DOMINATING SET, DISCRETE k-CENTER, INDEPENDENT SET AND DISPERSION PROBLEMS FOR PLANAR POINTS IN CONVEX POSITION

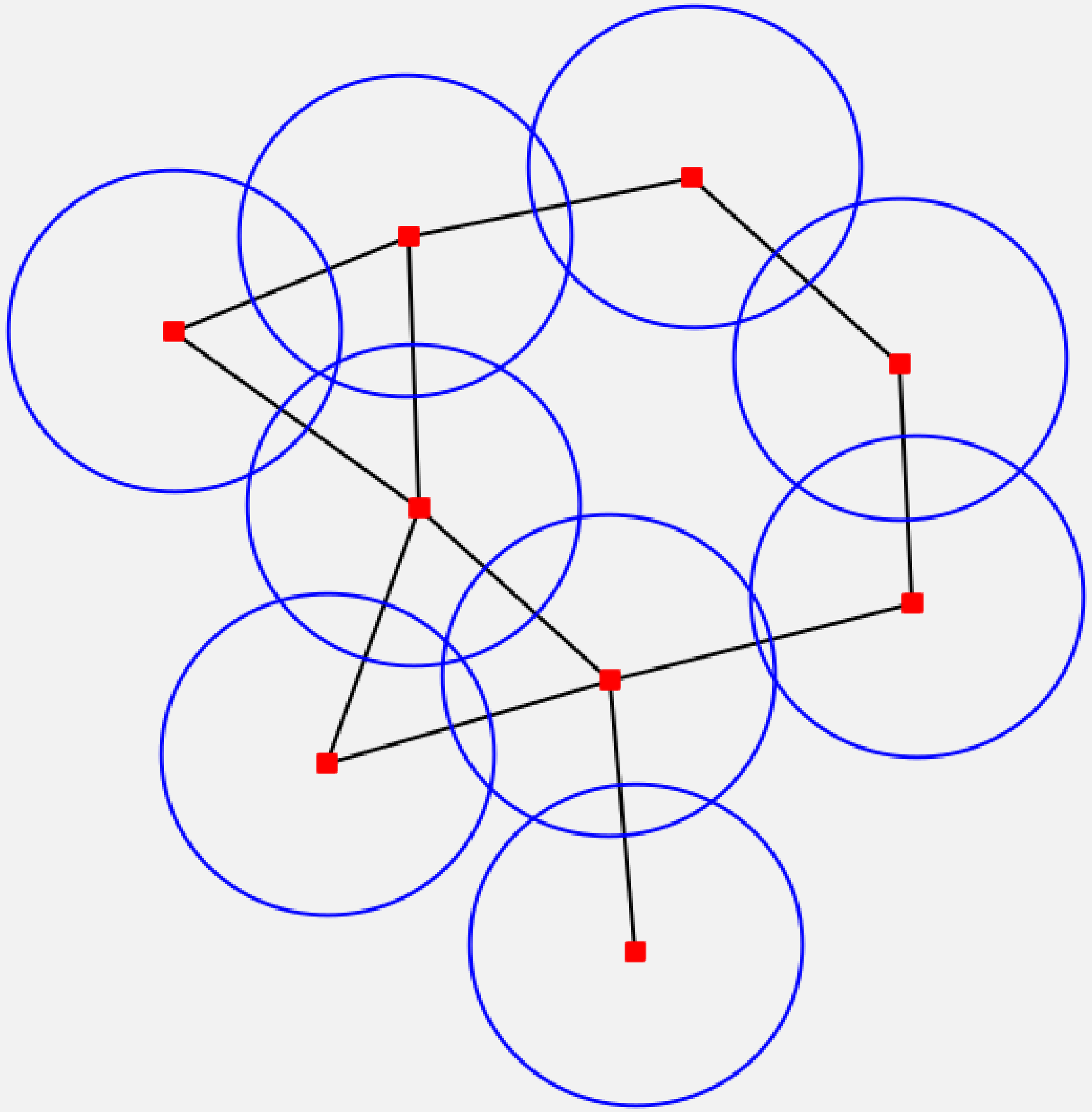
Anastasiia Tkachenko and Haitao Wang
The University of Utah, USA

THE UNIVERSITY OF UTAH®

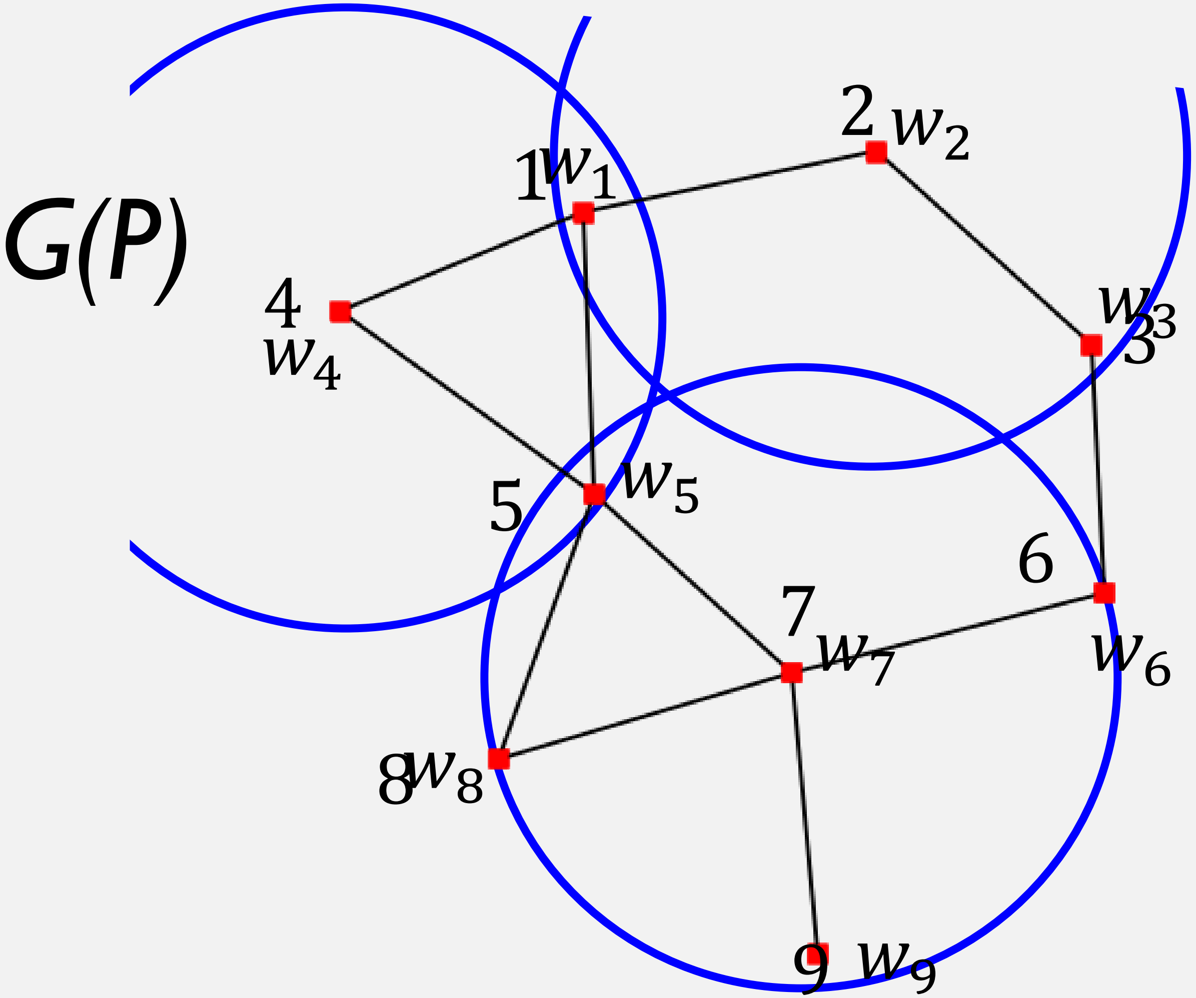
UNIT-DISK GRAPHS

P

G(P)



MINIMUM-WEIGHT DOMINATING SET PROBLEM



Dominating set example: {2, 4, 7}

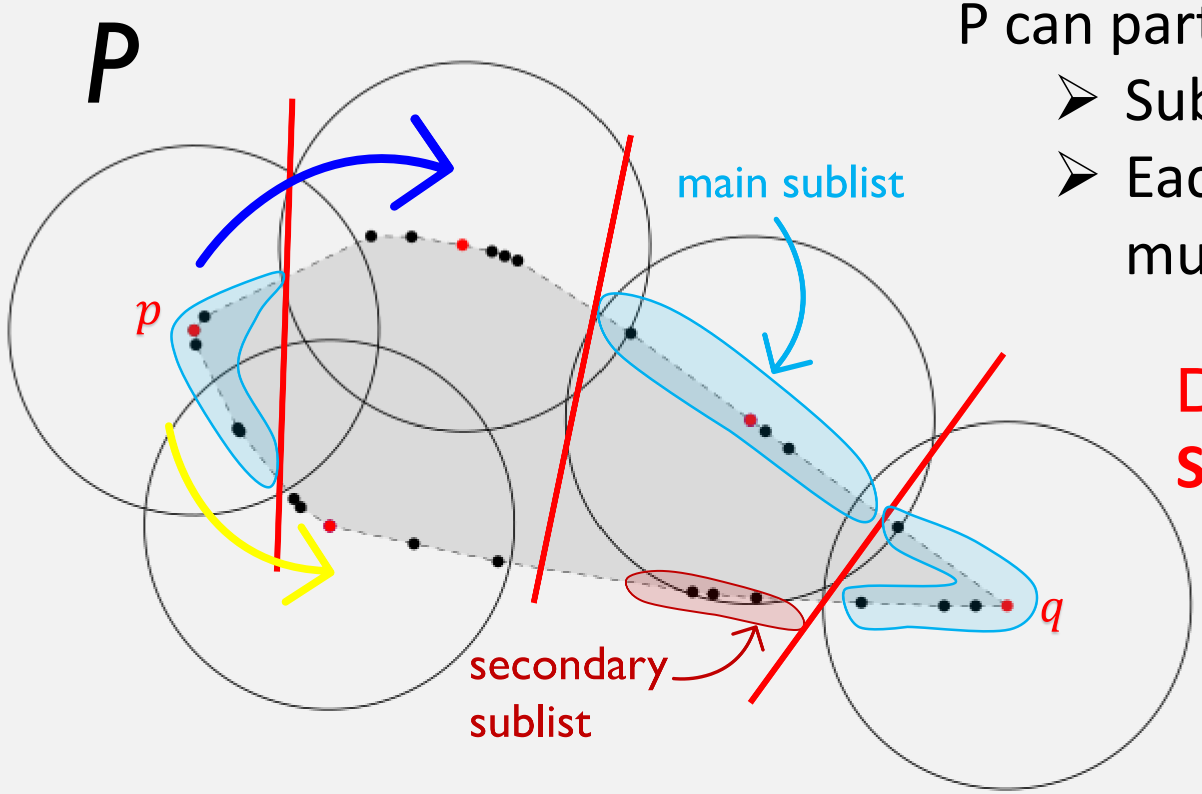
Problem statement: Given P, find a dominating set of $\mathrm{G}(P)$ with minimum total vertex weight.

NP-hard in the general case.

Our results:

- Convex position, *weighted*: $O(n^3 \log^2 n)$
- Convex position, *unweighted*: $O(n^2 \log n)$

STRUCTURAL PROPERTIES OF POINTS IN CONVEX POSITION



Line Separability Property :

P can partition into |**S**| subsets such that:

- Subsets are line separable and
- Each consists of at most 2 sublists, where one must be **main.**

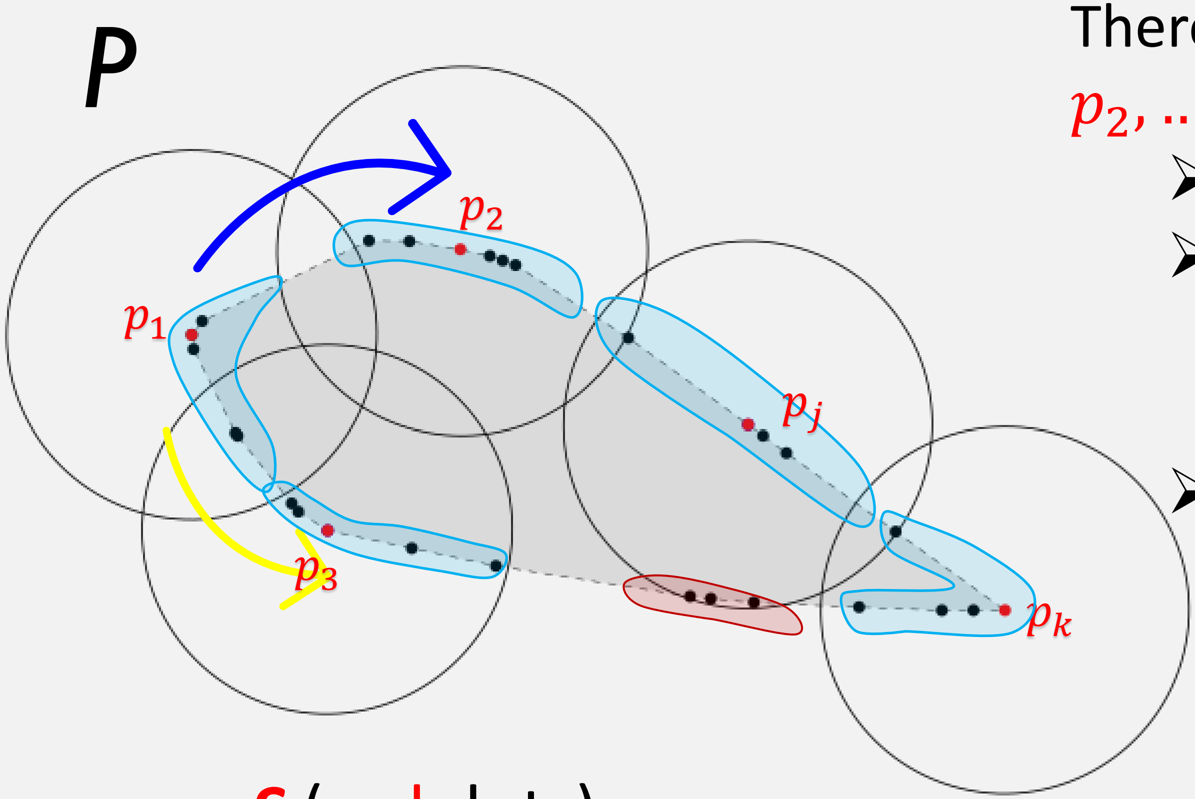
Decoupling pair Property :

S contains 2 centers p, q:

- Subsets for p and q are **main** sublists and
- For other subsets with 2 sublists, one is on the upper hull, the other is on the lower hull.

S (red dots): dominating set

STRUCTURAL PROPERTIES OF POINTS IN CONVEX POSITION



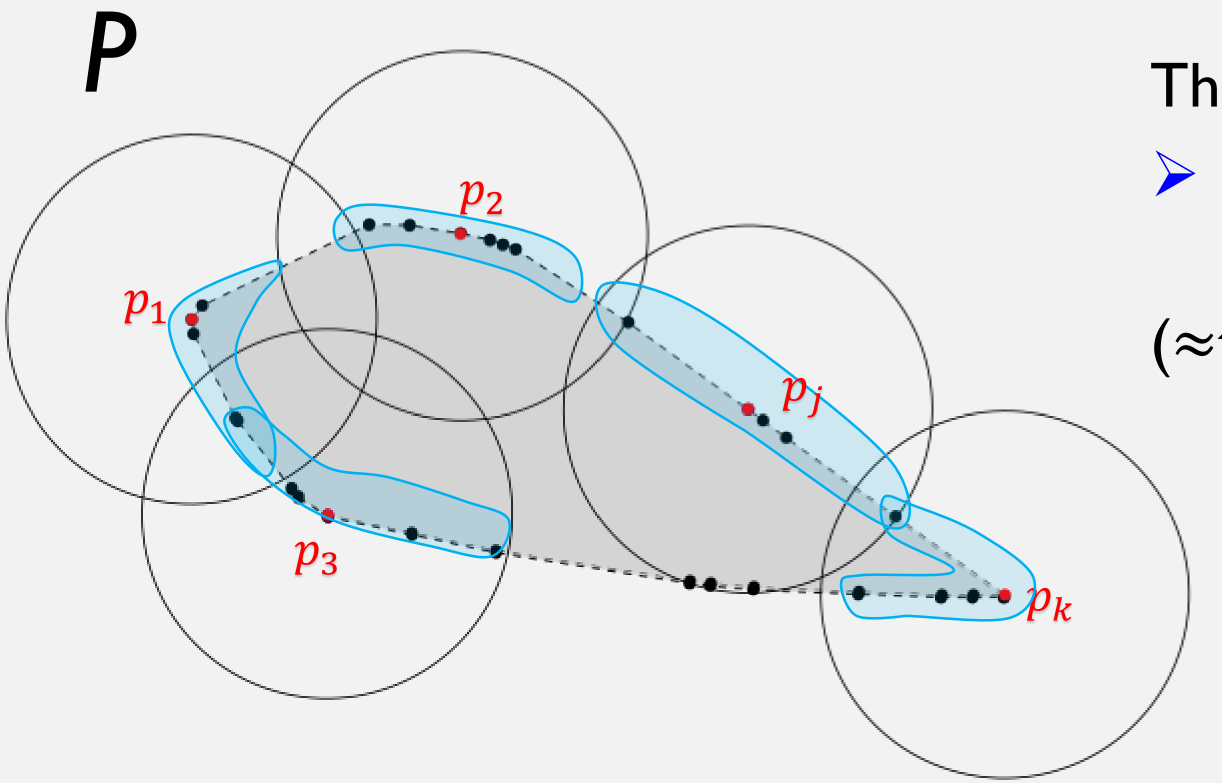
S (red dots)

Ordering Property :

There exists an ordering of the centers of **S** as p_1, $p_2, \ldots, p_k$ such that:

- p_1 (resp., p_k) is only assigned one sublist
- if p_j , 1 < j < k, is assigned two sublists, then one of them is on upper hull, the other is on the lower
- The order of the centers of the sublists along upper (resp., lower) hull from p_1 to p_k is a subsequence of the above ordering.

ALGORITHM



The 1st iteration:

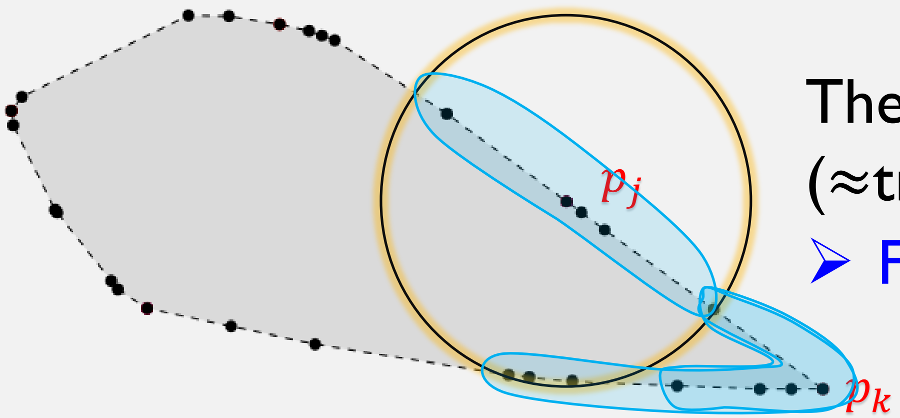
➢ For each p_j in P, compute *main* sublists of the weight w_j

(≈treat p_j as the 1st center in the *order*)

→ n sublists in O(nlogn)

ALGORITHM

P



The 1st iteration:

- For each p_j in P, compute *main* sublists of the weight w_j

(≈treat p_j as the 1st center in the *order*)

→ n sublists in O(nlogn)

The 2nd iteration:

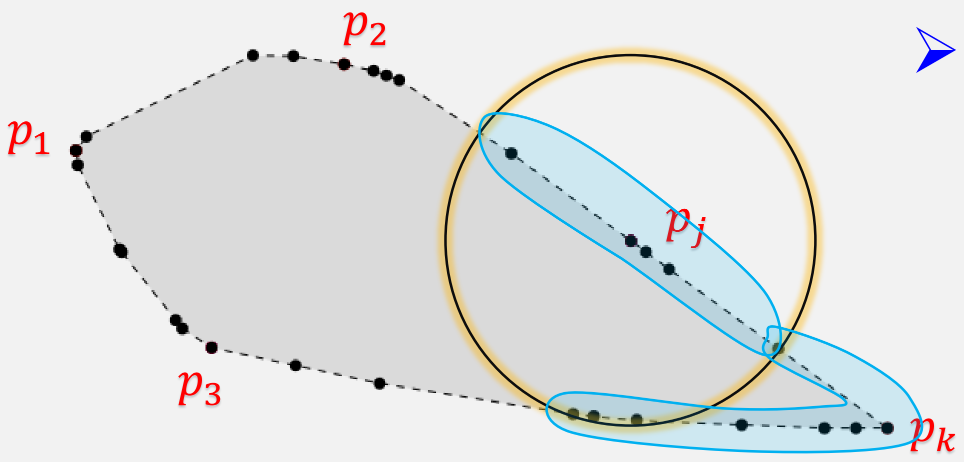
(≈treat p_j as the 2^{nd} center in the *order*)

- For each p_j in P:
  - Find the minimum-weight sublists computed at the 1^{st} iteration containing the 1^{st} (2^{nd}, …, n^{th}) point outside of the disk clockwise and counterclockwise
    - Extend a sublist by consecutive points covered by the disk

→ n^2 sublists in $O(n^2 \log^2 n)$

ALGORITHM

P



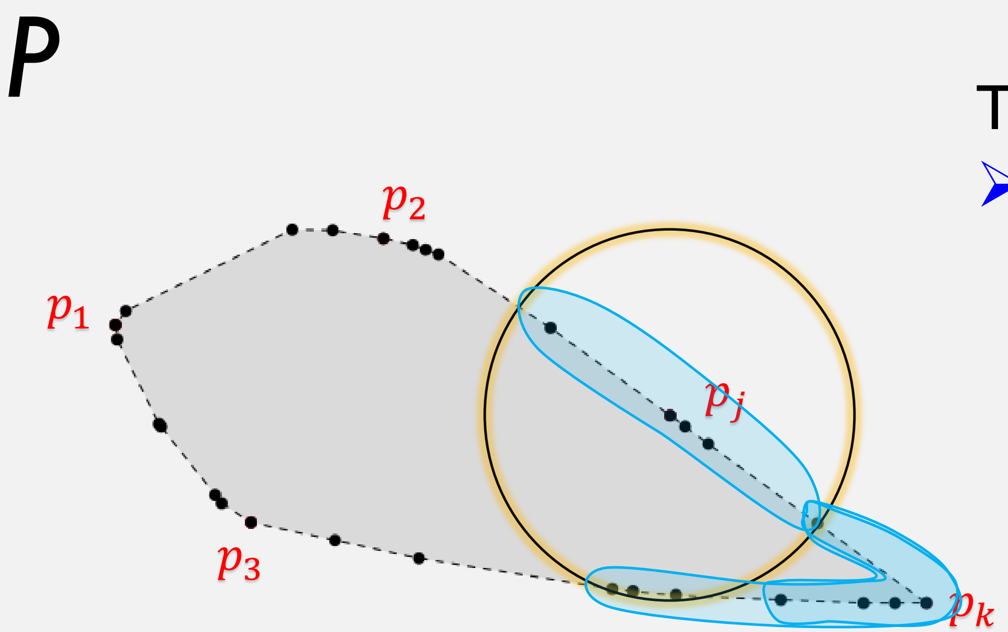
The k^{th} iteration:

- For each p_j in P:
  - Find the minimum-weight sublists computed at the $(k\text{-}1)^{th}$ iteration containing the 1^{st} (2^{nd}, …, n^{th}) point outside of the disk clockwise and counterclockwise
    - Extend a sublist by consecutive points covered by the disk

→ $O(n^2\ k\ \log^2 n)$ in total

(≈treat each point as the k^{th} center in the *order*)

ALGORITHM: UNWEIGHTED CASE



The k^th iteration:

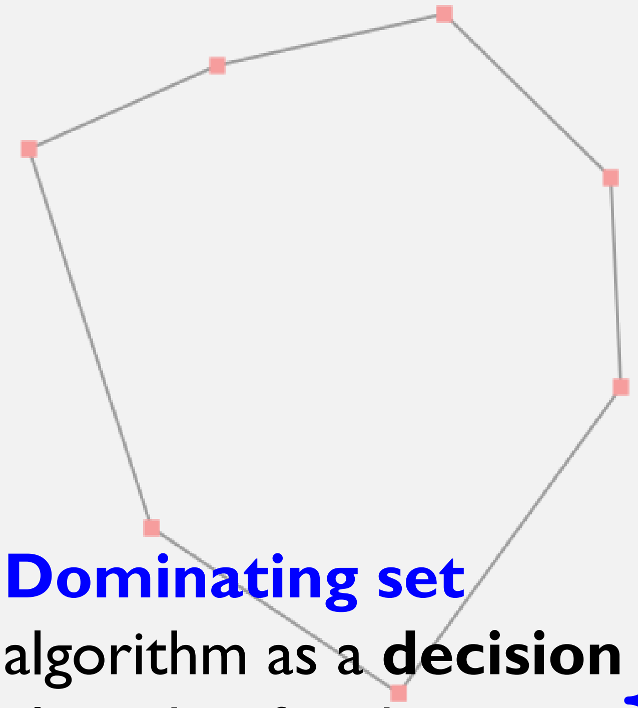
- For each p_j in P:
  - Find a sublist computed at the (k-1)^th iteration containing the farthest clockwise (and the farthest counterclockwise) point outside of the disk
    - Extend the sublist by consecutive points covered by the disk

→ O(n k logn) in total

DISCRETE k-CENTER PROBLEM

Definition: Given a set P of n points and an integer k, compute a subset of k points in P (centers) such that the maximum distance between any point in P and its nearest center is minimized.

NP-hard for arbitrary position of P!



Previous results:

- Convex position, continuous: $O(\min\{k, \log n\} n^2 \log n + k^2 n \log n)$; *Choi, Lee, and Ahn , 2023*
- Arbitrary position, continuous, k = 2: $O(n \log n)$; *Cho, Oh, Wang, and Xue, 2024*
- Arbitrary position, discrete, k = 2: $O(n^{4/3} \log^5 n)$; *Agarwal, Sharir, and Welzl, 1998*

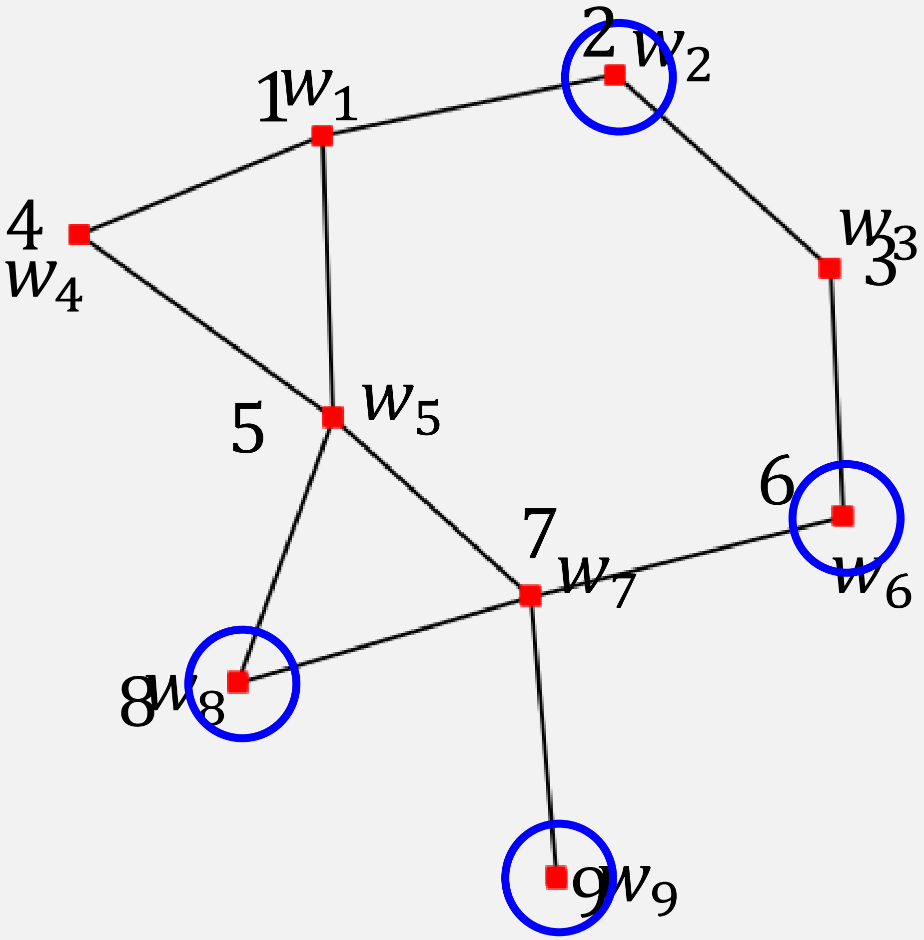
Dominating set algorithm as a **decision** algorithm for discrete k-center problem

Our result:

- Convex position, discrete: $O(\min\{n^{4/3} \log n + kn \log^2 n , k^2 n \log^2 n\})$ using distance selection or parametric search technique.

MAXIMUM-WEIGHT INDEPENDENT SET PROBLEM

$G(P)$

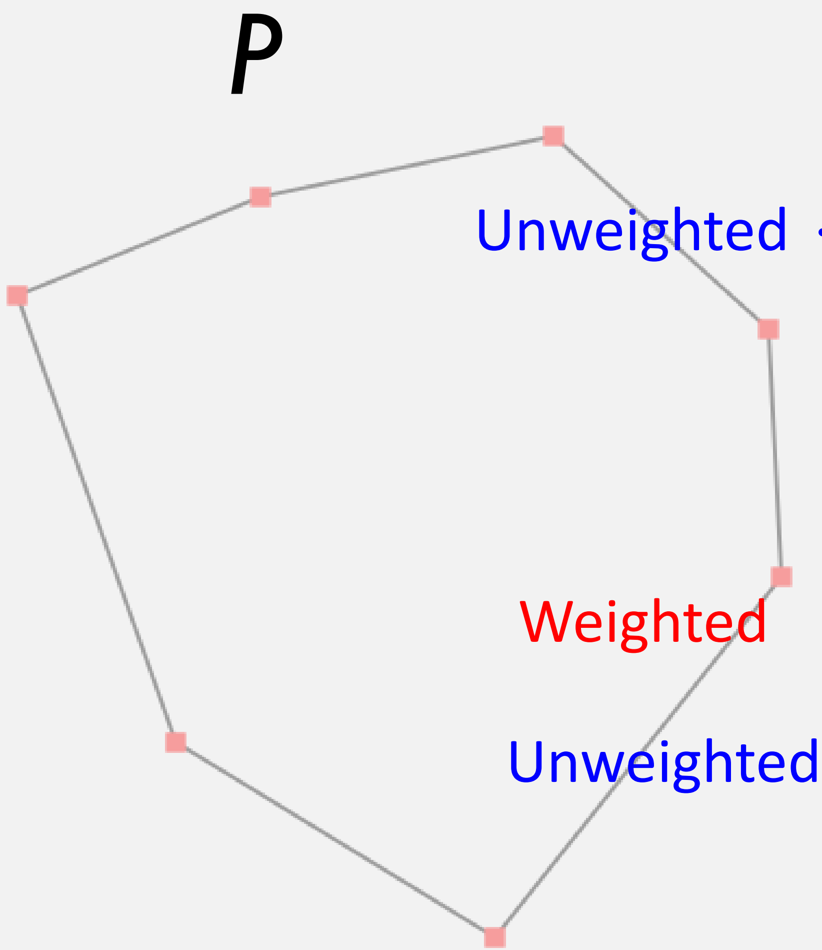


Independent set example: {2, 8, 6, 9}

Problem statement: Given P, find an independent subset of $\mathrm{G}(P)$ whose total vertex weight is maximized.

NP-hard for unit disk graphs!

PREVIOUS AND OUR RESULTS



Previous results:

Unweighted:

- Convex position: $O(n^6 \log n)$
  Singireddy, Basappa, and Mitchell, 2023
- Arbitrary position, independent set of size 3: $O(n^{4/3} \log^2 n)$
  Agarwal, Overmars, and Sharir, 2006

Our results:

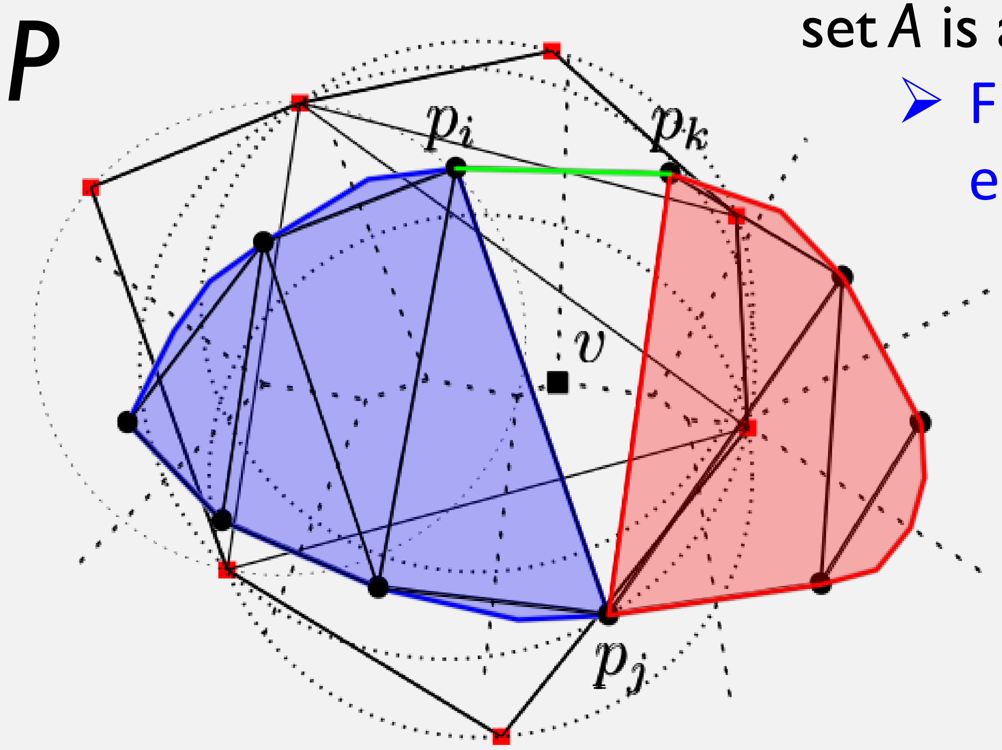
Weighted:

- Convex position: $O\left(n^{7/2}\right)$
- Arbitrary position, independent set of size 3: $O\left(n^{\frac{5}{3}+\delta}\right)$

Unweighted:

- Convex position, independent set of size 3: $O(n \log n)$

OBSERVATIONS FROM PREVIOUS WORK (*SBM'23*)



S (black dots)
Delaunay triangulation

Property of Delaunay triangulation: Closest pair of points in a set *A* is an edge of its Delaunay triangulation.

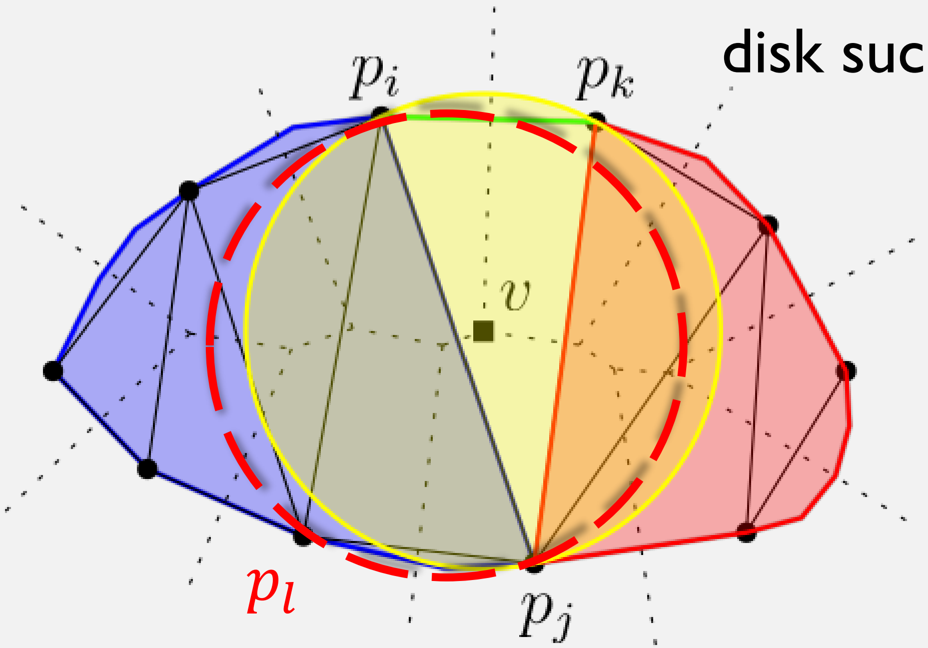
- Find a maximum-weight subset $S \subseteq P$ with the shortest edge of its Delaunay triangulation larger than 1.

Let p_i, p_j, p_k *be* a triangle in Delaunay triangulation of S.

- Delaunay triangulation of S does not contain an edge with vertices of different color!
  - Search for other vertices of S in blue, red and green portions independently!
  - A dynamic program based on the above

ALGORITHM: DYNAMIC PROGRAM

P



S (black dots)

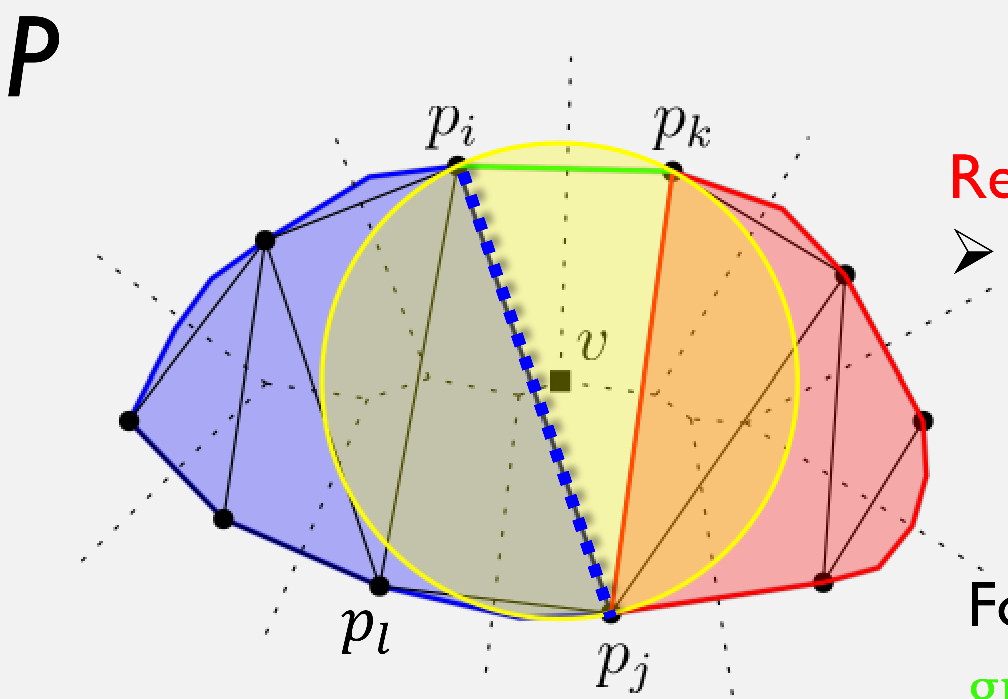
What is the subproblem?

- Maximum total weight of a subset P' of the blue outside yellow disk such that $P' \cup \{p_i, p_j\}$ is an independent set – $f(i,j,k)$

What is the recursive relation?

- $f(i,j,k) = \max_{\substack{p_l \in \text{blue} \\ p_l \notin \text{disk}}} (f(i,l,j) + f(l,j,i) + w_l)$

SUBPROBLEM IMPLEMENTATION



S (black dots)

Straightforward implementation: $O(n^4)$

- Can we do better? Yes, using disk range queries!

Recursive relation:

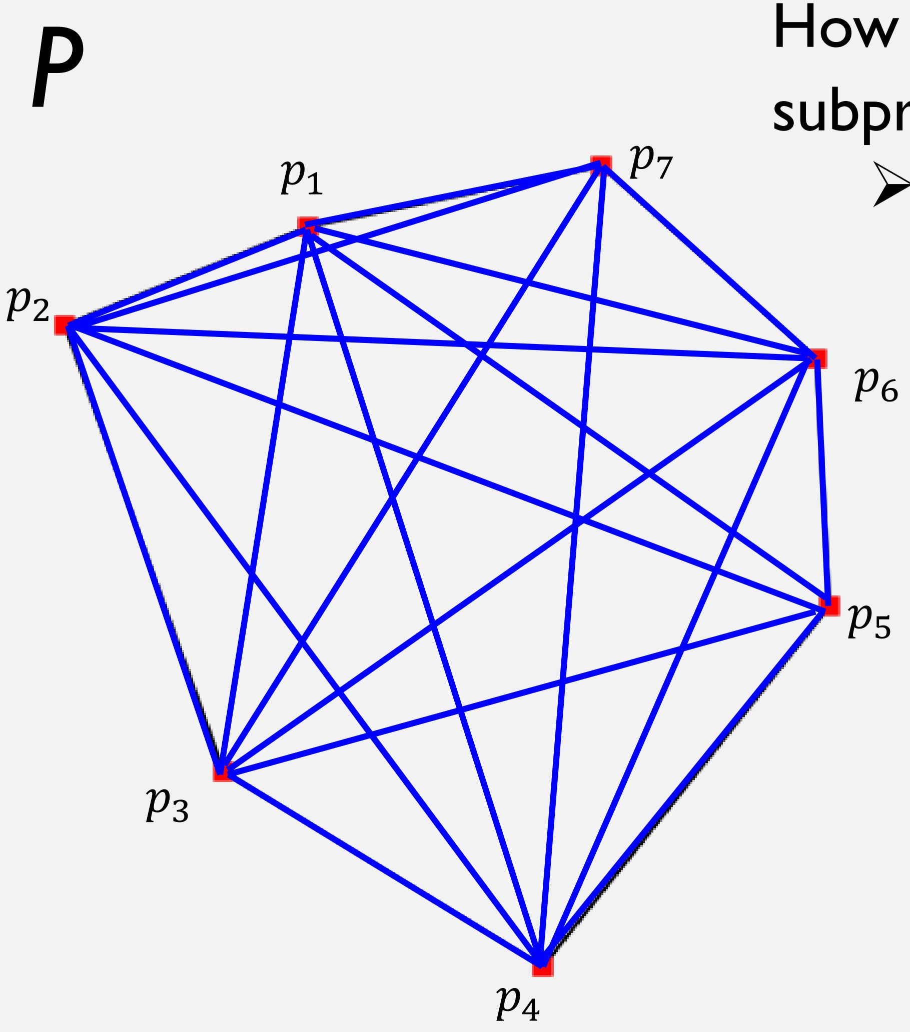
➢ $f(i,j,k) = \max_{\substack{p_l \in \text{blue} \\ p_l \notin \text{disk}}} \underbrace{(f(i,l,j) + f(l,j,i) + w_l)}_{\text{Cost of } p_l}$

For every disk through p_i, p_j and $p_k \in \text{red} \cup \text{green}$, find max-cost point outside the disk.

➢ Offline outside-disk range max-cost query!

➢ Can be solved in $O\left(n^{3/2}\right)$ for all p_k using cuttings

SUBPROBLEM IMPLEMENTATION



How to process diagonals $p_i p_j$ so that corresponding subproblems are already computed?

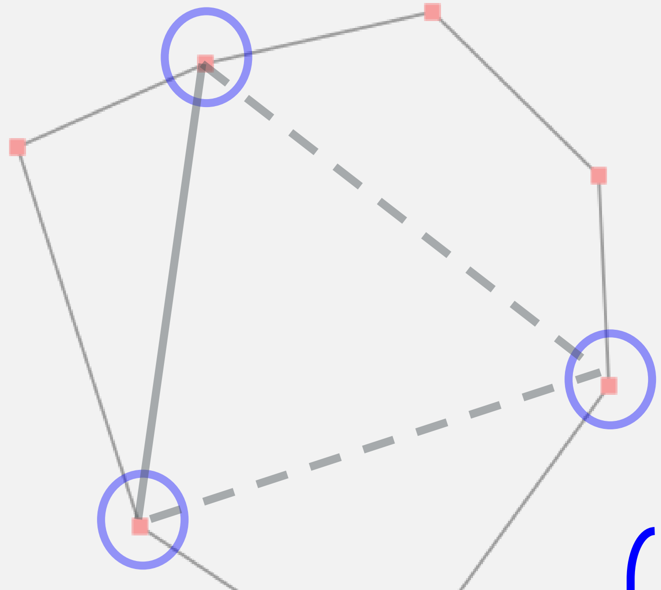
- for each $j = 2, \ldots, n$ and $i = j-1, j-2, \ldots, 1$ process each $p_i p_j$ in $O\left(n^{3/2}\right)$

$O(n^2)$ diagonals $p_i p_j \Rightarrow$ Total time is $O\left(n^{7/2}\right)$

DISPERSION PROBLEM

Definition: Given a set P of n points and an integer k, find a subset of k points from P such that the minimum pairwise distance of the points in the subset is maximized.

NP-hard for arbitrary position of P!



Previous results:

- Convex position: $O(n^4 k^2)$; *Singireddy, Basappa, and Mitchell, 2023*
- Arbitrary position, k = 3: $O(n^{4/3} \log^3 n)$; *Agarwal, Overmars, and Sharir, 2006*

Maximum independent set algorithm as a **decision** algorithm for dispersion problem

Our results:

- Convex position: $O(n^{7/2} \log n)$ using distance selection
- Convex position, k = 3:
  - $O(n \log^2 n)$ using parametric search
  - $O(n \log n)$ randomized, Chan's technique

THANK YOU FOR YOUR ATTENTION