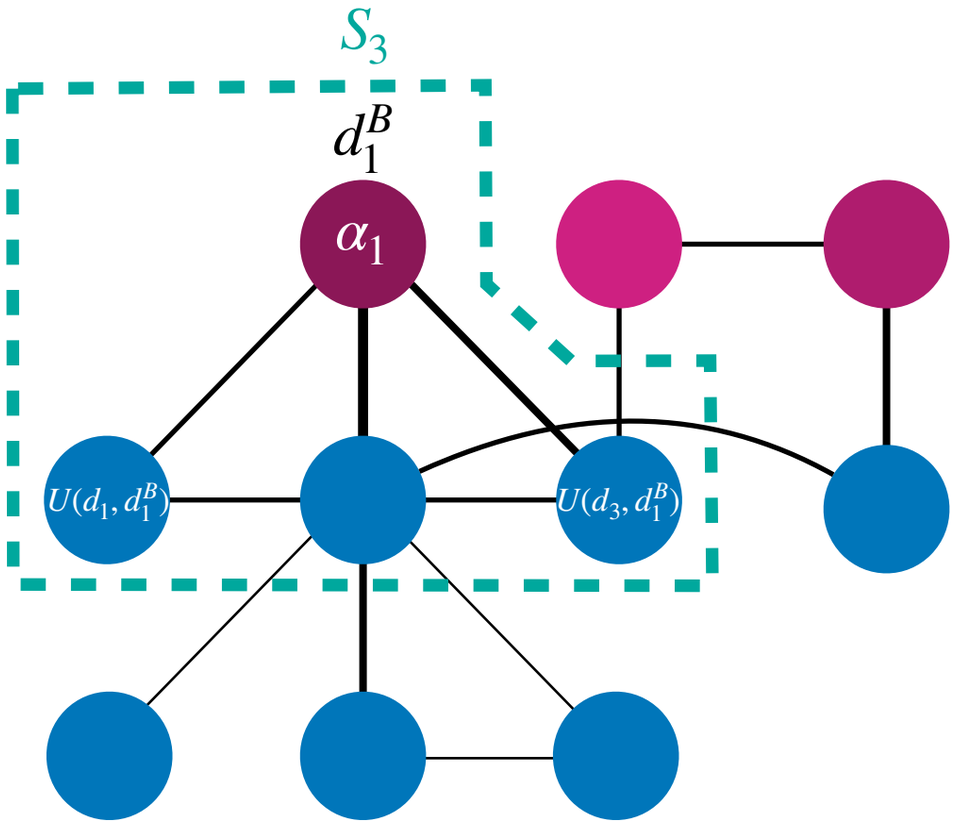
Algorithm

- For each boundary vertex d_i^B with a fixed value α_i, the upper bound on the other vertices imposed by d_i^B is calculated in the following steps.
- Let S_i be the set of vertices which their upper bound (imposed by d_i^B) is calculated.
- In each step, we find the least upper bound among the neighbors of S_i imposed by S_i and S_{i+1} is defined by adding the corresponding vertex to S_i.
- Let $U(d, d')$ be the lowest upper bound on d imposed by d'.



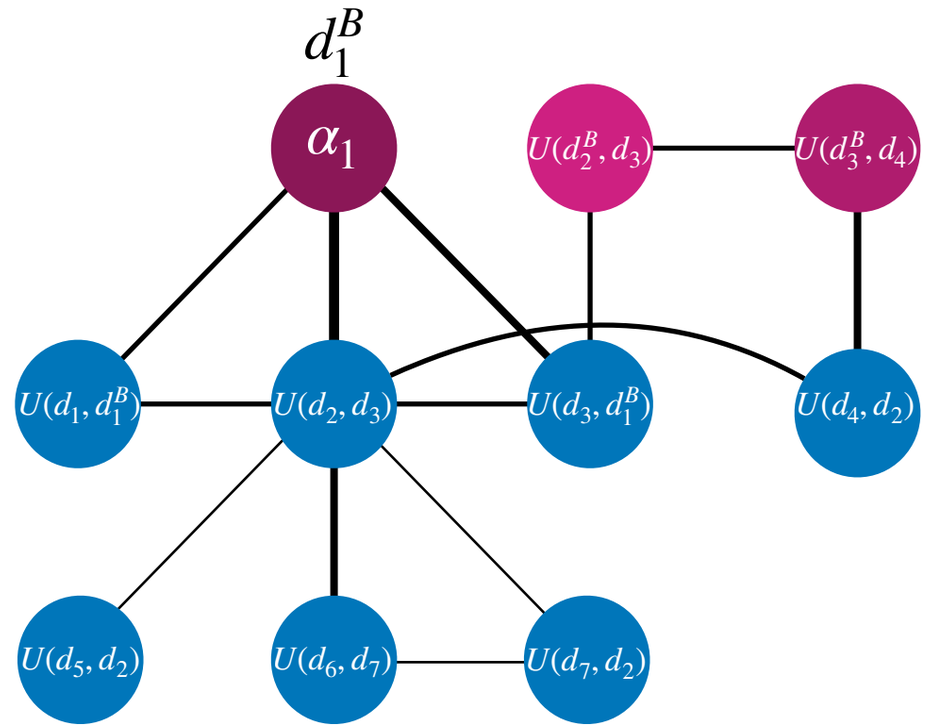
Algorithm

- For each boundary vertex d_i^B with a fixed value α_i, the upper bound on the other vertices imposed by d_i^B is calculated in the following steps.
- Let S_i be the set of vertices which their upper bound (imposed by d_i^B) is calculated.
- In each step, we find the least upper bound among the neighbors of S_i imposed by S_i and S_{i+1} is defined by adding the corresponding vertex to S_i.
- Let $U(d, d')$ be the lowest upper bound on d imposed by d'.



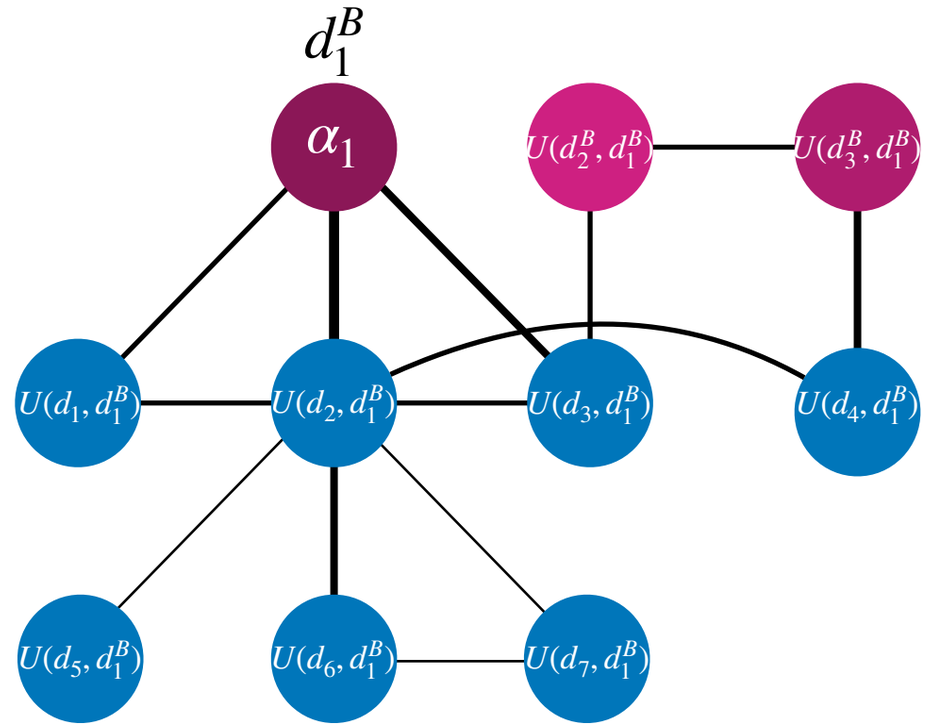
Algorithm

- For each boundary vertex d_i^B with a fixed value α_i, the upper bound on the other vertices imposed by d_i^B is calculated in the following steps.
- Let S_i be the set of vertices which their upper bound (imposed by d_i^B) is calculated.
- In each step, we find the least upper bound among the neighbors of S_i imposed by S_i and S_{i+1} is defined by adding the corresponding vertex to S_i.
- Let $U(d, d')$ be the lowest upper bound on d imposed by d'.



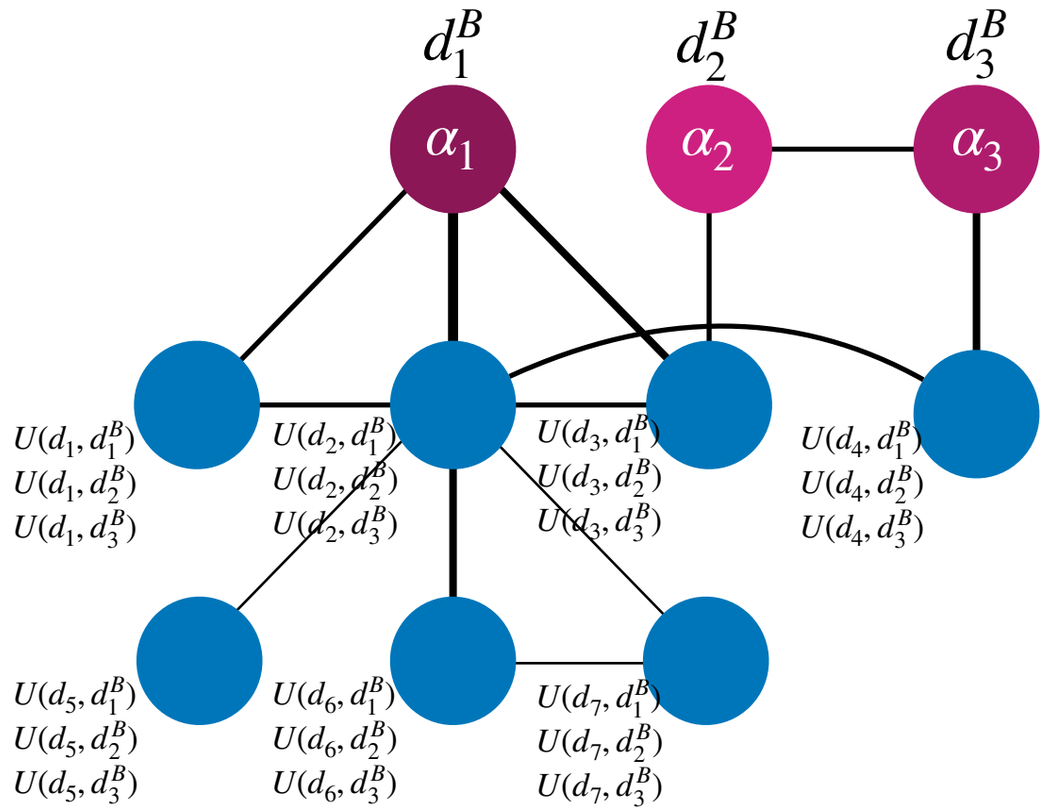
Algorithm

- For each boundary vertex d_i^B with a fixed value α_i, the upper bound on the other vertices imposed by d_i^B is calculated in the following steps.
- Let S_i be the set of vertices which their upper bound (imposed by d_i^B) is calculated.
- In each step, we find the least upper bound among the neighbors of S_i imposed by S_i and S_{i+1} is defined by adding the corresponding vertex to S_i.
- Let $U(d, d_i^B)$ be the lowest upper bound on d imposed by d_i^B.



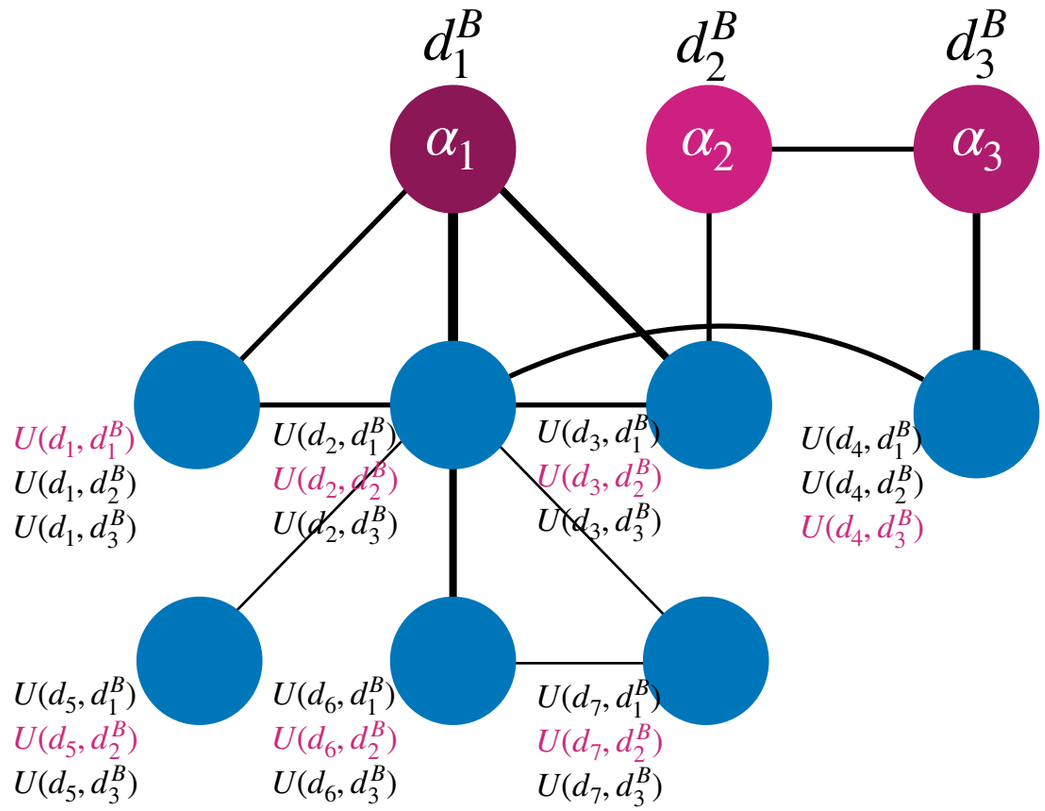
Algorithm

- Let $U(d)$ be the lowest upper bound on d over all the choices of d_i^B.



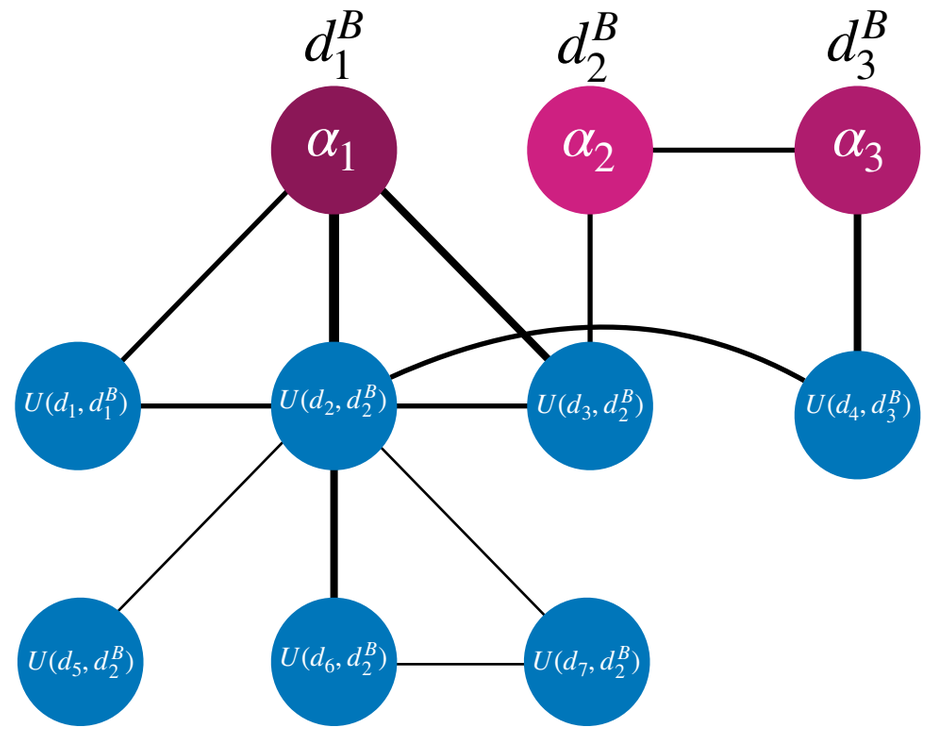
Algorithm

- Let $U(d)$ be the lowest upper bound on d over all the choices of d_i^B.



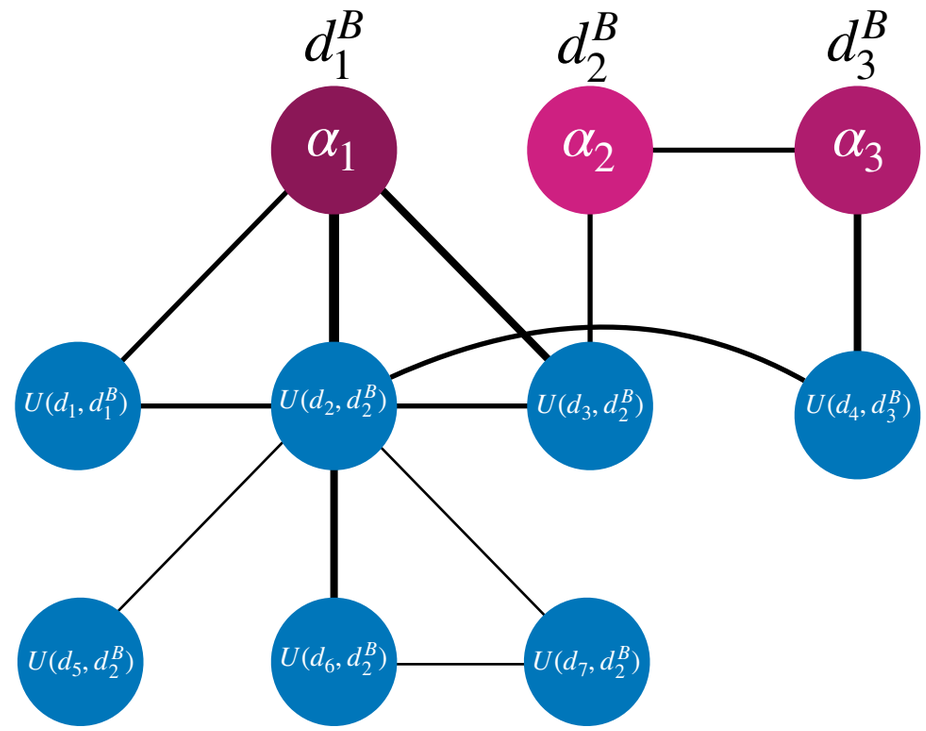
Algorithm

- Let $U(d)$ be the lowest upper bound on d over all the choices of d_i^B.



Algorithm

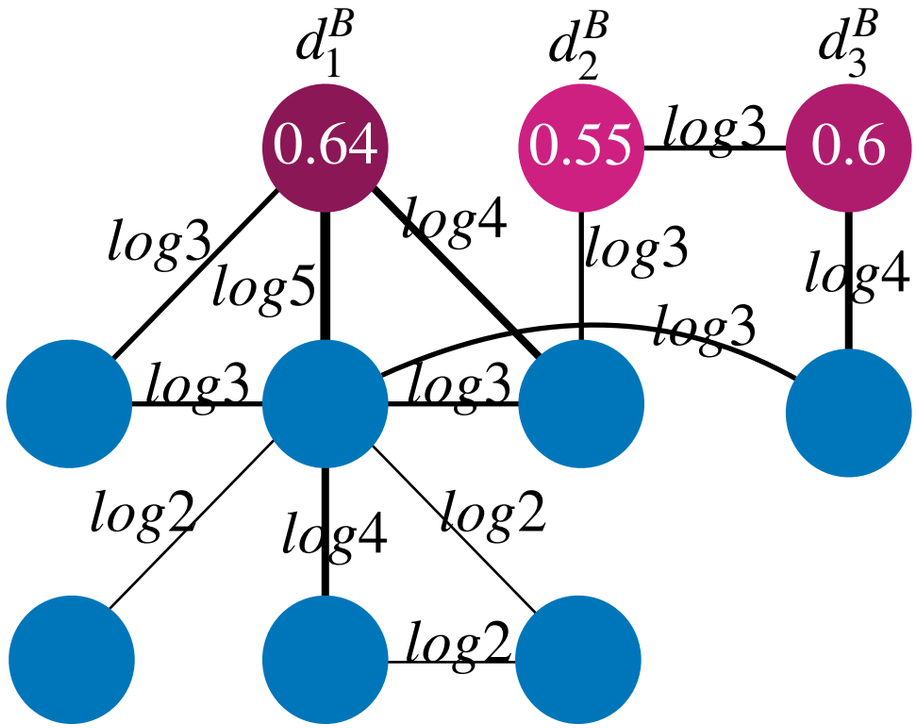
- Let $U(d)$ be the lowest upper bound on d over all the choices of d_i^B.



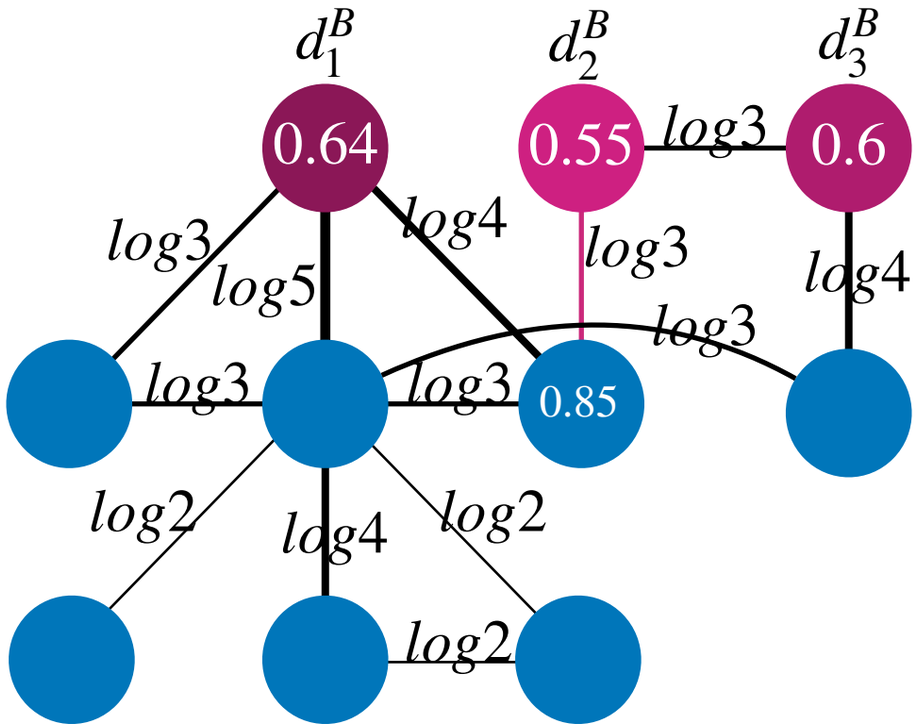
Theorem

The algorithm above finds the unique optimal heterogeneous DP mechanism in polynomial time, in the order of the number of vertices and edges.

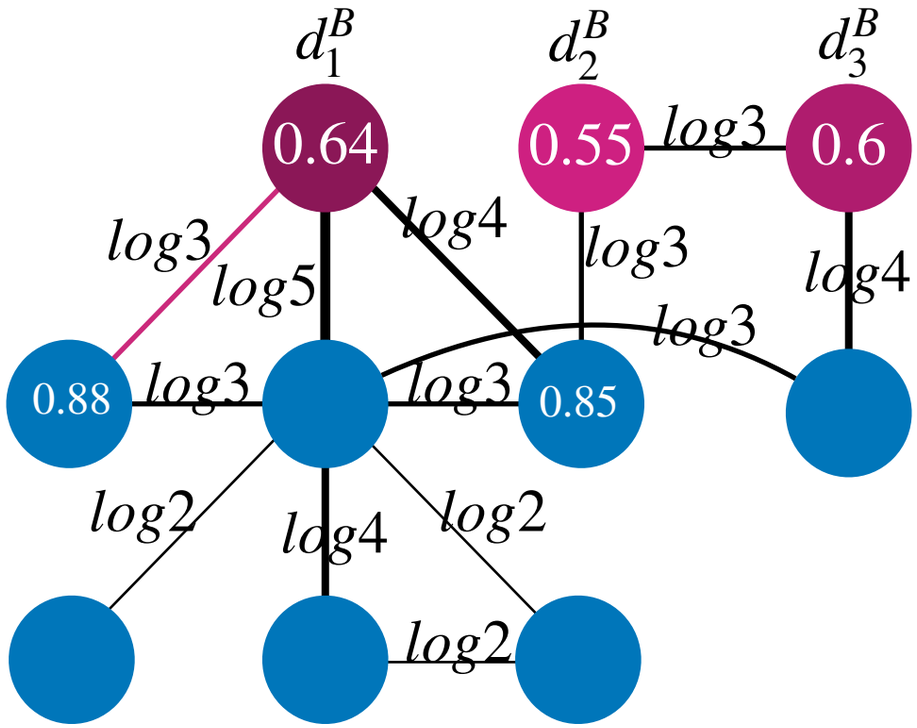
Example:


d_1^B
d_2^B
d_3^B
0.64
0.55
0.6
log3
log3
log4
log5
log3
log4
log3
log3
log3
log2
log4
log2
log2

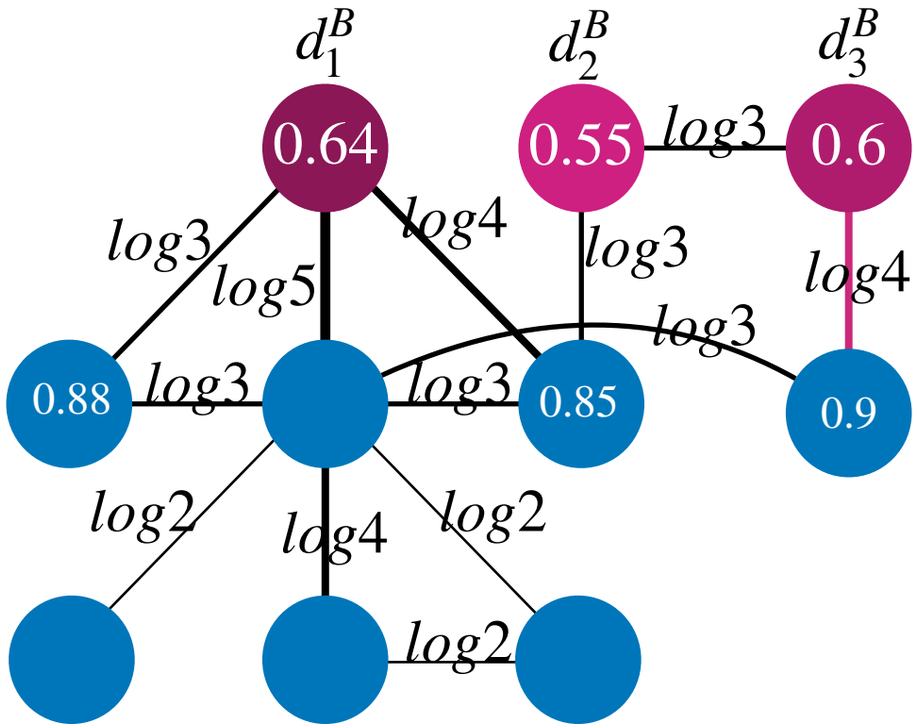
Example:


d_1^B
d_2^B
d_3^B
0.64
0.55
0.6
0.85
log3
log4
log3
log5
log3
log4
log3
log3
log3
log2
log4
log2
log2

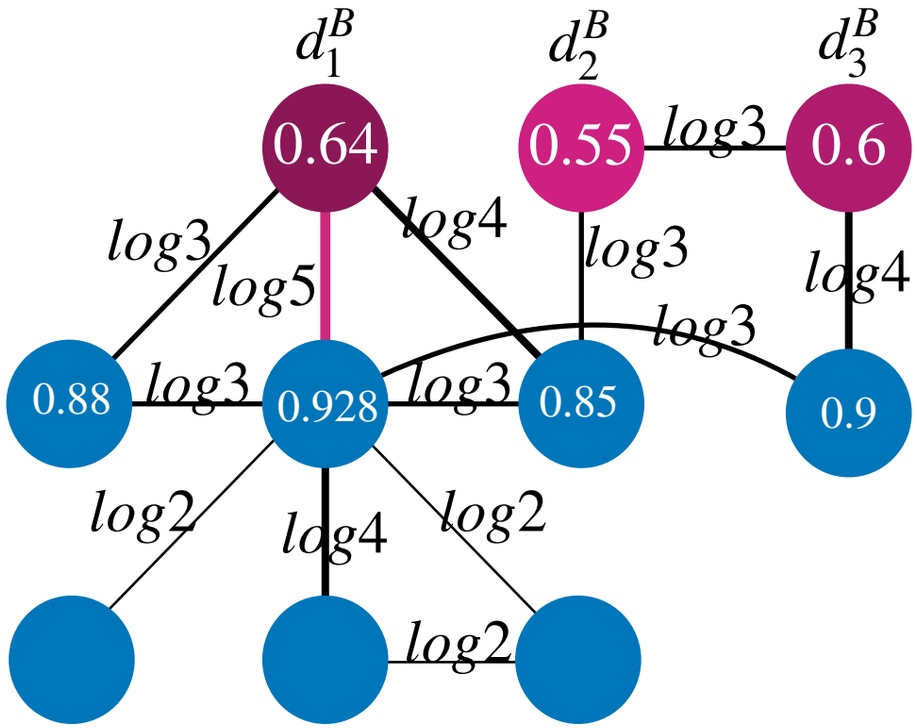
Example:


d_1^B
d_2^B
d_3^B
0.64
0.55
0.6
0.88
0.85
log3
log4
log3
log5
log3
log3
log3
log4
log2
log4
log2
log2

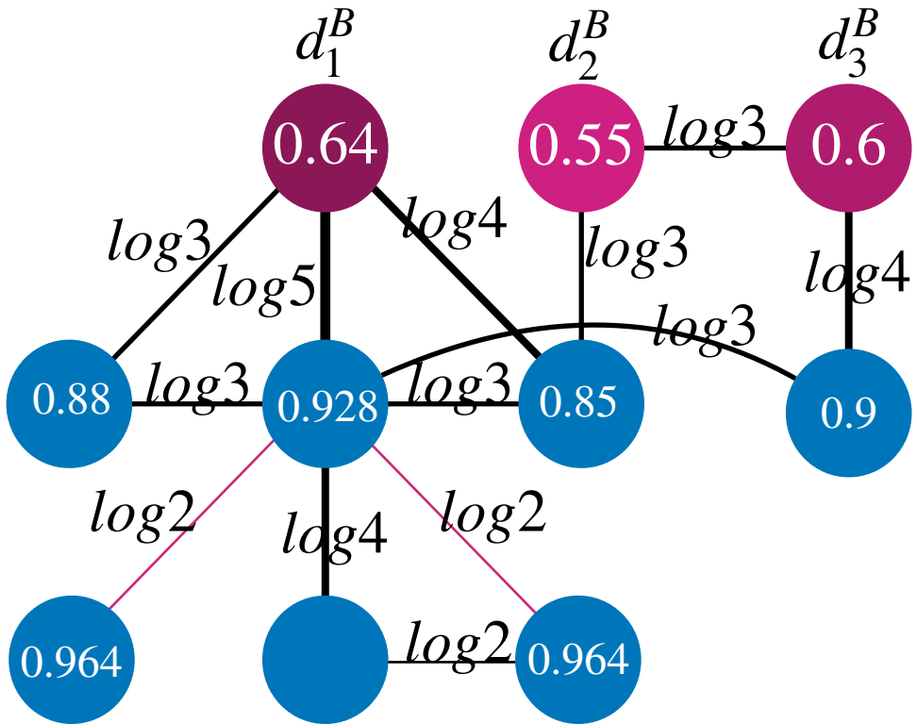
Example:


d_1^B
d_2^B
d_3^B
0.64
0.55
0.6
log3
log4
log3
log5
log4
log3
log3
0.88
log3
log3
0.85
0.9
log2
log4
log2
log2

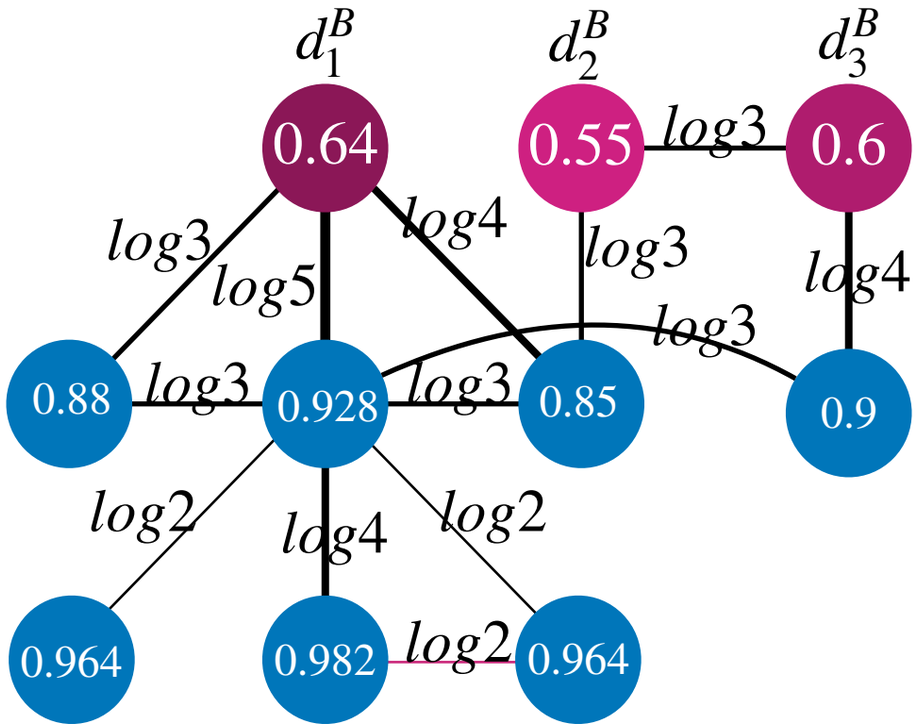
Example:


d_1^B
d_2^B
d_3^B
0.64
0.55
0.6
log3
log3
log4
log5
log4
log3
log3
0.88
log3
0.928
log3
0.85
0.9
log2
log4
log2
log2

Example:


d_1^B
d_2^B
d_3^B
0.64
0.55
0.6
log3
log3
log4
log5
log3
log4
log3
0.88
log3
0.928
log3
0.85
0.9
log2
log4
log2
0.964
log2
0.964

Example:


d_1^B
d_2^B
d_3^B
0.64
0.55
0.6
log3
log3
log4
log5
log3
log4
log3
0.88
log3
0.928
log3
0.85
0.9
log2
log4
log2
0.964
0.982
log2
0.964

Thank You
For Your Attention

Any Questions?