Simultaneous Nearest Neighbor Search

Piotr Indyk
MIT

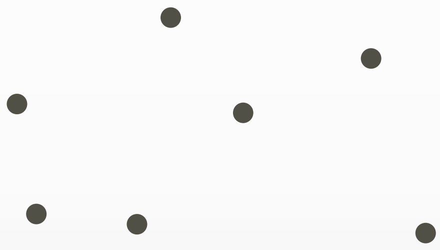
Robert Kleinberg
Cornell

Sepideh Mahabadi
MIT

Yang Yuan
Cornell

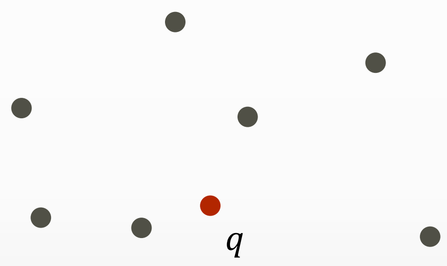
Nearest Neighbor

- Dataset of n points P in a metric space (X, d_X)



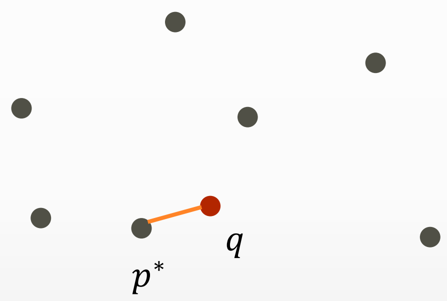
Nearest Neighbor

- Dataset of n points P in a metric space (X, d_X)
- A query point comes online q



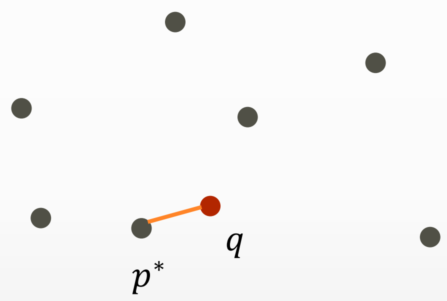
Nearest Neighbor

- Dataset of n points P in a metric space (X, d_X)
- A query point comes online q
- Goal:
  - Find the nearest data-set point p^*



Nearest Neighbor

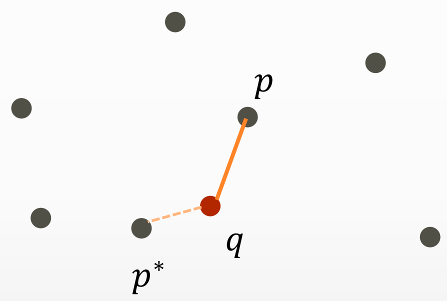
- Dataset of n points P in a metric space (X, d_X)
- A query point comes online q
- Goal:
  - Find the nearest data-set point p^*
  - Do it in sub-linear time



Approximate Nearest Neighbor

MIT CSAIL

- Dataset of n points P in a metric space (X, d_X)
- A query point comes online q



- Goal:
  - Find the nearest data-set point p^*
  - Do it in sub-linear time
    - Approximate Nearest Neighbor

What if

We have multiple queries

We need the results of the queries to be related.

What if

We have multiple queries

We need the results of the queries to be related.

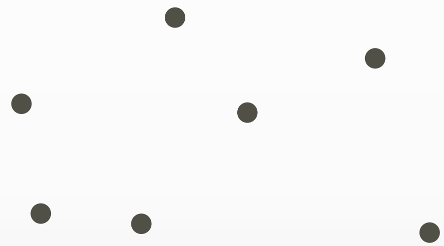
Example:

- Noisy image
- For each pixel find the true color
- Neighboring pixels have similar color

Simultaneous NN Problem

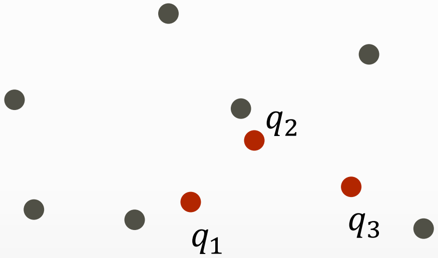
The SNN problem
(Felzenszwalb'15)

- Dataset of n points P in a metric space (X, d_X)



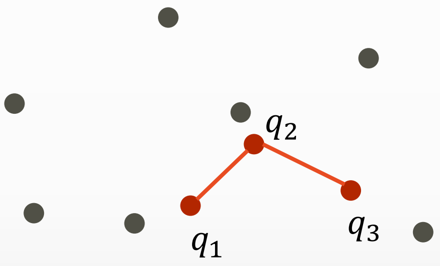
The SNN problem
(Felzenszwalb'15)

- Dataset of n points P in a metric space (X, d_X)
- Query comes online and contains
  - k query points $Q = (q_1, \ldots, q_k)$



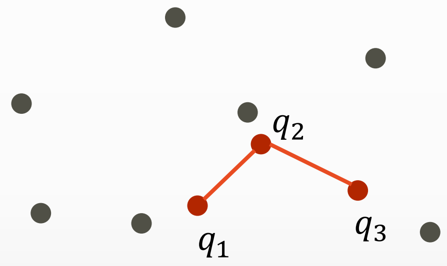
The SNN problem
(Felzenszwalb'15)

- Dataset of n points P in a metric space (X, d_X)
- Query comes online and contains
  - k query points $Q = (q_1, \ldots, q_k)$
  - A compatibility graph $G = (Q, E_G)$



The SNN problem
(Felzenszwalb'15)

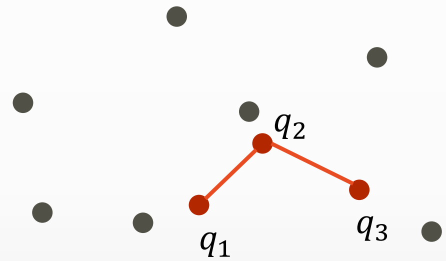
- Dataset of n points P in a metric space (X, d_X)
- Query comes online and contains
  - k query points $Q = (q_1, \dots, q_k)$
  - A compatibility graph $G = (Q, E_G)$

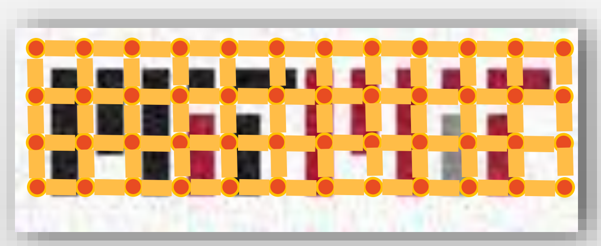


The SNN problem (Felzenszwalb'15)

- Dataset of n points P in a metric space (X, d_X)

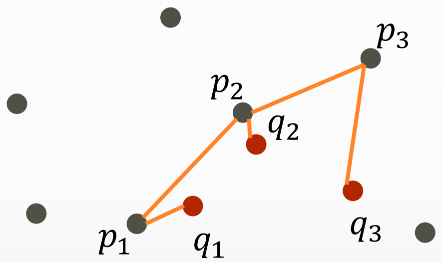
- Query comes online and contains
  - k query points $Q = (q_1, \ldots, q_k)$
  - A compatibility graph $G = (Q, E_G)$





The SNN problem
(Felzenszwalb'15)

- Dataset of n points P in a metric space (X, d_X)
- Query comes online and contains
  - k query points $Q = (q_1, \ldots, q_k)$
  - A compatibility graph $G = (Q, E_G)$

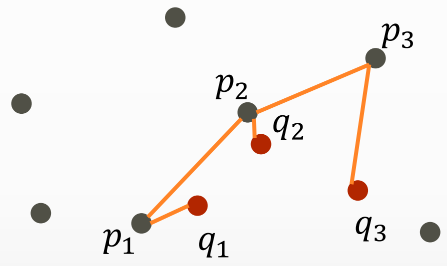


- Goal is to report $(p_1, \ldots, p_k)$, $p_i \in P$, that minimizes

$$\sum_{i=1}^{k} d_X(q_i, p_i) + \sum_{(q_i, q_j) \in E_G} d_X(p_i, p_j)$$

The Generalized SNN

- Dataset of n points P in a metric space (X, d_X)
- Query comes online and contains
  - k query points $Q = (q_1, \ldots, q_k)$
  - A compatibility graph $G = (Q, E_G)$



- Goal is to report $(p_1, \ldots, p_k)$, $p_i \in P$, that minimizes

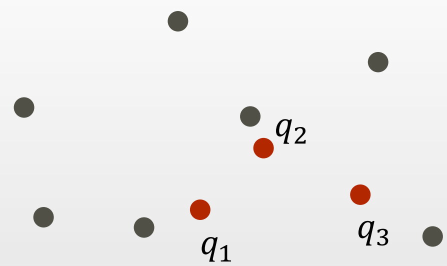
$$\sum_{i=1}^{k} \kappa_i d_Y(q_i, p_i) + \sum_{(q_i, q_j) \in E_G} \lambda_{i,j} d_X(p_i, p_j)$$

Independent NN Algorithm

Independent NN Algorithm

INN Algorithm

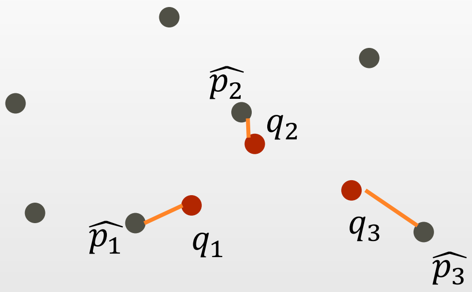
- For each query point $q_i \in Q$



Independent NN Algorithm

INN Algorithm

- For each query point $q_i \in Q$
  - Independently find a (approximate) nearest neighbor $\boldsymbol{\hat{p}_i}$ **(Searching step)**

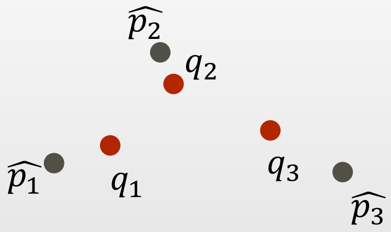


Independent NN Algorithm

INN Algorithm

- For each query point $q_i \in Q$
  - Independently find a (approximate) nearest neighbor $\widehat{\boldsymbol{p}}_{\boldsymbol{i}}$

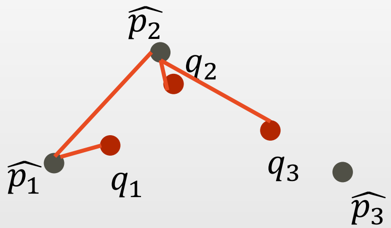
  (Searching step)
- Replace the label set P with the reduced set $\widehat{\boldsymbol{P}} = \{\widehat{\boldsymbol{p_1}}, \dots, \widehat{\boldsymbol{p_k}}\}$ **(Pruning step)**



Independent NN Algorithm

INN Algorithm

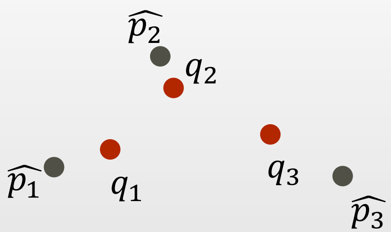
- For each query point $q_i \in Q$
  - Independently find a (approximate) nearest neighbor $\widehat{\boldsymbol{p}_{\boldsymbol{i}}}$ **(Searching step)**
- Replace the label set P with the reduced set $\widehat{\boldsymbol{P}} = \{\widehat{\boldsymbol{p}_{\boldsymbol{1}}}, \dots, \widehat{\boldsymbol{p}_{\boldsymbol{k}}}\}$ **(Pruning step)**
- Solve the problem for $\widehat{P}$



Independent NN Algorithm

INN Algorithm

- For each query point $q_i \in Q$
  - Independently find a (approximate) nearest neighbor $\boldsymbol{\widehat{p_i}}$ **(Searching step)**
- Replace the label set P with the reduced set $\boldsymbol{\widehat{P}} = \{\boldsymbol{\widehat{p_1}}, \ldots, \boldsymbol{\widehat{p_k}}\}$ **(Pruning step)**
- Solve the problem for $\widehat{P}$

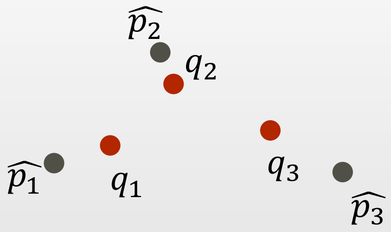


➢ Reduces the size of labels from n down to k

Independent NN Algorithm

INN Algorithm

- For each query point $q_i \in Q$
  - Independently find a (approximate) nearest neighbor $\widehat{\boldsymbol{p}}_{\boldsymbol{i}}$ **(Searching step)**
- Replace the label set P with the reduced set $\widehat{\boldsymbol{P}} = \{\widehat{\boldsymbol{p_1}}, \dots, \widehat{\boldsymbol{p_k}}\}$ **(Pruning step)**
- Solve the problem for $\widehat{P}$



➢ Reduces the size of labels from n down to k

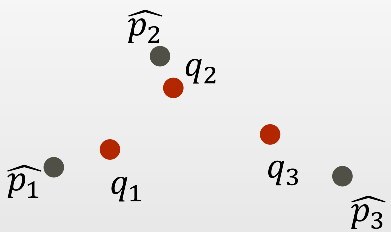
➢ The optimal value increases by a factor $\boldsymbol{\alpha}$

- ➢ **pruning gap**

Independent NN Algorithm

INN Algorithm

- For each query point $q_i \in Q$
  - Independently find a (approximate) nearest neighbor $\widehat{\boldsymbol{p}_{\boldsymbol{i}}}$ **(Searching step)**
- Replace the label set P with the reduced set $\widehat{\boldsymbol{P}} = \{\widehat{\boldsymbol{p}_{\boldsymbol{1}}}, \ldots, \widehat{\boldsymbol{p}_{\boldsymbol{k}}}\}$ **(Pruning step)**
- Solve the problem for $\widehat{P}$



➢ Reduces the size of labels from n down to k

➢ The optimal value increases by a factor $\boldsymbol{\alpha}$

- ➢ **pruning gap**

➢ Any metric labeling β-approximate algorithm can be used on the reduced set , giving us $(\alpha \cdot \beta)$-approximate algorithm.

Results

Results

- Prove bounds for the pruning gap

Results

- Prove bounds for the pruning gap
  - $\boldsymbol{\alpha = O\left(\frac{\log k}{\log \log \mathrm{k}}\right)}$

Results

- Prove bounds for the pruning gap
  - $\alpha = O\left(\frac{\log k}{\log \log \mathrm{k}}\right)$
  - $\alpha = \Omega\left(\sqrt{\log k}\right)$

Results

- Prove bounds for the pruning gap
  - $\boldsymbol{\alpha = O\left(\frac{\log k}{\log \log k}\right)}$
  - $\boldsymbol{\alpha = \Omega\left(\sqrt{\log k}\right)}$
- For r-sparse graph: $\boldsymbol{\alpha = O(r)}$

Results

- Prove bounds for the pruning gap
  - $\alpha = O\left(\frac{\log k}{\log \log \mathrm{k}}\right)$
  - $\alpha = \Omega\left(\sqrt{\log k}\right)$
- For r-sparse graph: $\alpha = O(r)$
  - Graphs with pseudo-arboricity r: each edge can be mapped to a vertex such that at most r edges are mapped to any vertex

Results

- Prove bounds for the pruning gap
  - $\boldsymbol{\alpha = O\left(\frac{\log k}{\log \log \mathrm{k}}\right)}$
  - $\boldsymbol{\alpha = \Omega\left(\sqrt{\log k}\right)}$
- For r-sparse graph: $\boldsymbol{\alpha = O(r)}$
  - Graphs with pseudo-arboricity r: each edge can be mapped to a vertex such that at most r edges are mapped to any vertex
  - Would mean constant approximation factor for trees, grids, planar graphs, …, and in particular $O(r)$-approximation for r-degree graphs

Results

- Prove bounds for the pruning gap
  - $\alpha = O\left(\frac{\log k}{\log \log k}\right)$
  - $\alpha = \Omega\left(\sqrt{\log k}\right)$
- For r-sparse graph: $\alpha = O(r)$
  - Graphs with pseudo-arboricity r: each edge can be mapped to a vertex such that at most r edges are mapped to any vertex
  - Would mean constant approximation factor for trees, grids, planar graphs, …, and in particular $O(r)$-approximation for r-degree graphs
- α is very close to one in experiments

Results

- Prove bounds for the pruning gap
  - $\alpha = O\left(\frac{\log k}{\log\log k}\right)$
  - $\alpha = \Omega\left(\sqrt{\log k}\right)$
- For r-sparse graph: $\alpha = O(r)$
  - Graphs with pseudo-arboricity r: each edge can be mapped to a vertex such that at most r edges are mapped to any vertex
  - Would mean constant approximation factor for trees, grids, planar graphs, …, and in particular $O(r)$-approximation for r-degree graphs
- α is very close to one in experiments

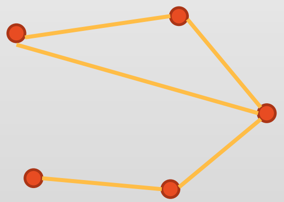
Overview of the proof for

$$\alpha = O\left(\frac{\log k}{\log\log k}\right)$$

0-Extension Problem [Kar98]

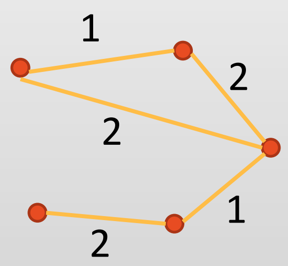
0-Extension Problem [Kar98]

- The input:
  - a graph $H(V, E)$



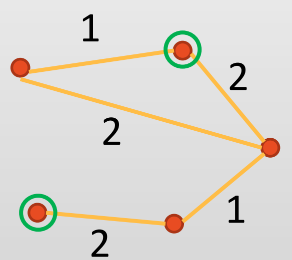
0-Extension Problem [Kar98]

- The input:
  - a graph $H(V, E)$
  - a weight function $w(e)$



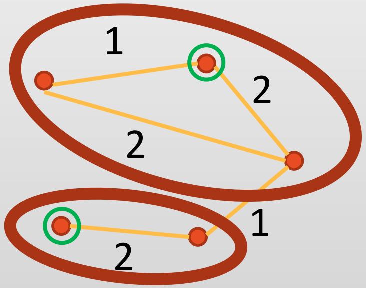
0-Extension Problem [Kar98]

- The input:
  - a graph $H(V, E)$
  - a weight function $w(e)$
  - a set of terminals $T \subset V$



0-Extension Problem [Kar98]

- The input:
  - a graph $H(V, E)$
  - a weight function $w(e)$
  - a set of terminals $T \subset V$
- The goal: find a mapping $f: V \to T$ s.t.
  - Each terminal is mapped to itself
  - Minimize $\sum_{(u,v)\in E} w(u,v) d(f(u), f(v))$



Cost = $1 \cdot d(t_1, t_2)$

0-Extension Problem

Prior work: [CKR05, FHRT03, AFHKTT04, LN04]

0-Extension Problem

Prior work: [CKR05, FHRT03, AFHKTT04, LN04]

- **Upper bounds:**
  - **E.g.** $O(\frac{\log |T|}{\log \log |T|})$ approximation algorithm [CKR05]

0-Extension Problem

Prior work: [CKR05, FHRT03, AFHKTT04, LN04]

- **Upper bounds:**
  - **E.g.** $O(\frac{\log |T|}{\log \log |T|})$ approximation algorithm [CKR05]
  - consider the metric relaxation of the LP for the problem
  - Solve LP
  - Round the solution

0-Extension Problem

Prior work: [CKR05, FHRT03, AFHKTT04, LN04]

- **Upper bounds:**
  - **E.g.** $O(\frac{\log |T|}{\log \log |T|})$ approximation algorithm [CKR05]
  - consider the metric relaxation of the LP for the problem
  - Solve LP
  - Round the solution
- $\Omega(\sqrt{\log |T|})$ integrality gap [FHRT03]

0-Extension Problem

Prior work: [CKR05, FHRT03, AFHKTT04, LN04]

- **Upper bounds:**
  - **E.g.** $O(\frac{\log |T|}{\log \log |T|})$ approximation algorithm [CKR05]
  - consider the metric relaxation of the LP for the problem
  - Solve LP
  - Round the solution
- $\Omega(\sqrt{\log |T|})$ integrality gap [FHRT03]
- Efficient if the number of terminals is low

Connection to SNN

- **SNN:** $(\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{G})$
- **0-Extension:** $(\boldsymbol{V}, \boldsymbol{H}, \boldsymbol{w}, \boldsymbol{T})$

Connection to SNN

- **SNN:** $(\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{G})$
- **0-Extension:** $(\boldsymbol{V}, \boldsymbol{H}, \boldsymbol{w}, \boldsymbol{T})$

1. 0-extension can be solved using generalized SNN
   - $Q = V$, $P = T$, $\lambda_{i,j} = w(i,j)$, $\kappa_i = \infty \; if \; q_i \in T \; and \; 0 \; O.w.$

Connection to SNN

- **SNN:** $(\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{G})$
- **0-Extension:** $(\boldsymbol{V}, \boldsymbol{H}, \boldsymbol{w}, \boldsymbol{T})$

1. 0-extension can be solved using generalized SNN
   - $Q = V$, $P = T$, $\lambda_{i,j} = w(i,j)$, $\kappa_i = \infty$ if $q_i \in T$ and 0 $O.w.$
2. SNN can be solved using 0-extension in a black-box manner
   - Set: $T = P$, $V = Q \cup P$, $w = 1$, $H = G \cup \{(q_i, \widehat{p_i}) | i\}$

Connection to SNN

- **SNN:** $(\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{G})$
- **0-Extension:** $(\boldsymbol{V}, \boldsymbol{H}, \boldsymbol{w}, \boldsymbol{T})$

1. 0-extension can be solved using generalized SNN
   - $Q = V$, $P = T$, $\lambda_{i,j} = w(i,j)$, $\kappa_i = \infty$ $if\ q_i \in T\ and\ 0\ O.w.$
2. SNN can be solved using 0-extension in a black-box manner
   - Set: $T = P$, $V = Q \cup P$, $w = 1$, $H = G \cup \{(q_i, \widehat{p_i}) | i\}$
   - giving $\boldsymbol{O}(\frac{\boldsymbol{log\ n}}{\boldsymbol{log\ log\ n}})$ **approximation** algorithm

Connection to SNN

- **SNN:** $(\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{G})$
- **0-Extension:** $(\boldsymbol{V}, \boldsymbol{H}, \boldsymbol{w}, \boldsymbol{T})$

1. 0-extension can be solved using generalized SNN
   - $Q = V$, $P = T$, $\lambda_{i,j} = w(i,j)$, $\kappa_i = \infty$ $if\ q_i \in T\ and\ 0\ O.w.$
2. SNN can be solved using 0-extension in a black-box manner
   - Set: $T = P$, $V = Q \cup P$, $w = 1$, $H = G \cup \{(q_i, \widehat{p_i}) | i\}$
   - giving $\boldsymbol{O}(\frac{\boldsymbol{log\ n}}{\boldsymbol{log\ log\ n}})$ **approximation** algorithm
3. Improve to depend only on $\boldsymbol{k}$ **not n**

Connection to SNN

- **SNN:** $(\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{G})$
- **0-Extension:** $(\boldsymbol{V}, \boldsymbol{H}, \boldsymbol{w}, \boldsymbol{T})$

1. 0-extension can be solved using generalized SNN
   - $Q = V$, $P = T$, $\lambda_{i,j} = w(i,j)$, $\kappa_i = \infty$ $if\ q_i \in T\ and\ 0\ O.w.$
2. SNN can be solved using 0-extension in a black-box manner
   - Set: $T = P$, $V = Q \cup P$, $w = 1$, $H = G \cup \{(q_i, \widehat{p_i}) | i\}$
   - giving $\boldsymbol{O}(\frac{\boldsymbol{log\ n}}{\boldsymbol{log\ log\ n}})$ **approximation** algorithm
3. Improve to depend only on $\boldsymbol{k}$ **not n**
   - Analyzing INN using 0-extension in a “grey-box” manner

Connection to SNN

- **SNN:** $(\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{G})$
- **0-Extension:** $(\boldsymbol{V}, \boldsymbol{H}, \boldsymbol{w}, \boldsymbol{T})$

1. 0-extension can be solved using generalized SNN
   - $Q = V$, $P = T$, $\lambda_{i,j} = w(i,j)$, $\kappa_i = \infty$ if $q_i \in T$ and 0 $O.w.$
2. SNN can be solved using 0-extension in a black-box manner
   - Set: $T = P$, $V = Q \cup P$, $w = 1$, $H = G \cup \{(q_i, \widehat{p_i}) | i\}$
   - giving $\boldsymbol{O}(\frac{\boldsymbol{log\ n}}{\boldsymbol{log\ log\ n}})$ **approximation** algorithm
3. Improve to depend only on $\boldsymbol{k}$ **not n**
   - Analyzing INN using 0-extension in a “grey-box” manner
   - Using subtle properties of existing algorithms for 0-extension

Connection to SNN

- **SNN:** $(\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{G})$
- **0-Extension:** $(\boldsymbol{V}, \boldsymbol{H}, \boldsymbol{w}, \boldsymbol{T})$

1. 0-extension can be solved using generalized SNN
   - $Q = V \text{ , } P = T \text{ , } \lambda_{i,j} = w(i,j) \text{ , } \kappa_i = \infty \text{ } if \text{ } q_i \in T \text{ } and \text{ } 0 \text{ } O.w.$
2. SNN can be solved using 0-extension in a black-box manner
   - Set: $T = P \text{ , } V = Q \cup P \text{ , } w = 1 \text{ , } H = G \cup \{(q_i, \widehat{p_i}) | i\}$
   - giving $\boldsymbol{O}(\frac{\boldsymbol{log\ n}}{\boldsymbol{log\ log\ n}})$ **approximation** algorithm
3. Improve to depend only on $\boldsymbol{k}$ **not n**
   - Analyzing INN using 0-extension in a “grey-box” manner
   - Using subtle properties of existing algorithms for 0-extension
   - Leads to an $\boldsymbol{O}(\frac{\boldsymbol{log\ k}}{\boldsymbol{log\ log\ k}})$ **approximation**

Experiments

Experimental Results

- **De-noising problem**
  - Each pixel is a query point
  - Data set P : all $[256]^3$ possible colors
  - Graph: the grid

Experimental Results

- **De-noising problem**
  - Each pixel is a query point
  - Data set P : all $[256]^3$ possible colors
  - Graph: the grid

- **Algorithm**
  - Only consider the colors that appear in the noisy image

Experimental Results

- **De-noising problem**
  - Each pixel is a query point
  - Data set P : all $[256]^3$ possible colors
  - Graph: the grid

- **Algorithm**
  - Only consider the colors that appear in the noisy image

- **Result:** empirical pruning gap α is very close to 1, at most 1.024

Experimental Results

Image

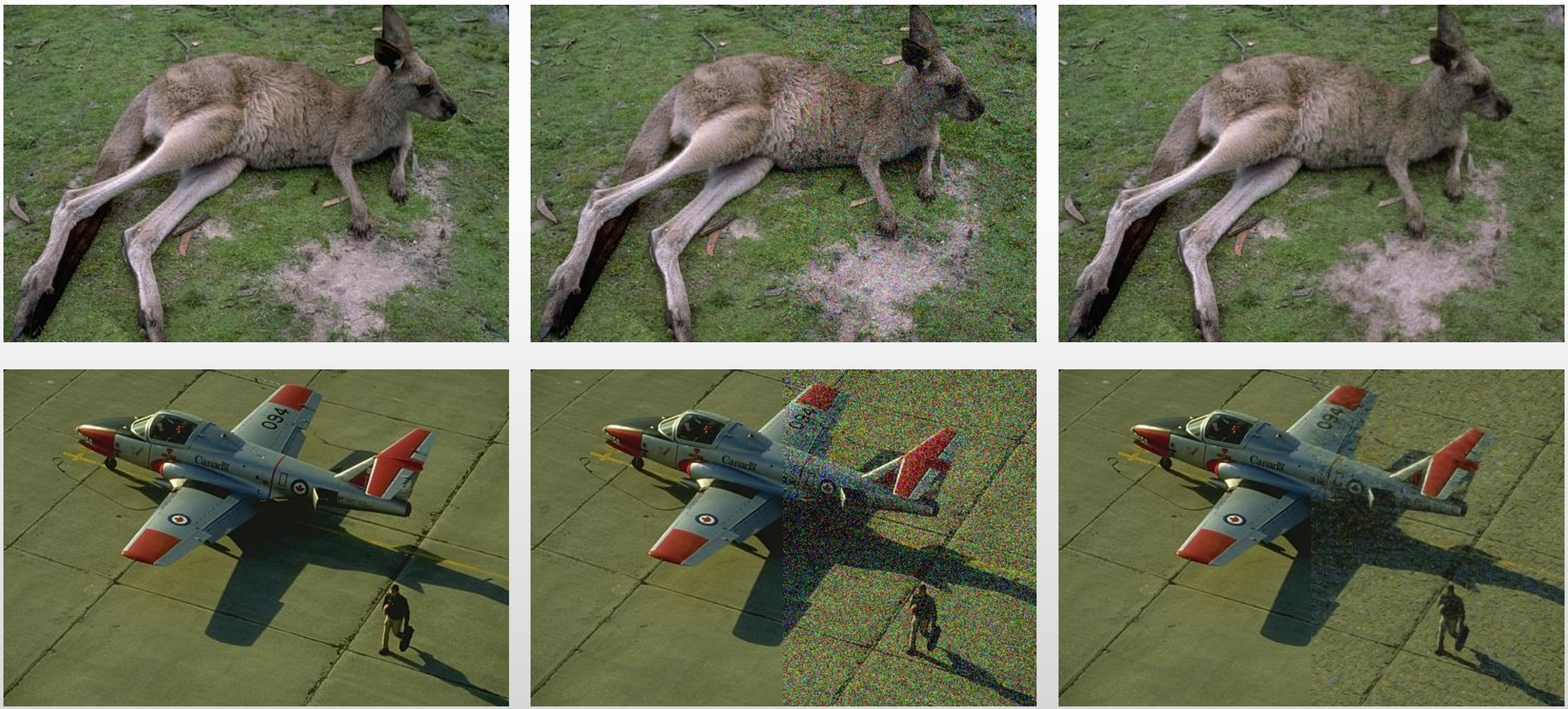
Noisy Image

De-noised using all colors

De-noised using noisy image colors

Experimental Results

Image | Half-Noisy | De-noised



Conclusion

- **Summary of Results**

Conclusion

- **Summary of Results**
  - Presented Independent NN pruning

Conclusion

- **Summary of Results**
  - Presented Independent NN pruning
  - Induces an extra factor α

Conclusion

MIT CSAIL

- **Summary of Results**
  - Presented Independent NN pruning
  - Induces an extra factor α
  - $\alpha = O(\frac{\log k}{\log\log k})$, $\alpha = \Omega(\sqrt{\log k})$

Conclusion

- **Summary of Results**
  - Presented Independent NN pruning
  - Induces an extra factor α
  - $\alpha = O(\frac{\log k}{\log \log k})$, $\alpha = \Omega(\sqrt{\log k})$
  - $\alpha = O(1)$ for sparse graphs that are mostly used in applications

Conclusion

- **Summary of Results**
  - Presented Independent NN pruning
  - Induces an extra factor α
  - $\alpha = O(\frac{\log k}{\log \log k})$, $\alpha = \Omega(\sqrt{\log k})$
  - $\alpha = O(1)$ for sparse graphs that are mostly used in applications
  - $\alpha \approx 1$ in the denoising experiments

Conclusion

MIT CSAIL

- **Summary of Results**
  - Presented Independent NN pruning
  - Induces an extra factor α
  - $\alpha = O(\frac{\log k}{\log \log k})$, $\alpha = \Omega(\sqrt{\log k})$
  - $\alpha = O(1)$ for sparse graphs that are mostly used in applications
  - $\alpha \approx 1$ in the denoising experiments
- **Open Problems**

Conclusion

- **Summary of Results**
  - Presented Independent NN pruning
  - Induces an extra factor α
  - $\alpha = O(\frac{\log k}{\log \log k})$, $\alpha = \Omega(\sqrt{\log k})$
  - $\alpha = O(1)$ for sparse graphs that are mostly used in applications
  - $\alpha \approx 1$ in the denoising experiments
- **Open Problems**
  - Prove tighter bounds for α

Conclusion

- **Summary of Results**
  - Presented Independent NN pruning
  - Induces an extra factor α
  - $\alpha = O(\frac{\log k}{\log \log k})$, $\alpha = \Omega(\sqrt{\log k})$
  - $\alpha = O(1)$ for sparse graphs that are mostly used in applications
  - $\alpha \approx 1$ in the denoising experiments
- **Open Problems**
  - Prove tighter bounds for α
  - Get better guarantees using different algorithm, i.e., instead of picking the closest point pick a few points.

Conclusion

- **Summary of Results**
  - Presented Independent NN pruning
  - Induces an extra factor α
  - $\alpha = O(\frac{\log k}{\log \log k})$, $\alpha = \Omega(\sqrt{\log k})$
  - $\alpha = O(1)$ for sparse graphs that are mostly used in applications
  - $\alpha \approx 1$ in the denoising experiments
- **Open Problems**
  - Prove tighter bounds for α
  - Get better guarantees using different algorithm, i.e., instead of picking the closest point pick a few points.
  - Solve the general case of the problem, i.e.,
    - where the metrics $d_Y(q_i, p_i)$ and $d_X(p_i, p_j)$ are different
    - There are weights

Conclusion

- **Summary of Results**
  - Presented Independent NN pruning
  - Induces an extra factor α
  - $\alpha = O(\frac{\log k}{\log \log k})$, $\alpha = \Omega(\sqrt{\log k})$
  - $\alpha = O(1)$ for sparse graphs that are mostly used in applications
  - $\alpha \approx 1$ in the denoising experiments
- **Open Problems**
  - Prove tighter bounds for α
  - Get better guarantees using different algorithm, i.e., instead of picking the closest point pick a few points.
  - Solve the general case of the problem, i.e.,
    - where the metrics $d_X(q_i, p_i)$ and $d_Y(p_i, p_j)$ are different
    - There are weights

Thank You!