The Maximum Exposure Problem

Neeraj Kumar, Stavros Sintos, Subhash Suri
University of California, Santa Barbara Duke University
Set of points $P$ in the plane,
Problem Description

Set of points $P$ in the plane, set of rectangular ranges $R$ covering them, integer parameter $k$
Problem Description

Set of points $P$ in the plane, set of rectangular ranges $\mathcal{R}$ covering them, integer parameter $k$

find $k$ ranges to delete so as to ‘expose’ a maximum number of points
Problem Description

Set of points $P$ in the plane, set of rectangular ranges $\mathcal{R}$ covering them, integer parameter $k$

find $k$ ranges to delete so as to ‘expose’ a maximum number of points
Problem Description

Set of points $P$ in the plane, set of rectangular ranges $\mathcal{R}$ covering them, integer parameter $k$

find $k$ ranges to delete so as to ‘expose’ a maximum number of points
Set of points $P$ in the plane, set of rectangular ranges $\mathcal{R}$ covering them, integer parameter $k$

find $k$ ranges to delete so as to ‘expose’ a maximum number of points
Motivation

- **Reliability of coverage:** points correspond to clients, ranges correspond to coverage of facilities
Motivation

- Reliability of coverage: points correspond to clients, ranges correspond to coverage of facilities

Which $k$ facilities to disable so as to affect maximum number of clients?
Motivation

- **Reliability of coverage**: points correspond to clients, ranges correspond to coverage of facilities

\[ \text{Which } k \text{ facilities to disable so as to affect maximum number of clients?} \]

- **Geometric constraint removal**: ranges correspond to *constraints*, points correspond to *rewards*

\[ \text{Maximize rewards by removing at most } k \text{ constraints} \]
Hardness of Max Exposure

Geometric counterpart of the *densest k-subhypergraph* problem

– studied recently in (APPROX’16, SODA’17), conditionally hard to approximate within $|V|^{1-\epsilon}$
Geometric counterpart of the densest $k$-subhypergraph problem

- studied recently in (APPROX’16, SODA’17), conditionally hard to approximate within $|V|^{1-\epsilon}$
- ranges $\mathcal{R}$ correspond to vertices of the hypergraph, points $P$ correspond to edges (defined by containment relation)
Hardness of Max Exposure

- Geometric counterpart of the *densest k-subhypergraph* problem
  - studied recently in (APPROX’16, SODA’17), conditionally hard to approximate within $|V|^{1-\epsilon}$
  - ranges $\mathcal{R}$ correspond to vertices of the hypergraph, points $P$ correspond to edges (defined by containment relation)

- With convex polygons, max-exposure is as hard as densest $k$-subhypergraph
  - Hypergraph $H = (X, E)$ can be transformed into max-exposure of convex ranges $\mathcal{R}$ and points $P$
Hardness of Max Exposure

- Geometric counterpart of the *densest k-subhypergraph* problem
  - studied recently in (APPROX’16, SODA’17), conditionally hard to approximate within $|V|^{1-\epsilon}$
  - ranges $\mathcal{R}$ correspond to vertices of the hypergraph, points $P$ correspond to edges (defined by containment relation)

- With convex polygons, max-exposure is as hard as densest $k$-subhypergraph
  - Hypergraph $H = (X, E)$ can be transformed into max-exposure of convex ranges $\mathcal{R}$ and points $P$

What about rectangle ranges?
Hardness of Max Exposure

- Geometric counterpart of the densest $k$-subhypergraph problem
  - studied recently in (APPROX’16, SODA’17), conditionally hard to approximate within $|V|^{1 - \epsilon}$
  - ranges $\mathcal{R}$ correspond to vertices of the hypergraph, points $P$ correspond to edges (defined by containment relation)

- With convex polygons, max-exposure is as hard as densest $k$-subhypergraph
  - Hypergraph $H = (X, E)$ can be transformed into max-exposure of convex ranges $\mathcal{R}$ and points $P$

What about rectangle ranges?

- NP-hard and also ‘conditionally’ hard to approximate within $O(n^{1/4})$ even when rectangles in $\mathcal{R}$ are translates of two fixed rectangles

$$n = |\mathcal{R}|$$
Hardness of Max Exposure

- Geometric counterpart of the *densest $k$-subhypergraph* problem
  - studied recently in (APPROX’16, SODA’17), conditionally hard to approximate within $|V|^{1-\epsilon}$
  - ranges $\mathcal{R}$ correspond to vertices of the hypergraph, points $P$ correspond to edges (defined by containment relation)

- With convex polygons, max-exposure is as hard as densest $k$-subhypergraph
  - Hypergraph $H = (X, E)$ can be transformed into max-exposure of convex ranges $\mathcal{R}$ and points $P$

What about rectangle ranges?

- NP-hard and also ‘conditionally’ hard to approximate within $O(n^{1/4})$ even when rectangles in $\mathcal{R}$ are translates of two fixed rectangles

Simple reduction from densest $k$-subgraph on bipartite graphs (*bipartite-DkS*)
Hardness of Max Exposure

- Geometric counterpart of the densest \( k \)-subhypergraph problem
  - studied recently in (APPROX’16, SODA’17), conditionally hard to approximate within \( |V|^{1-\epsilon} \)
  - ranges \( \mathcal{R} \) correspond to vertices of the hypergraph, points \( P \) correspond to edges (defined by containment relation)

- With convex polygons, max-exposure is as hard as densest \( k \)-subhypergraph
  - Hypergraph \( H = (X, E) \) can be transformed into max-exposure of convex ranges \( \mathcal{R} \) and points \( P \)

What about rectangle ranges?

- NP-hard and also ‘conditionally’ hard to approximate within \( O(n^{1/4}) \) even when rectangles in \( \mathcal{R} \) are translates of two fixed rectangles

\[
\begin{align*}
1 & \quad a \\
2 & \quad b \\
3 & \quad c
\end{align*}
\]

Simple reduction from densest \( k \)-subgraph on bipartite graphs (bipartite-DkS)

- Assuming Dense Vs Random conjecture, bipartite-DkS is hard to approximate within \( O(|V|^{1/4}) \)
Approximation Algorithms

Can we do somewhat better for arbitrary rectangles?

What happens if we only allow translates of a single rectangle?
Approximation Algorithms

Can we do somewhat better for arbitrary rectangles?

- A bicriteria $O(k)$-approximation for arbitrary rectangles
  - Expose at least $\Omega(1/k)$ of optimal points by removing $k^2$ rectangles
  - Approximation factor improves to $O(\sqrt{k})$ if rectangles have bounded aspect ratio

What happens if we only allow translates of a single rectangle?
Can we do somewhat better for arbitrary rectangles?

- A bicriteria $O(k)$-approximation for arbitrary rectangles
  - Expose at least $\Omega(1/k)$ of optimal points by removing $k^2$ rectangles
  - Approximation factor improves to $O(\sqrt{k})$ if rectangles have bounded aspect ratio

What happens if we only allow translates of a single rectangle?

- There exists a PTAS when $\mathcal{R}$ consists of translates of a single rectangle
  - Builds upon a polynomial time 2-approximation using shifting techniques
Approximation Algorithms

Can we do somewhat better for arbitrary rectangles?

- A bicriteria $O(k)$-approximation for arbitrary rectangles
  - Expose at least $\Omega(1/k)$ of optimal points by removing $k^2$ rectangles
  - Approximation factor improves to $O(\sqrt{k})$ if rectangles have bounded aspect ratio

What happens if we only allow translates of a single rectangle?

- There exists a PTAS when $\mathcal{R}$ consists of translates of a single rectangle
  - Builds upon a polynomial time 2-approximation using shifting techniques
  - Gives a constant approximation if ratio of smallest and longest sidelengths is bounded

rest of this talk
A Simple Bicriteria Approximation

The algorithm is essentially greedy:

\[ \mathcal{R}(p) = \text{set of ranges that contain point } p \]
A Simple Bicriteria Approximation

The algorithm is essentially greedy:

\[ \mathcal{R}(p) = \text{set of ranges that contain point } p \]

- Discard all points for which \(|\mathcal{R}(p)| > k\)
A Simple Bicriteria Approximation

The algorithm is essentially greedy:

\[ \mathcal{R}(p) = \text{set of ranges that contain point } p \]

- Discard all points for which \( |\mathcal{R}(p)| > k \)
- Partition \( P \) into a set \( \mathcal{G} \) of groups:
  
  each group is an equivalence class of points with same \( \mathcal{R}(p) \)
A Simple Bicriteria Approximation

The algorithm is essentially greedy:

- \( \mathcal{R}(p) = \text{set of ranges that contain point } p \)
- Discard all points for which \( |\mathcal{R}(p)| > k \)
- Partition \( P \) into a set \( \mathcal{G} \) of groups:
  - each group is an equivalence class of points with same \( \mathcal{R}(p) \)
- Sort groups in \( \mathcal{G} \) by decreasing size and return points in first \( k \) groups
A Simple Bicriteria Approximation

The algorithm is essentially greedy:

\[ \mathcal{R}(p) = \text{set of ranges that contain point } p \]

- Discard all points for which \( |\mathcal{R}(p)| > k \)
- Partition \( P \) into a set \( \mathcal{G} \) of groups:
  - Each group is an equivalence class of points with same \( \mathcal{R}(p) \)
- Sort groups in \( \mathcal{G} \) by decreasing size and return points in first \( k \) groups

Total deleted ranges is at most \( k \cdot \max |\mathcal{R}(p)| = k^2 \)
A Simple Bicriteria Approximation

The algorithm is essentially greedy:

\[ \mathcal{R}(p) = \text{set of ranges that contain point } p \]

- Discard all points for which \( |\mathcal{R}(p)| > k \)

- Partition \( P \) into a set \( \mathcal{G} \) of groups:
  
  each group is an equivalence class of points with same \( \mathcal{R}(p) \)

- Sort groups in \( \mathcal{G} \) by decreasing size and return points in first \( k \) groups

Total deleted ranges is at most \( k \cdot \max |\mathcal{R}(p)| = k^2 \)

\# of groups \( \mathcal{G}^* \) in optimal \( \leq \# \) of cells in arrangement of \( k \) rectangles \( \leq c \cdot k^2 \)
A Simple Bicriteria Approximation

The algorithm is essentially greedy:

\[ \mathcal{R}(p) = \text{set of ranges that contain point } p \]

- Discard all points for which \( |\mathcal{R}(p)| > k \)
- Partition \( P \) into a set \( \mathcal{G} \) of groups:
  - each group is an equivalence class of points with same \( \mathcal{R}(p) \)
- Sort groups in \( \mathcal{G} \) by decreasing size and return points in first \( k \) groups

Total deleted ranges is at most \( k \cdot \max |\mathcal{R}(p)| = k^2 \)

\# of groups \( \mathcal{G}^* \) in optimal \( \leq \) \# of cells in arrangement of \( k \) rectangles

\( \leq c \cdot k^2 \)

Holds for any polygon with \( O(1) \) complexity
Translates of a Single Rectangle
Translates of a Single Rectangle

First, scale the rectangles so that they become squares

Goal now is to compute max-exposure of unit square ranges

Does not change any point-rectangle containment
Translates of a Single Rectangle

First, scale the rectangles so that they become squares

Goal now is to compute max-exposure of unit square ranges

Consider an even simpler problem: all points lie inside a unit square
Translates of a Single Rectangle

First, scale the rectangles so that they become squares

Goal now is to compute max-exposure of unit square ranges

Consider an even simpler problem: all points lie inside a unit square

Roadmap

Within a unit square → Within a horizontal strip of unit width → PTAS
(polytime) (polytime) (shifting techniques)
⇒ 4-approximation ⇒ 2-approximation
Translates of a Single Rectangle

First, scale the rectangles so that they become squares

\[ \Rightarrow \]

Goal now is to compute max-exposure of unit square ranges

Consider an even simpler problem: all points lie inside a unit square

Roadmap

Within a unit square \((\text{polytime})\) \(\Rightarrow\) 4-approximation

Within a horizontal strip of unit width \(\Rightarrow\) PTAS \((\text{polytime})\)

\(\Rightarrow\) 2-approximation (shifting techniques)
Max-Exposure Within a Unit Square

Consider the dynamic programming formulation: DP-template-0
Max-Exposure Within a Unit Square

Consider the dynamic programming formulation: DP-template-0

- Process points in $P$ by increasing $x$-coordinates

Active ranges: ranges that have at least one corner to the right of $x = x_i$
Max-Exposure Within a Unit Square

Consider the dynamic programming formulation: **DP-template-0**

- Process points in \( P \) by increasing \( x \)-coordinates

Active ranges: ranges that have at least one corner to the right of \( x = x_i \)

Expose \( p_i \) ⇔ delete all ranges in \( \mathcal{R}(p_i) \)
Max-Exposure Within a Unit Square

Consider the dynamic programming formulation: \textbf{DP-template-0}

- Process points in $P$ by increasing $x$-coordinates

Exposure \( p_i \) $\iff$ delete all ranges in $\mathcal{R}(p_i)$

Active ranges: ranges that have at least one corner to the right of $x = x_i$

\[
S(i, k', \mathcal{R}_d) = \max \left\{ \begin{array}{ll}
\text{do not expose } p_i \\
\text{expose } p_i
\end{array} \right. 
\]
Max-Exposure Within a Unit Square

Consider the dynamic programming formulation: **DP-template-0**

- Process points in $P$ by increasing $x$-coordinates

Active ranges: ranges that have at least one corner to the right of $x = x_i$

$S(i, k', R_d) = \max \begin{cases} 
S(i + 1, k' - k_i, R_d \cup R(p_i)) + 1 & \text{expose } p_i \\
S(i + 1, k', R_d) & \text{do not expose } p_i 
\end{cases}$

# of ranges that can be deleted to right of $x = x_i$ $(0 \leq k' \leq k)$
Max-Exposure Within a Unit Square

Consider the dynamic programming formulation: **DP-template-0**

- Process points in $P$ by increasing $x$-coordinates

Active ranges: ranges that have at least one corner to the right of $x = x_i$

Set of active ranges that were already deleted

$S(i, k', \mathcal{R}_d) = \max \left\{ \begin{array}{ll}
\text{do not expose } p_i \\
\text{expose } p_i \\
\end{array} \right.$

Set of active ranges that were already deleted

# of ranges that can be deleted to right of $x = x_i \ (0 \leq k' \leq k)$

Expose $p_i \iff$ delete all ranges in $\mathcal{R}(p_i)$
Max-Exposure Within a Unit Square

Consider the dynamic programming formulation: DP-template-0

- Process points in $P$ by increasing $x$-coordinates

Active ranges: ranges that have at least one corner to the right of $x = x_i$

Expose $p_i \iff$ delete all ranges in $\mathcal{R}(p_i)$

Active ranges:

$$S(i, k', \mathcal{R}_d) = \max \left\{ \begin{array}{l}
\text{do not expose } p_i \\
\text{expose } p_i \\
\text{Set of active ranges that were already deleted} \\
\text{# of ranges that can be deleted to right of } x = x_i \ (0 \leq k' \leq k) \\
\text{Optimal solution: } S(0, k, \emptyset) \end{array} \right\}$$
Max-Exposure Within a Unit Square

Consider the dynamic programming formulation: **DP-template-0**

- Process points in $P$ by increasing $x$-coordinates

Expose $p_i \iff$ delete all ranges in $R(p_i)$

Active ranges: ranges that have at least one corner to the right of $x = x_i$

$$S(i, k', \mathcal{R}_d) = \max \left\{ \begin{array}{ll} S(i + 1, k', \mathcal{R}_d) & \text{do not expose } p_i \\ S(i + 1, k' - k_i, \mathcal{R}_d \cup R(p_i)) + 1 & \text{expose } p_i \end{array} \right.$$ 

Set of active ranges that were already deleted

# of ranges that can be deleted to right of $x = x_i$ ($0 \leq k' \leq k$)

Optimal solution: $S(0, k, \emptyset)$
Max-Exposure Within a Unit Square

Consider the dynamic programming formulation: **DP-template-0**

- Process points in $P$ by increasing $x$-coordinates

![Diagram showing the process of exposing points within a unit square.](image)

**Active ranges**: ranges that have at least one corner to the right of $x = x_i$

**Expose** $p_i \iff$ delete all ranges in $\mathcal{R}(p_i)$

$$S(i, k', \mathcal{R}_d) = \max \begin{cases} S(i + 1, k', \mathcal{R}_d) & \text{do not expose } p_i \\ S(i + 1, \ldots) & \text{expose } p_i \end{cases}$$

- Set of active ranges that were already deleted
- # of ranges that can be deleted to right of $x = x_i$ ($0 \leq k' \leq k$)

**Optimal solution**: $S(0, k, \emptyset)$
Max-Exposure Within a Unit Square

Consider the dynamic programming formulation: DP-template-0

– Process points in $P$ by increasing $x$-coordinates

Active ranges: ranges that have at least one corner to the right of $x = x_i$

$$S(i, k', R_d) = \max \begin{cases} S(i + 1, k', R_d) & \text{do not expose } p_i \\ S(i + 1, k', R_d \cup R(p_i)) + 1 & \text{expose } p_i \end{cases}$$

Set of active ranges that were already deleted

# of ranges that can be deleted to right of $x = x_i$ ($0 \leq k' \leq k$)

Optimal solution: $S(0, k, \emptyset)$
Max-Exposure Within a Unit Square

Consider the dynamic programming formulation: DP-template-0

- Process points in $P$ by increasing $x$-coordinates

Active ranges: ranges that have at least one corner to the right of $x = x_i$

\[
S(i, k', R_d) = \max \begin{cases} 
S(i + 1, k', R_d) & \text{do not expose } p_i \\
S(i + 1, k' - k_i, R_d \cup R(p_i)) + 1 & \text{expose } p_i
\end{cases}
\]

Expose $p_i \iff$ delete all ranges in $R(p_i)$

Optimal solution: $S(0, k, \emptyset)$
Max-Exposure Within a Unit Square

How do we keep track of deleted range set $\mathcal{R}_d$ using polynomial space?
Max-Exposure Within a Unit Square

How do we keep track of deleted range set $\mathcal{R}_d$ using polynomial space?

**Type-0**: Unit square ranges that intersect $x = 0$
Max-Exposure Within a Unit Square

How do we keep track of deleted range set $\mathcal{R}_d$ using polynomial space?

Type-0: Unit square ranges that intersect $x = 0$
Type-1: Unit square ranges that intersect $x = 1$
Max-Exposure Within a Unit Square

How do we keep track of deleted range set $\mathcal{R}_d$ using polynomial space?

Type-0: Unit square ranges that intersect $x = 0$
Type-1: Unit square ranges that intersect $x = 1$

Suppose we only had Type-0 ranges:

$R_3$ is ‘anchored’ to $\ell_0$
Max-Exposure Within a Unit Square

How do we keep track of deleted range set $\mathcal{R}_d$ using polynomial space?

**Type-0**: Unit square ranges that intersect $x = 0$

**Type-1**: Unit square ranges that intersect $x = 1$

Suppose we only had Type-0 ranges:

$q_0 = \text{Exposed point to left of } x = x_i \text{ closest to } \ell_0$

$R_3$ is ‘anchored’ to $\ell_0$  
$\Rightarrow$ must contain $q_0$
Max-Exposure Within a Unit Square

How do we keep track of deleted range set \( R_d \) using polynomial space?

**Type-0**: Unit square ranges that intersect \( x = 0 \)

**Type-1**: Unit square ranges that intersect \( x = 1 \)

Suppose we only had Type-0 ranges:

- \( q_0 = \) Exposed point to left of \( x = x_i \) closest to \( \ell_0 \)
- \( q_1 = \) Exposed point to left of \( x = x_i \) closest to \( \ell_1 \)

\( R_3 \) is ‘anchored’ to \( \ell_0 \) \( \Rightarrow \) must contain \( q_0 \)
Max-Exposure Within a Unit Square

How do we keep track of deleted range set $\mathcal{R}_d$ using polynomial space?

**Type-0**: Unit square ranges that intersect $x = 0$

**Type-1**: Unit square ranges that intersect $x = 1$

Suppose we only had Type-0 ranges:

$q_0 =$ Exposed point to left of $x = x_i$ closest to $\ell_0$

$q_1 =$ Exposed point to left of $x = x_i$ closest to $\ell_1$

$$\mathcal{R}_d = \mathcal{R}(q_0) \cup \mathcal{R}(q_1)$$

$R_3$ is ‘anchored’ to $\ell_0$  
$\Rightarrow$ must contain $q_0$
Max-Exposure Within a Unit Square

How do we keep track of deleted range set $\mathcal{R}_d$ using polynomial space?

**Type-0**: Unit square ranges that intersect $x = 0$

**Type-1**: Unit square ranges that intersect $x = 1$

Suppose we only had Type-0 ranges:

$q_0 = \text{Exposed point to left of } x = x_i \text{ closest to } \ell_0$

$q_1 = \text{Exposed point to left of } x = x_i \text{ closest to } \ell_1$

$$\mathcal{R}_d = \mathcal{R}(q_0) \cup \mathcal{R}(q_1)$$

Can keep track of Type-0 deleted ranges by remembering $q_0, q_1$
Handling Type-1 Ranges

Need an alternative dynamic programming formulation: **DP-template-1**
Handling Type-1 Ranges

Need an alternative dynamic programming formulation: \textbf{DP-template-1}

- Process ‘events’ in $P$ by increasing $x$-coordinates $x_i$
Handling Type-1 Ranges

Need an alternative dynamic programming formulation: **DP-template-1**

- Process ‘events’ in $P$ by increasing $x$-coordinates $x_i$

Diagram: 
- Begin-range events
- Point events
- Active Points: with $x$-coordinates $\geq x_i$
Handling Type-1 Ranges

Need an alternative dynamic programming formulation: **DP-template-1**

– Process ‘events’ in $P$ by increasing $x$-coordinates $x_i$

\[
S(i, k', P_f) = \max \begin{cases} 
S(i + 1, k' - 1, P_f) & \text{delete range } R_i \\
S(i + 1, k', P_f \cup P(R_i)) & \text{do not delete } R_i \\
S(i + 1, k', P_f) & \text{if } p_i \in P_f, \text{ cannot expose } p_i \\
S(i + 1, k', P_f) + 1 & \text{otherwise, expose } p_i
\end{cases}
\]

Maintain set of **forbidden points** $P_f$

active points that lie in a range that was not deleted

**Active Points**: with $x$-coordinates $\geq x_i$

Optimal solution: $S(0, k, \emptyset)$
Handling Type-1 Ranges

Need an alternative dynamic programming formulation: **DP-template-1**

- Process ‘events’ in $P$ by increasing $x$-coordinates $x_i$

Active Points: with $x$-coordinates $\geq x_i$

Maintain set of **forbidden points** $P_f$
active points that lie in a range that was not deleted

Optimal solution: $S(0, k, \emptyset)$

$$S(i, k', P_f) = \max \begin{cases} 
S(i + 1, k' - 1, P_f) & \text{delete range } R_i \\
S(i + 1, k', P_f \cup P(R_i)) & \text{do not delete } R_i
\end{cases}$$
Handling Type-1 Ranges

Need an alternative dynamic programming formulation: **DP-template-1**

- Process ‘events’ in $P$ by increasing $x$-coordinates $x_i$

$$S(i, k', P_f) = \max \begin{cases} 
  S(i + 1, k' - 1, P_f) & \text{delete range } R_i \\
  S(i + 1, k', P_f \cup P(R_i)) & \text{do not delete } R_i
\end{cases}$$

**Active Points**: with $x$-coordinates $\geq x_i$

**Maintain set of forbidden points $P_f$**: active points that lie in a range that was not deleted

Optimal solution: $S(0, k, \emptyset)$
Handling Type-1 Ranges

Need an alternative dynamic programming formulation: **DP-template-1**

- Process ‘events’ in $P$ by increasing $x$-coordinates $x_i$

**Active Points**: with $x$-coordinates $\geq x_i$

**Maintain set of forbidden points** $P_f$

active points that lie in a range that was not deleted

\[
S(i, k', P_f) = \max \begin{cases} 
S(i + 1, k' - 1, P_f) & \text{delete range } R_i \\
S(i + 1, k', P_f \cup P(R_i)) & \text{do not delete } R_i
\end{cases}
\]

\[
= \max \begin{cases} 
S(i + 1, k', P_f) & \text{if } p_i \in P_f, \text{ cannot expose } p_i \\
S(i + 1, k', P_f) + 1 & \text{otherwise, expose } p_i
\end{cases}
\]

Optimal solution: $S(0, k, \emptyset)$
Handling Type-1 Ranges

How do we keep track of forbidden points $P_f$ using polynomial space?

$Q_0 = \text{Undeleted range to left of } x = x_i \text{ farthest from } \ell_0$
Handling Type-1 Ranges

How do we keep track of forbidden points $P_f$ using polynomial space?

$Q_0 = $ Undeleted range to left of $x = x_i$ farthest from $\ell_0$
$Q_1 = $ Undeleted range to left of $x = x_i$ farthest from $\ell_1$

$P_f = P(Q_0) \cup P(Q_1)$

if $p \in P_f$, then $p$ must lie in either $Q_0$ or $Q_1$
Handling Type-1 Ranges

How do we keep track of forbidden points $P_f$ using polynomial space?

$Q_0 =$ Undeleted range to left of $x = x_i$ farthest from $\ell_0$
$Q_1 =$ Undeleted range to left of $x = x_i$ farthest from $\ell_1$

\[ P_f = P(Q_0) \cup P(Q_1) \]

Can keep track of forbidden points by remembering $Q_0, Q_1$
Handling Type-1 Ranges

How do we keep track of forbidden points $P_f$ using polynomial space?

$Q_0$ = Undeleted range to left of $x = x_i$ farthest from $\ell_0$
$Q_1$ = Undeleted range to left of $x = x_i$ farthest from $\ell_1$

$$P_f = P(Q_0) \cup P(Q_1)$$

if $p \in P_f$, then $p$ must lie in either $Q_0$ or $Q_1$

Can keep track of forbidden points by remembering $Q_0, Q_1$

Combine **DP-template-0** and **DP-template-1** to solve within a unit square:

Subproblems defined as: $S(i, k', q_0, q_1, Q_0, Q_1)$

updated appropriately at begin-range and point events
In Summary:
In Summary:

- Max-exposure: to expose maximum points by deleting $k$ ranges
In Summary:

- Max-exposure: to expose maximum points by deleting $k$ ranges
- Hard to approximate – even with restricted rectangular ranges
In Summary:

- Max-exposure: to expose maximum points by deleting $k$ ranges
- Hard to approximate – even with restricted rectangular ranges
- Exhibits a PTAS for unit-square ranges
  - Gives a constant approximation for rectangles if ratio of smallest and longest sidelengths is bounded
In Summary:

- Max-exposure: to expose maximum points by deleting $k$ ranges
- Hard to approximate – even with restricted rectangular ranges
- Exhibits a PTAS for unit-square ranges
  - Gives a constant approximation for rectangles if ratio of smallest and longest sidelengths is bounded
- Bi-criteria $O(k)$-approximation algorithm for rectangles, $O(\sqrt{k})$ for squares
In Summary:

- Max-exposure: to expose maximum points by deleting $k$ ranges
- Hard to approximate – even with restricted rectangular ranges
- Exhibits a PTAS for unit-square ranges
  - Gives a constant approximation for rectangles if ratio of smallest and longest sidelengths is bounded
- Bi-criteria $O(k)$-approximation algorithm for rectangles, $O(\sqrt{k})$ for squares
- Does there exist a constant approximation for arbitrary squares?
In Summary:

- Max-exposure: to expose maximum points by deleting $k$ ranges
- Hard to approximate – even with restricted rectangular ranges
- Exhibits a PTAS for unit-square ranges
  - Gives a constant approximation for rectangles if ratio of smallest and longest sidelengths is bounded
- Bi-criteria $O(k)$-approximation algorithm for rectangles, $O(\sqrt{k})$ for squares
- Does there exist a constant approximation for arbitrary squares?

Thanks!
Backup: Combined DP

\[ S(i, k', q_0, q_1, Q_0, Q_1) \]
Backup: Combined DP

\[ S(i, k', q_0, q_1, Q_0, Q_1) \]

\[
= \max \left\{ \begin{array}{ll}
S(i + 1, k', q_0, q_1, Q_0, Q_1) & \text{if } p_i \in P_f, \text{ cannot expose } p_i \\
S(i + 1, k', q_0, q_1, Q_0, Q_1) & \text{choose to not expose } p_i \\
S(i + 1, k' - k_i, \text{closer}(p_i, q_0), \text{closer}(p_i, q_1), Q_0, Q_1) + 1 & \text{otherwise, expose } p_i
\end{array} \right. 
\]
Backup: Combined DP

\[ S(i, k', q_0, q_1, Q_0, Q_1) \]

\[
= \max \begin{cases} 
S(i + 1, k', q_0, q_1, Q_0, Q_1) & \text{if } p_i \in Pf, \text{ cannot expose } p_i \\
S(i + 1, k', q_0, q_1, Q_0, Q_1) & \text{choose to not expose } p_i \\
S(i + 1, k' - k_i, \text{closer}(p_i, q_0), \text{closer}(p_i, q_1), Q_0, Q_1) + 1 & \text{otherwise, expose } p_i
\end{cases}
\]

\[
= \max \begin{cases} 
S(i + 1, k' - 1, q_0, q_1, Q_0, Q_1) & \text{delete Type-1 range } R_i \\
S(i + 1, k', q_0, q_1, \text{farther}(R_i, Q_0), Q_1) & R_i \text{ is not deleted and anchored to } \ell_0 \\
S(i + 1, k', q_0, q_1, Q_0, \text{farther}(R_i, Q_1)) & R_i \text{ is not deleted and anchored to } \ell_1
\end{cases}
\]