# On Efficient Spatial Matching

Raymond Chi-Wing Wong (the Chinese University of Hong Kong)
Yufei Tao (the Chinese University of Hong Kong)
Ada Wai-Chee Fu (the Chinese University of Hong Kong)
Xiaokui Xiao (the Chinese University of Hong Kong)

Presented by Raymond Chi-Wing Wong
Presented by Raymond Chi-Wing Wong

# Outline

1. Introduction
   - Related work – Bichromatic Reverse Nearest Neighbor
2. Problem
   - Spatial Matching Problem (SPM)
     - Unweighted SPM
     - Weighted SPM
3. Algorithm
   - Chain (for unweighed SPM)
   - Weighted Chain (for weighted SPM)
4. Empirical Study
5. Conclusion

# 1. Introduction

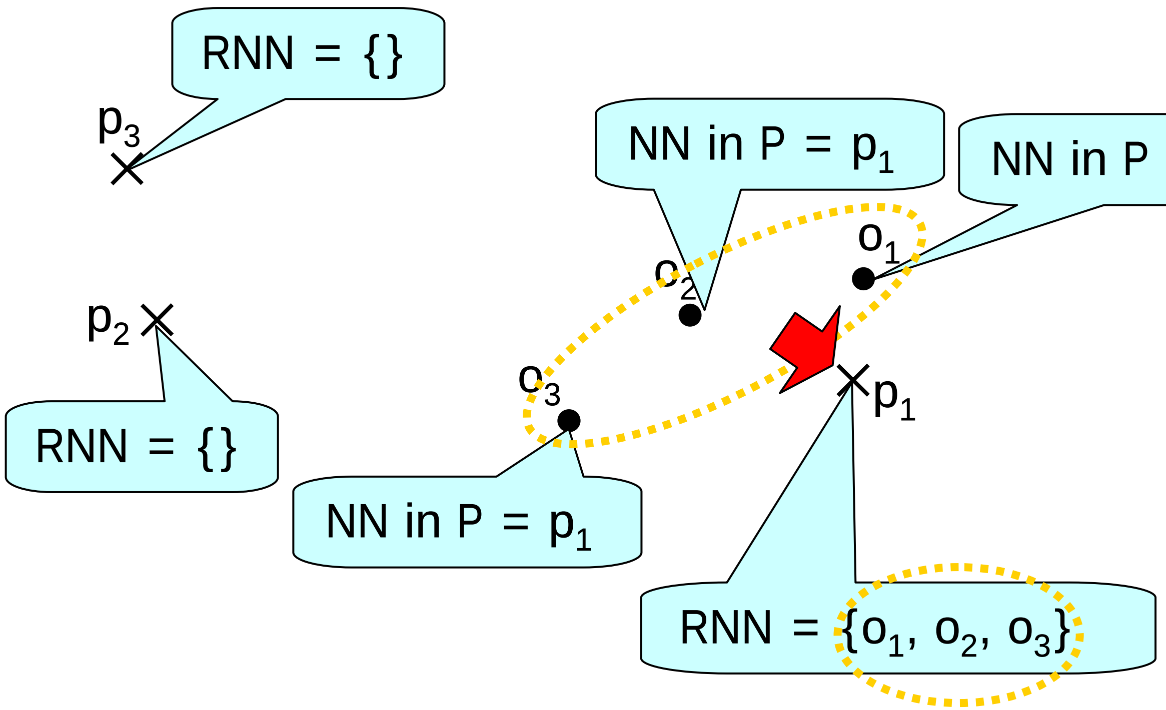
- Bichromatic Reverse Nearest Neighbor (BRNN)
  - Given
    - P and O are two sets of objects in the same data space
  - Problem
    - Given an object $p \in P$, a BRNN query finds all the objects $o \in O$ whose nearest neighbor (NN) in P are p.

# 1. Introduction

$P = \{p_1, p_2, p_3\}$ Polling places

$O = \{o_1, o_2, o_3\}$ Residential estates

NN: Nearest neighbor
RNN: Reverse nearest neighbor

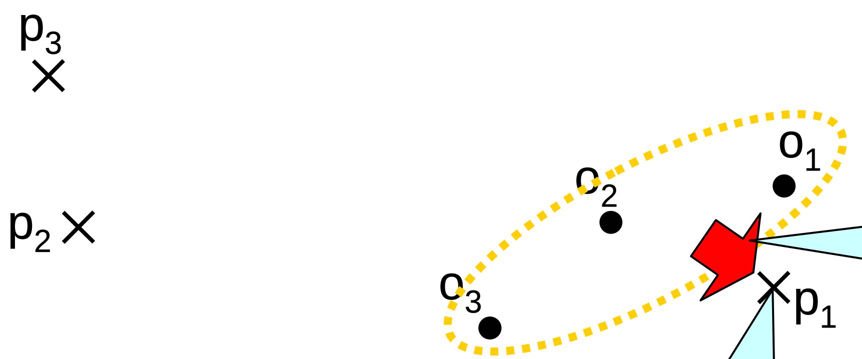


# 1. Introduction

$P = \{p_1, p_2, p_3\}$ Polling places

$O = \{o_1, o_2, o_3\}$ Residential estates

NN: Nearest neighbor
RNN: Reverse nearest neighbor




However, this assignment is not suitable because each polling place has a “**serving capacity**”.

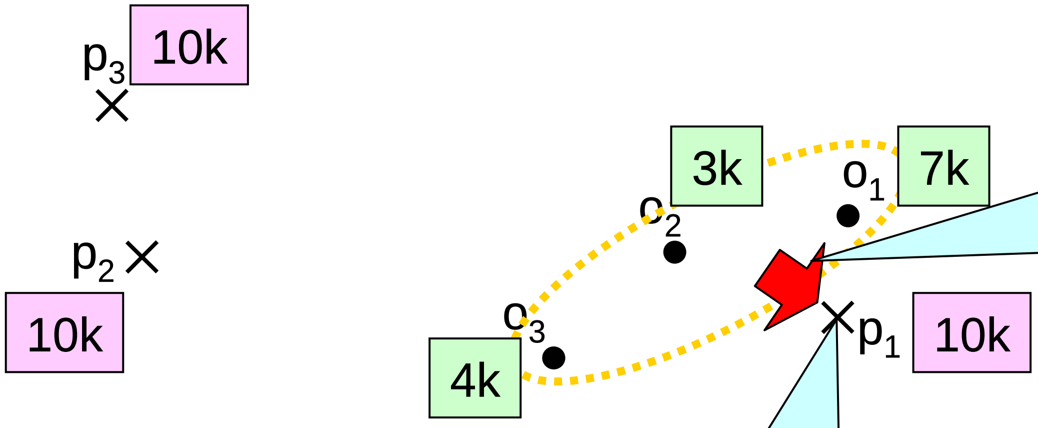
$RNN = \{o_1, o_2, o_3\}$

Spatial matching (SPM)

**Problem:** to find an assignment between P and O with the consideration of the population of $p_i \in P$ and the capacity of $o_j \in O$.

# 1. Introduction

**Idea:** SPM aims at allocating each estate $o \in O$ to the polling-place $p \in P$ that
(i) is as near to o as possible, and
(ii) its servicing capacity has not been exhausted in serving other closer estates.




The total population from $o_1$, $o_2$ and $o_3$ is equal to 14k, which is greater than the capacity of $p_1$

RNN = $\{o_1, o_2, o_3\}$

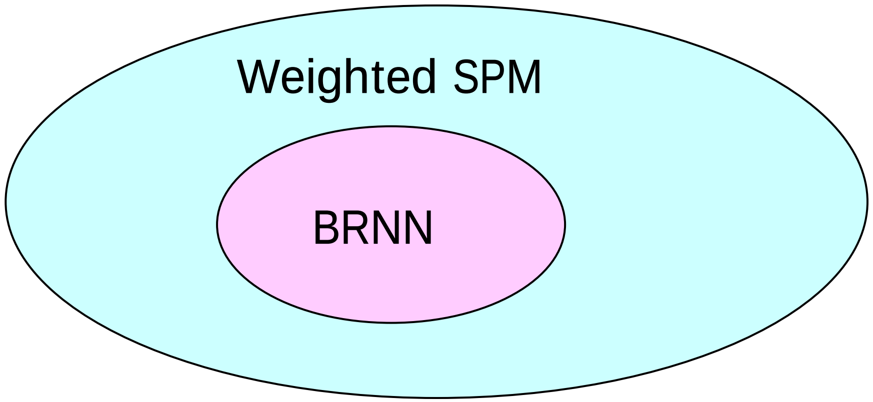
Thus, bichromatic RNN cannot handle this assignment problem.

# 2. Problem

1. Unweighted SPM
   - Population of $p_i \in P$ (denoted by $p_i.w$) = 1
   - Capacity of $o_j \in O$ (denoted by $o_j.w$) = 1
2. Weighted SPM
   - Population of $p_i \in P$ (denoted by $p_i.w$) $\geq 1$
   - Capacity of $o_j \in O$ (denoted by $o_j.w$) $\geq 1$

# 2. Problem

- **Theorem:** The problem of computing the BRNN set of each object $p \in P$ is an instance of weighted SPM, where
  - $p.w = |O|$ for every $p \in P$ and
  - $o.w=1$ for every $o \in O$.



# 2. Problem

- Related Work
  - Closest Pair
    - Running time = $O(|P| \times |O|^2)$
  - Stable Marriage
    - A classical problem in Computer Science
    - Running time = $O(|P| \times |O|)$
- Our Proposed Algorithm **Chain**
  - Running time = $O( |O| \times \log^{O(1)} |P| )$
  - Significant improvement on running time

Spatial matching (SPM)

**Problem:** to find an assignment between P and O with the consideration of the population of $p_i \in P$ and the capacity of $o_j \in O$.

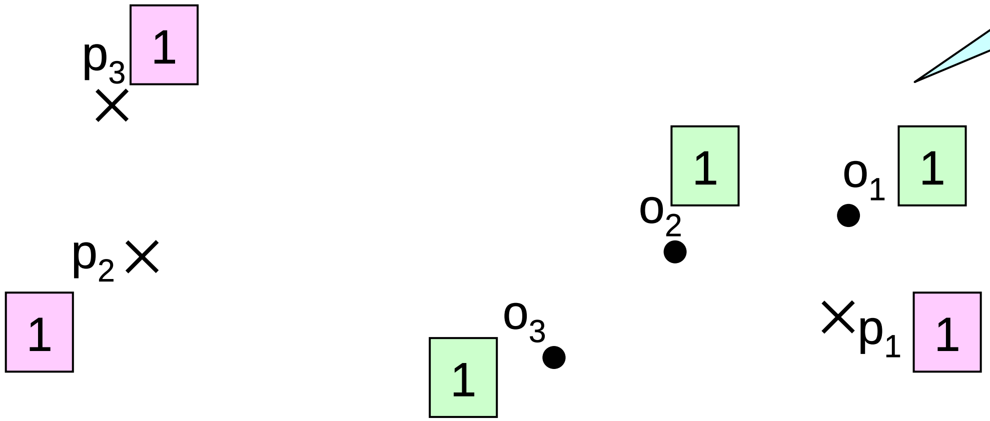
# 2. Problem

Unweighted SPM

$P = \{p_1, p_2, p_3\}$ Polling places

$O = \{o_1, o_2, o_3\}$ Residential estates

How can we perform an assignment between P and O?



Spatial matching (SPM)

**Problem:** to find an assignment between P and O with the consideration of the population of $p_i \in P$ and the capacity of $o_j \in O$.

# 2. Problem

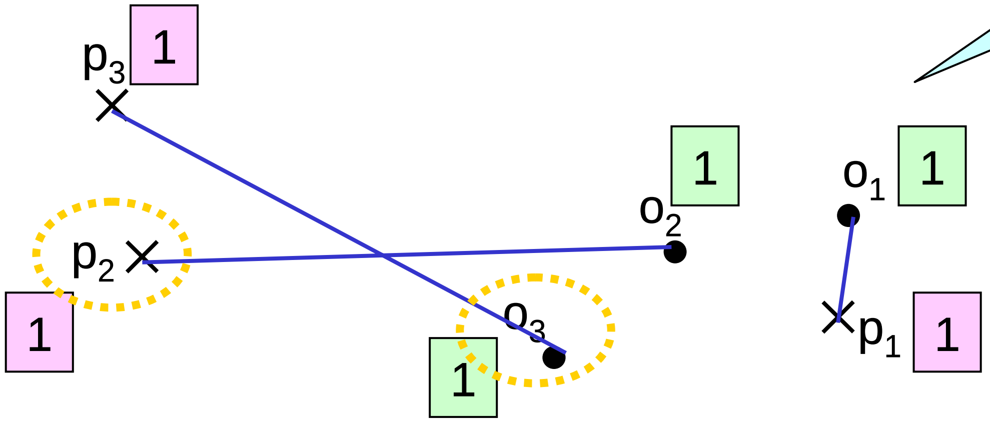
Unweighted SPM

$P = \{p_1, p_2, p_3\}$ Polling places

$O = \{o_1, o_2, o_3\}$ Residential estates

How can we perform an assignment between P and O?

First, we consider an assignment A.



Spatial matching (SPM)

**Problem:** to find an assignment between P and O with the consideration of the population of $p_i \in P$ and the capacity of $o_j \in O$.

# 2. Problem

Unweighted SPM

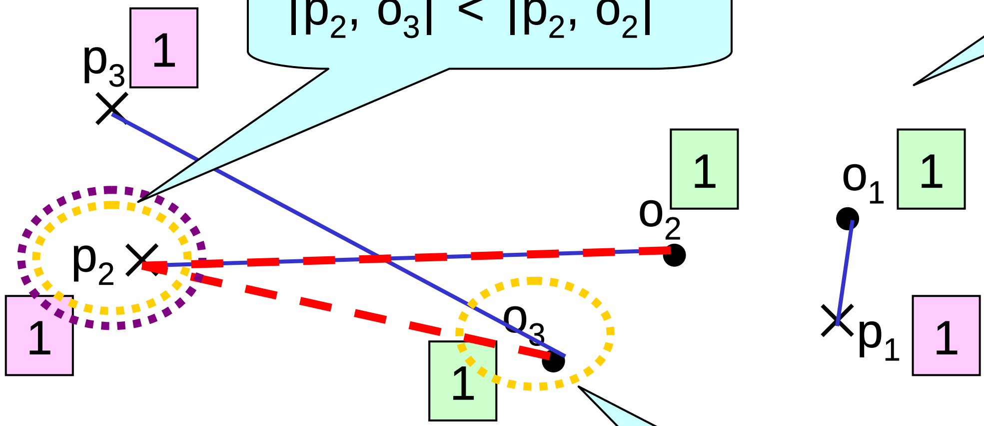
$P = \{p_1, p_2, p_3\}$ — Polling places

$O = \{o_1, o_2, o_3\}$ — Residential estates

How can we perform an assignment between P and O?

First, we consider an assignment A.

$|p_2, o_3| < |p_2, o_2|$




(p, o) is a **dangling pair** if

1. $|p, o|$ < the distance between o and its partner in A
2. $|p, o|$ < the distance between p and its partner in A

$(p_2, o_3)$ is a **dangling pair**.

# 2. Problem

Spatial matching (SPM)

**Problem:** to find an assignment between P and O with the consideration of the population of $p_i \in P$ and the capacity of $o_j \in O$.

Unweighted SPM

$P = \{p_1, p_2, p_3\}$ Polling places

$O = \{o_1, o_2, o_3\}$ Residential estates

If the assignment A does NOT contain any dangling pair, then the assignment is **fair**.

How can we perform

?

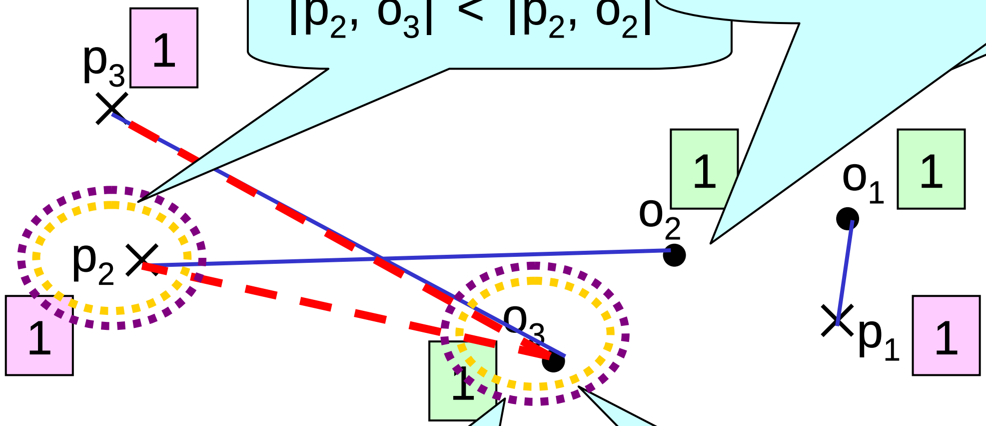
This assignment is NOT fair because we find a dangling pair.

$|p_2, o_3| < |p_2, o_2|$

First, we consider an assignment A.

(p, o) is a **dangling pair** if

1. $|p, o| <$ the distance between o and its partner in A
2. $|p, o| <$ the distance between p and its partner in A




$|p_2, o_3| < |p_3, o_3|$

$(p_2, o_3)$ is a **dangling pair**.

Spatial matching (SPM)

**Problem:** to find an assignment between P and O with the consideration of the population of $p_i \in P$ and the capacity of $o_j \in O$.

# 2. Problem

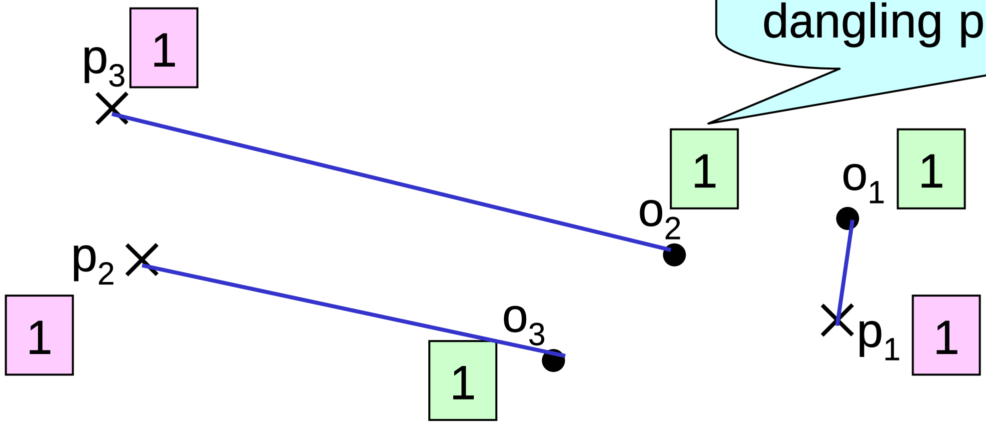
Unweighted SPM

$P = \{p_1, p_2, p_3\}$ Polling places

$O = \{o_1, o_2, o_3\}$ Residential estates

If the assignment A does NOT contain any dangling pair, then the assignment is **fair**.

This assignment is fair because we cannot find a dangling pair.




(p, o) is a **dangling pair** if

1. |p, o| < the distance between o and its partner in A
2. |p, o| < the distance between p and its partner in A

# 2. Problem

- Unweighted SPM
  - Dangling pair
- Weighted SPM
  - Dangling pair

# 3. Algorithm

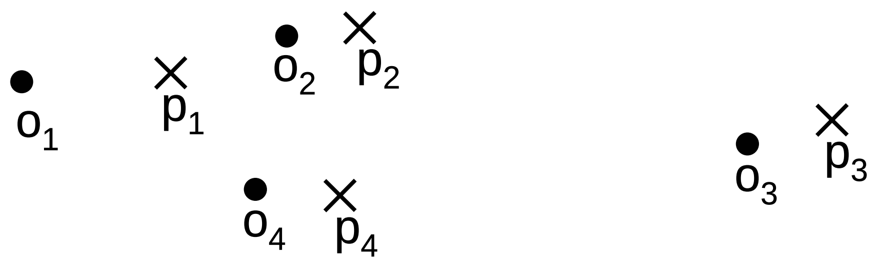
- Un-weighted SPM problem
  - Algorithm (Un-weighted) Chain
- Weighted SPM problem
  - Algorithm Weighted Chain

# 3.1 Algorithm

- Algorithm Chain makes use of **bichromatic mutual NN** to find the fair assignment.
- An object $p \in P$ and an object $o \in O$ are **bichromatic mutual NN** if
  - p is the NN of o in P and
  - o is the NN of p in O

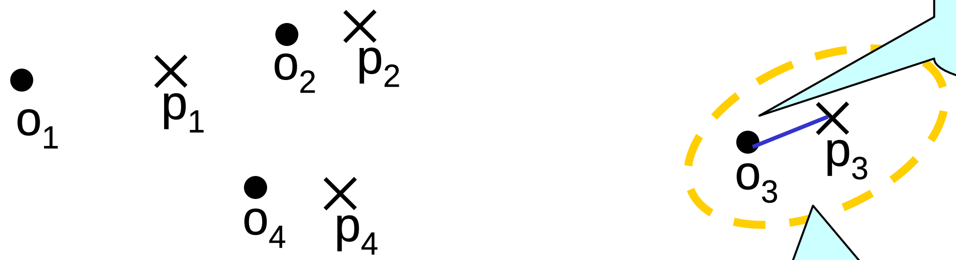
# 3.1 Algorithm

Unweighted SPM



# 3.1 Algorithm

Unweighted SPM




($p_3$, $o_3$) corresponds to a match.

We can remove it.

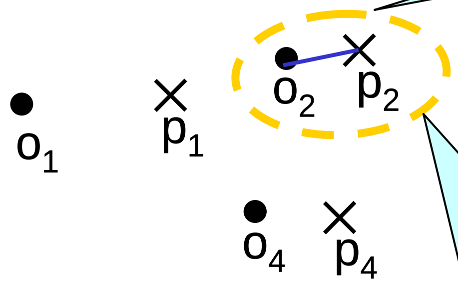
($p_3$, $o_3$) is a pair of mutual NN.

Assignment =
{ ($p_3$, $o_3$)
}

# 3.1 Algorithm

Unweighted SPM

($p_2$, $o_2$) corresponds to a match.



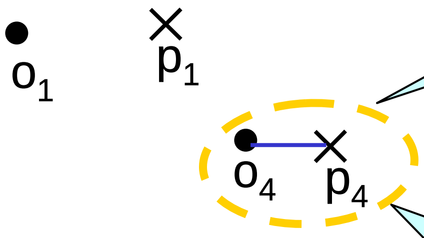

($p_2$, $o_2$) is a pair of mutual NN.

We can remove it.

Assignment = $\{ (p_3, o_3) , (p_2, o_2) \}$

# 3.1 Algorithm

Unweighted SPM




$(p_4, o_4)$ corresponds to a match.

$(p_4, o_4)$ is a pair of mutual NN.

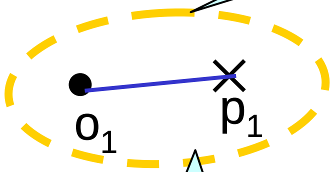
We can remove it.

Assignment =
$\{ (p_3, o_3), (p_2, o_2), (p_4, o_4) \}$

# 3.1 Algorithm

Unweighted SPM

($p_1$, $o_1$) corresponds to a match.




($p_1$, $o_1$) is a pair of mutual NN.

We can remove it.

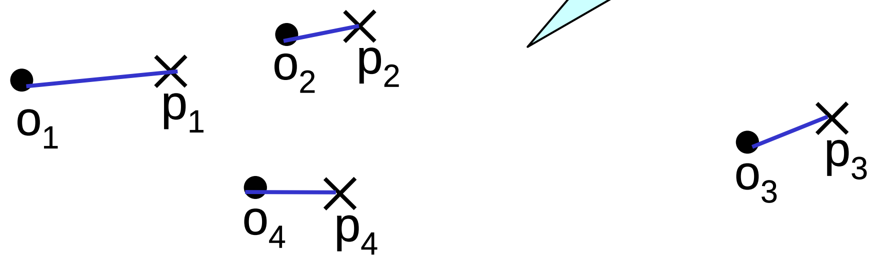
Assignment =
{ ($p_3$, $o_3$) , ($p_2$, $o_2$) ,
($p_4$, $o_4$) , ($p_1$, $o_1$) }

# 3.1 Algorithm

Unweighted SPM

We prove that this assignment is fair.

We can find a fair assignment by repeatedly removing pairs of mutual NN.




But, how can we find a pair of mutual NN efficiently?

Assignment =
$\{ (p_3, o_3), (p_2, o_2),$
$(p_4, o_4), (p_1, o_1) \}$

We propose Algorithm Chain to perform mutual NN search efficiently.

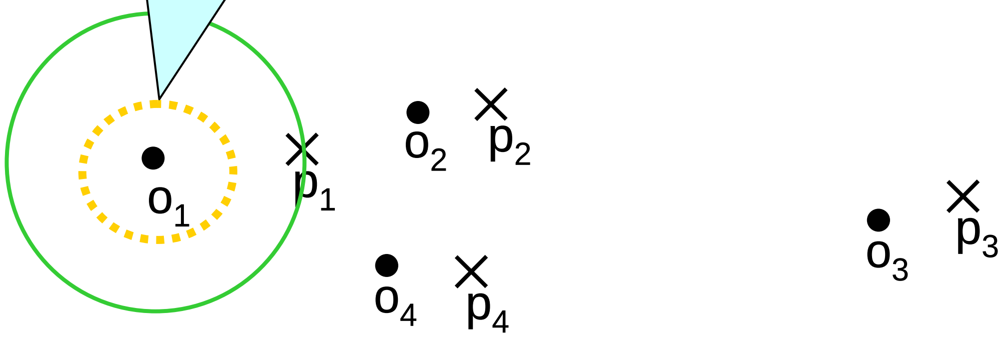
# 3.1 Algorithm

- Find the first mutual NN (nearest neighbor) and remove it
- Find the second mutual NN and remove it
- …
- Find the n-th mutual NN and remove it

# 3.1 Algorithm Chain

Unweighted SPM

From $o_1$, find NN in P (i.e., $p_1$)




Randomly find a data point o

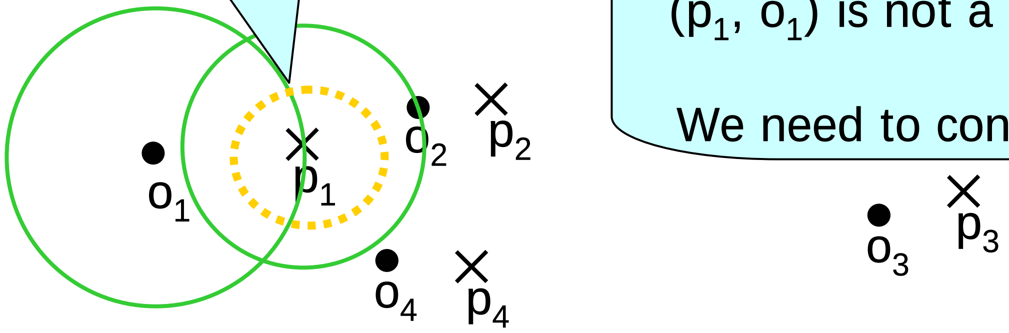
# 3.1 Algorithm Chain

Unweighted SPM

From $p_1$, find NN in O (i.e., $o_2$)

Since $o_2$ is NOT equal to $o_1$,
($p_1$, $o_1$) is not a pair of mutual NN.

We need to continue the process.



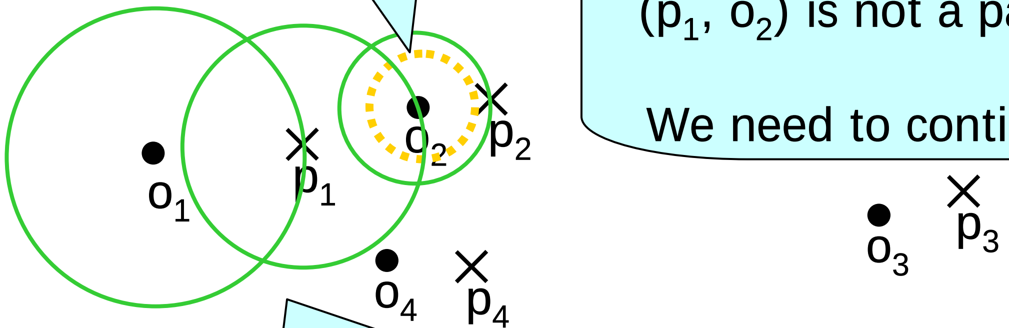
# 3.1 Algorithm Chain

Unweighted SPM

From $o_2$, find NN in P (i.e., $p_2$)

Since $p_2$ is NOT equal to $p_1$,
($p_1$, $o_2$) is not a pair of mutual NN.

We need to continue the process.




Note that we are expanding a **chain** from data point $o_1$.

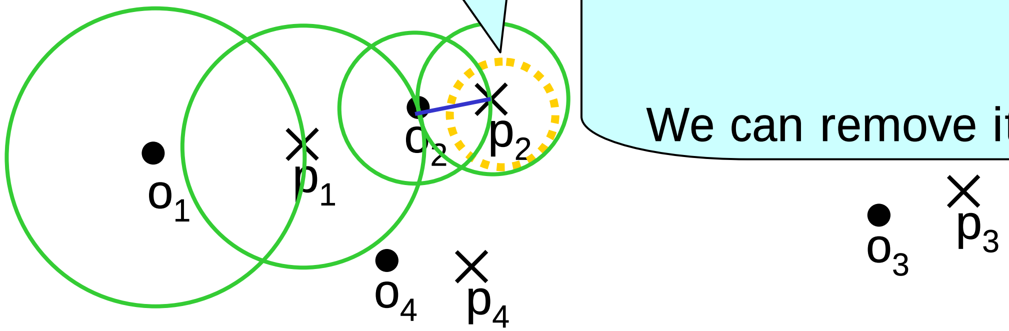
# 3.1 Algorithm Chain

Unweighted SPM

From $p_2$, find NN in O (i.e., $o_2$)

Now, we find a pair of mutual NN ($p_2$, $o_2$).

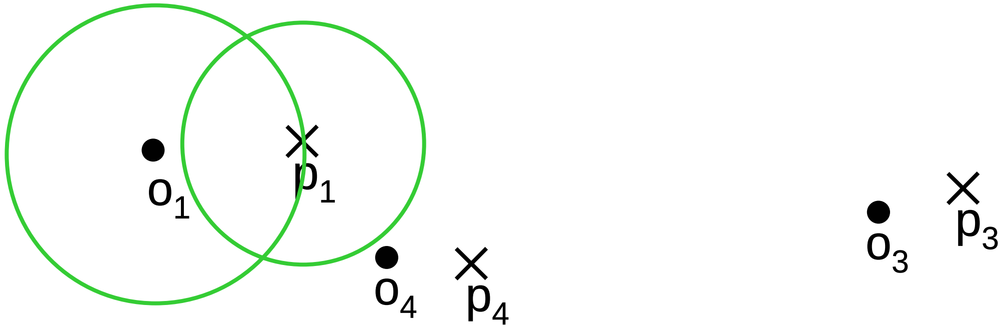
We can remove it.




Assignment =
{ ($p_2$, $o_2$)
}

# 3.1 Algorithm Chain

Unweighted SPM




We find the FIRST mutual NN.

Assignment =

Should we perform similar steps to find the SECOND mutual NN?

That is, should we randomly select a data point again and re-start the chain?

Yes. We can do in this way.
But, it is NOT efficient.
Instead, we can re-use the existing chain to find the SECOND mutual NN.

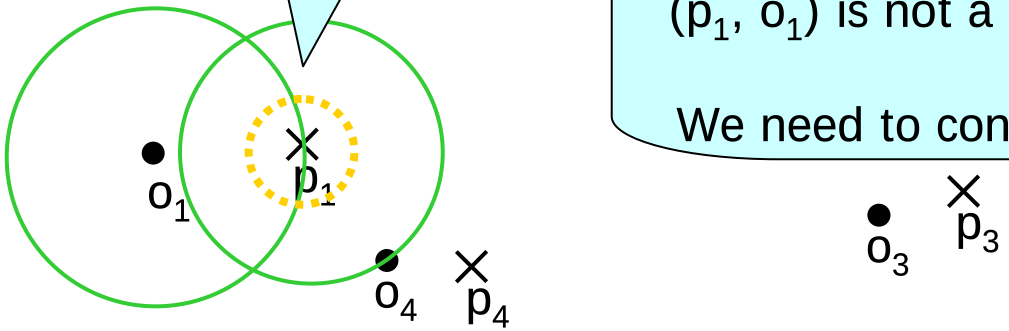
# 3.1 Algorithm Chain

Unweighted SPM

From $p_1$, find NN in O (i.e., $o_4$)

Since $o_4$ is NOT equal to $o_1$,
($p_1$, $o_1$) is not a pair of mutual NN.

We need to continue the process.




Assignment =
{ ($p_2$, $o_2$)
}

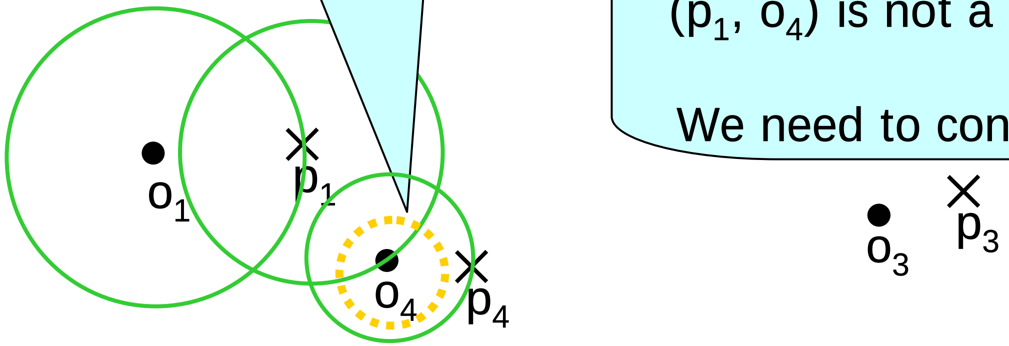
# 3.1 Algorithm Chain

Unweighted SPM

From $o_4$, find NN in P (i.e., $p_4$)

Since $p_4$ is NOT equal to $p_1$,
($p_1$, $o_4$) is not a pair of mutual NN.

We need to continue the process.




Assignment =
{ ($p_2$, $o_2$)
}

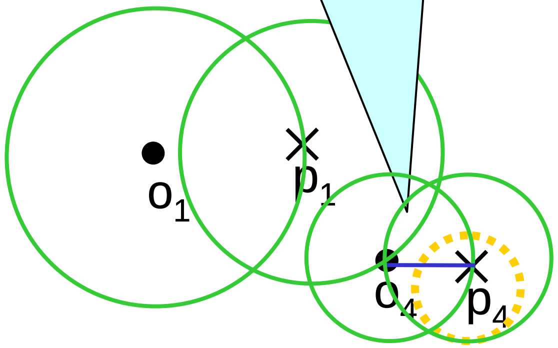
# 3.1 Algorithm Chain

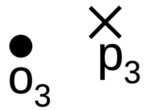
Unweighted SPM

From $p_4$, find NN in O (i.e., $o_4$)

Now, we find a pair of mutual NN ($p_4$, $o_4$).

We can remove it.




Assignment =
{ ($p_2$, $o_2$) , ($p_4$, $o_4$) }

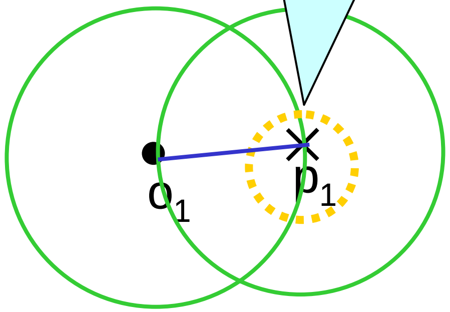
# 3.1 Algorithm Chain

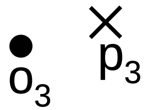
Unweighted SPM

From $p_1$, find NN in O (i.e., $o_1$)

Now, we find a pair of mutual NN ($p_1$, $o_1$).

We can remove it.



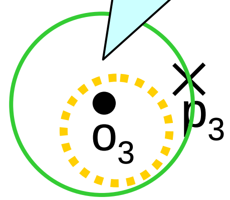




Assignment =
{ ($p_2$, $o_2$) , ($p_4$, $o_4$) ,
($p_1$, $o_1$) }

# 3.1 Algorithm Chain

Unweighted SPM

From $o_3$, find NN in P (i.e., $p_3$)



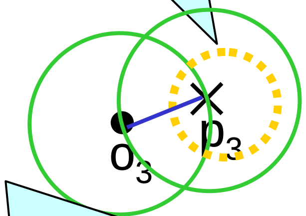

Randomly find a data point o

Assignment =
{ $(p_2, o_2)$ , $(p_4, o_4)$ ,
$(p_1, o_1)$ }

# 3.1 Algorithm Chain

Unweighted SPM

From $p_3$, find NN in O (i.e., $o_3$)




Now, we find a pair of mutual NN ($p_3$, $o_3$).

We can remove it.

Assignment =
{ ($p_2$, $o_2$) , ($p_4$, $o_4$) ,
($p_1$, $o_1$) , ($p_3$, $o_3$) }

# 3.1 Algorithm Chain

Unweighted SPM

- **Theorem:** (Un-weighted) Chain performs at most 3|O| NN queries and exactly 2|O| object deletions.

$\alpha(n)$: worst case complexity of an NN query on dataset of size n
$\beta(n)$: worst case complexity of an object deletion on dataset of size n

- **Theorem:** The running time of (Un-weighted) Chain is $O(\ |O|\ \times (\ \alpha(|P|)+\beta(|P|)\ )\ \ )$

$\alpha(n)$ and $\beta(n)$ can be accomplished in $O(\log^{O(1)}(n))$.

Thus, the running time is $O(\ |O|\ \times \log^{O(1)} |P|\ )$

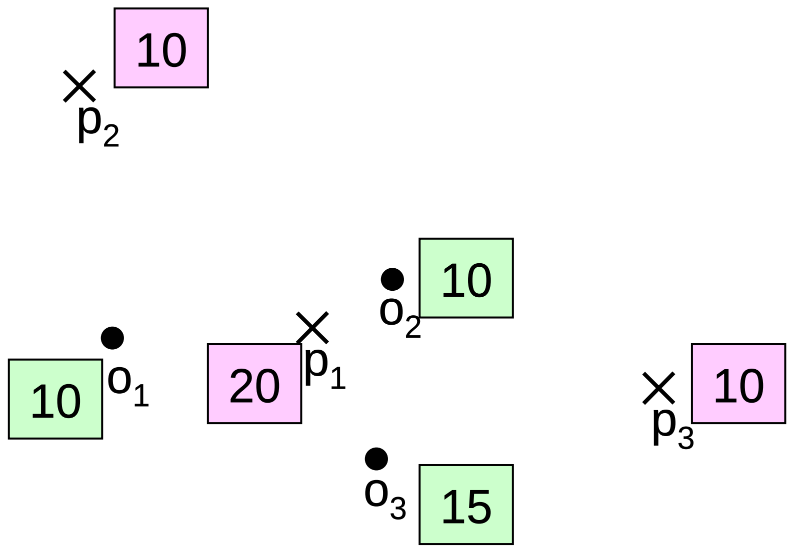
T.M. Chan, A Dynamic Data Structure for 3-d Convex Hulls and 2-d Nearest Neighbor Queries, SODA 2006

# 3.2 Algorithm Weighted Chain

- Similar to (Unweighted) Chain
- Consider the population and the capacity of each point

# 3.2 Algorithm Weighted Chain

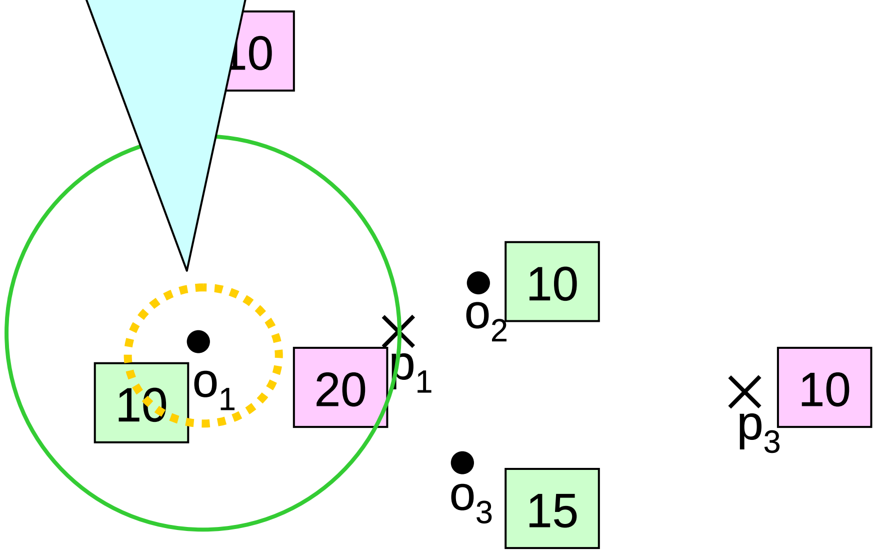
Weighted SPM



# 3.2 Algorithm Weighted Chain

Weighted SPM

From $o_1$, find NN in P (i.e., $p_1$)



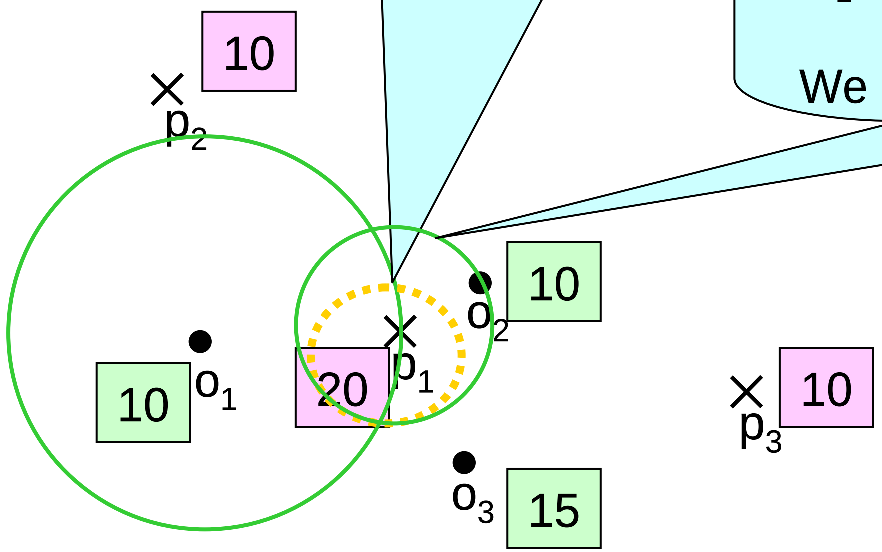
# 3.2 Algorithm Weighted Chain

Weighted SPM

From $p_1$, find NN in O (i.e., $o_2$)

Since $o_2$ is NOT equal to $o_1$,
($p_1$, $o_1$) is not a pair of mutual NN.

We need to continue the process.



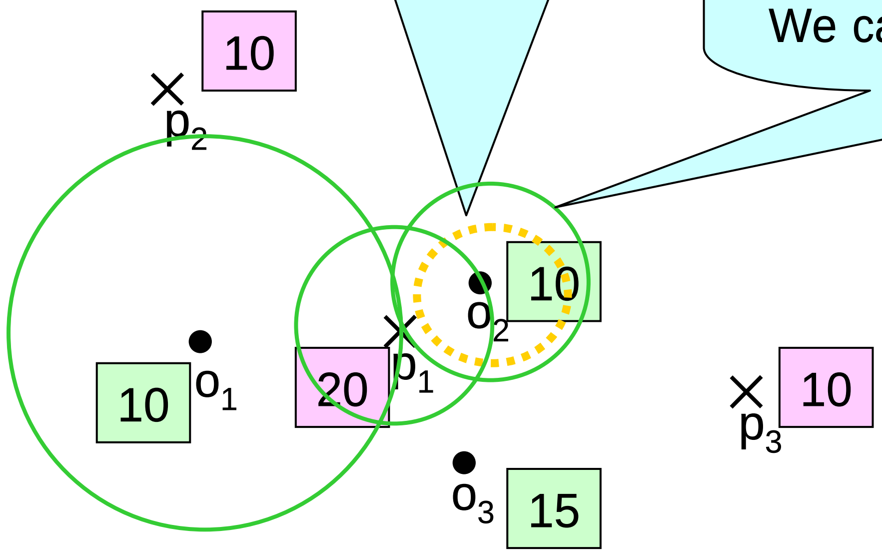
# 3.2 Algorithm Weighted Chain

Weighted SPM

From $o_2$, find NN in P (i.e., $p_1$)

Now, we find a pair of mutual NN ($p_1$, $o_2$).

We can remove ($p_1$, $o_2$, 10).




Assignment =
{ ($p_1$, $o_2$, 10)
}

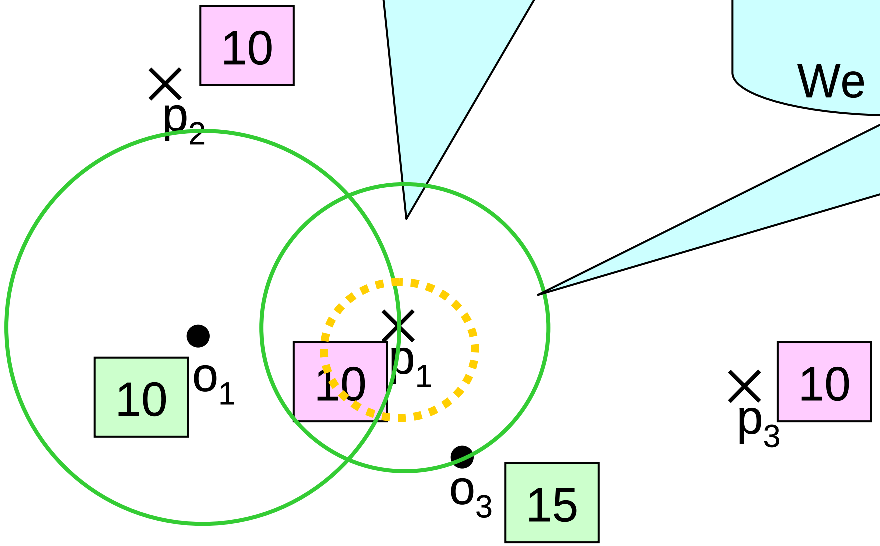
# 3.2 Algorithm Weighted Chain

Weighted SPM

From $p_1$, find NN in O (i.e., $o_3$)

Since $o_3$ is NOT equal to $o_1$,
($p_1$, $o_1$) is not a pair of mutual NN.

We need to continue the process.




Assignment =
{ ($p_1$, $o_2$, 10)
}

Similar steps are performed.

# 3.2 Algorithm Weighted Chain

Weighted SPM

- **Theorem:** Weighted Chain performs at most 3(|P| + |O|) NN queries and at most |P| + |O| object deletions

$\alpha(n)$: worst case complexity of an NN query on dataset of size n
$\beta(n)$: worst case complexity of an object deletion on dataset of size n

- **Theorem:** The running time of Weighted Chain is $O(\ (|P|\ +\ |O|)\ \times\ (\ \alpha(|P|)+\beta(|P|)+\alpha(|O|)+\beta(|O|))\ \ )$

$\alpha(n)$ and $\beta(n)$ can be accomplished in $O(\log^{O(1)}(n))$.

Thus, the running time is $O(\ (|P|\ +\ |O|)\ \times\ (\log^{O(1)} |P|\ +\ \log^{O(1)} |O|)\ \ )$

# 4. Empirical Study

- Synthetic Dataset
  - P: Gaussian distribution
  - O: Zipfian distribution
- Real Dataset
  - Rtree Portal
    http://www.rtreeportal.org/spatial.html
    - CA (62,556)
    - LB (53,145)
    - GR (23,268)
    - GM (36,334)
  - P: one of the above datasets
  - O: one of the above datasets

# 4. Empirical Study

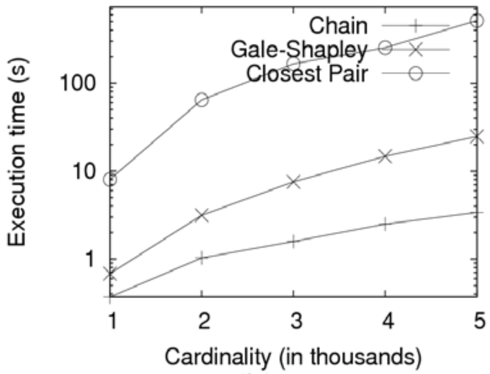
- NN query in Chain
  - Build R*-tree on P
  - Build R*-tree on O

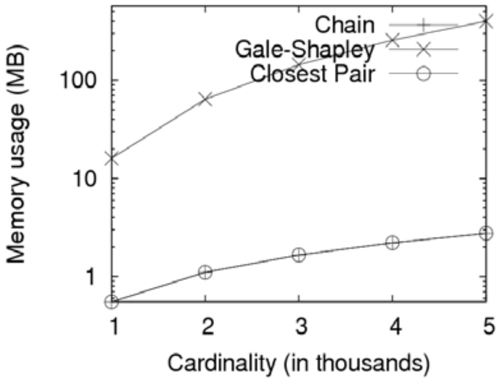
# 4. Empirical Study

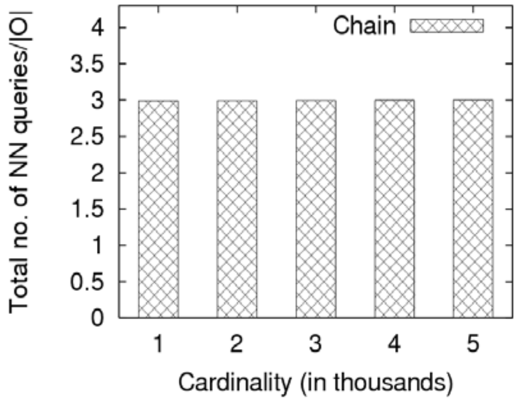
- Measurements
  - Execution Time
  - Memory Usage
  - Total no. of NN queries/|O|
  - Total no. of NN queries/(|P| + |O|)
- Comparison with adapted algorithms
  - Gale-Shapley
  - Closest Pair

# 4. Empirical Study

- Un-weighted SPM

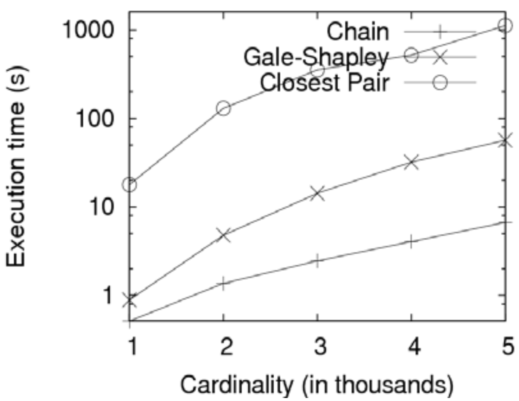


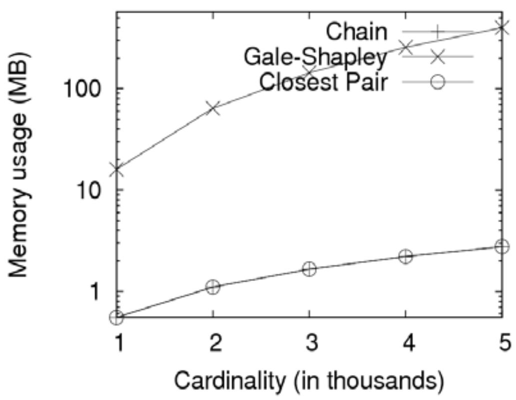



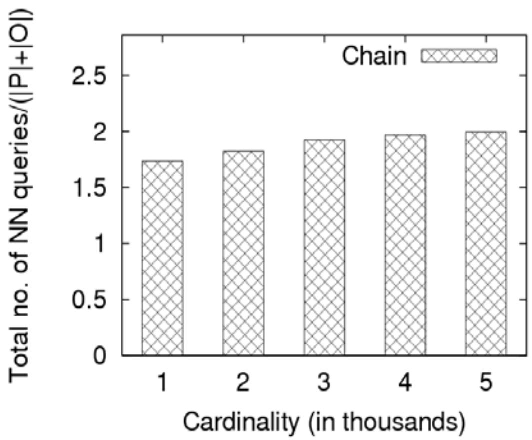



# 4. Empirical Study

- Weighted SPM

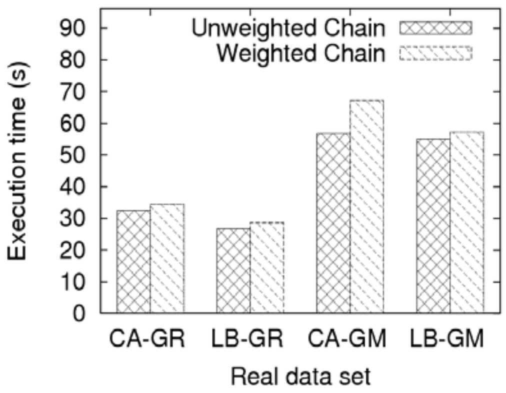


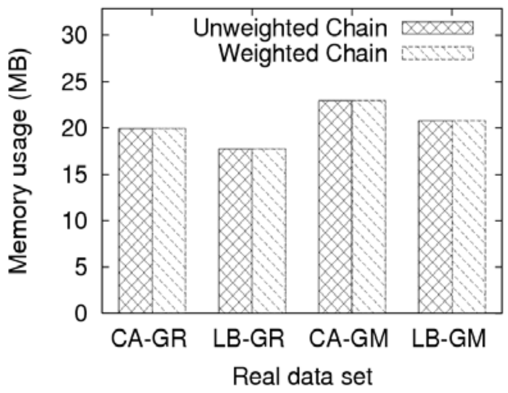



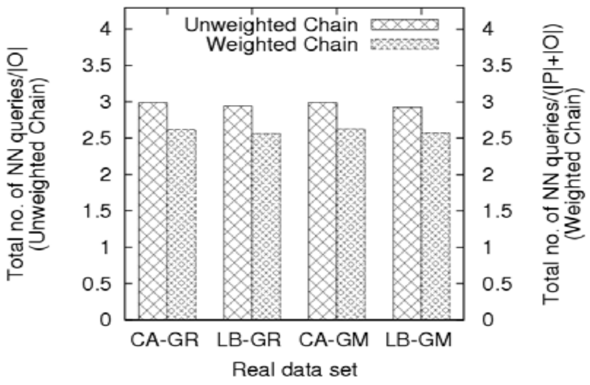



# 4. Empirical Study

- Real Data Set



# 5. Conclusion

- Un-weighted and Weighted Spatial Matching Problem
  - A general model of BRNN
- Algorithm Chain
  - Theoretical Analysis of Running Time
  - Significant Improvement on Running Time
- Experiments

# FAQ

# Stable Marriage

- Two sets O (for woman) and P (for man)
- For each woman $o \in O$,
  - there is a **preference list** which sorts the men in descending order of how much o loves them.
- For each man $p \in P$,
  - there is a **preference list** which sorts the women in descending order of how much p loves them.

- **Stable Marriage**
  - the absence of a man p and a woman o, such that
    - p loves o more than his current partner, and
    - o loves p more than her current partner.

# Stable Marriage

- Reduction to Stable Marriage
  - For each $o \in O$
    - We create a preference list in ascending order of $|o, p|$ for all $p \in P$
  - For each $p \in P$
    - We create a preference list in ascending order of $|o, p|$ for all $o \in O$

# 2. Problem

Spatial matching (SPM)

**Problem:** to find an assignment between P and O with the consideration of the population of $p_i \in P$ and the capacity of $o_j \in O$.

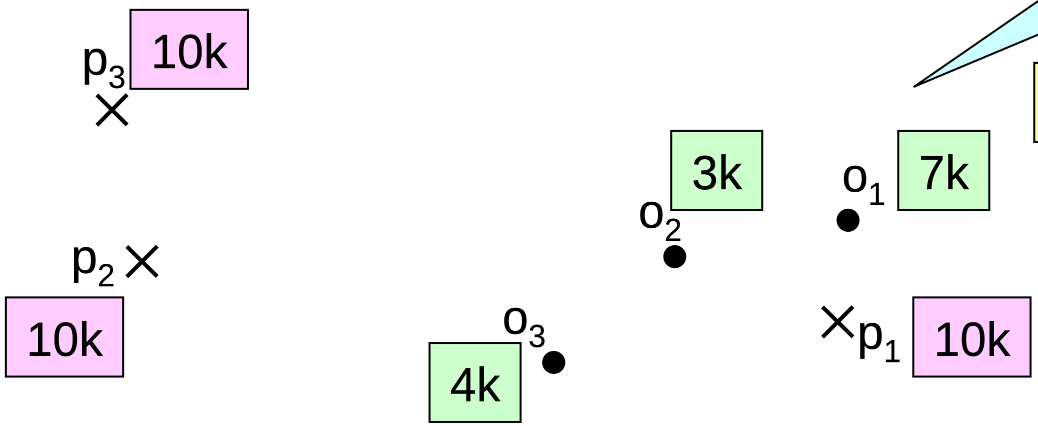
Weighted SPM

$P = \{p_1, p_2, p_3\}$ Polling places

$O = \{o_1, o_2, o_3\}$ Residential estates

How can we perform an assignment between P and O?

First, we consider an assignment A.



Spatial matching (SPM)

**Problem:** to find an assignment between P and O with the consideration of the population of $p_i \in P$ and the capacity of $o_j \in O$.

# 2. Problem

Weighted SPM

$P = \{p_1, p_2, p_3\}$

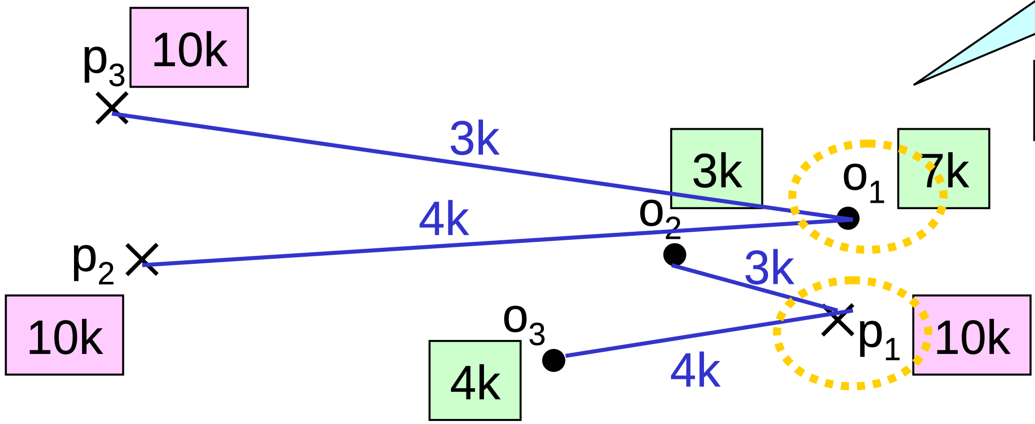
Polling places

$O = \{o_1, o_2, o_3\}$

Residential estates

How can we perform an assignment between P and O?

First, we consider an assignment A.



Spatial matching (SPM)

**Problem:** to find an assignment between P and O with the consideration of the population of $p_i \in P$ and the capacity of $o_j \in O$.

# 2. Problem

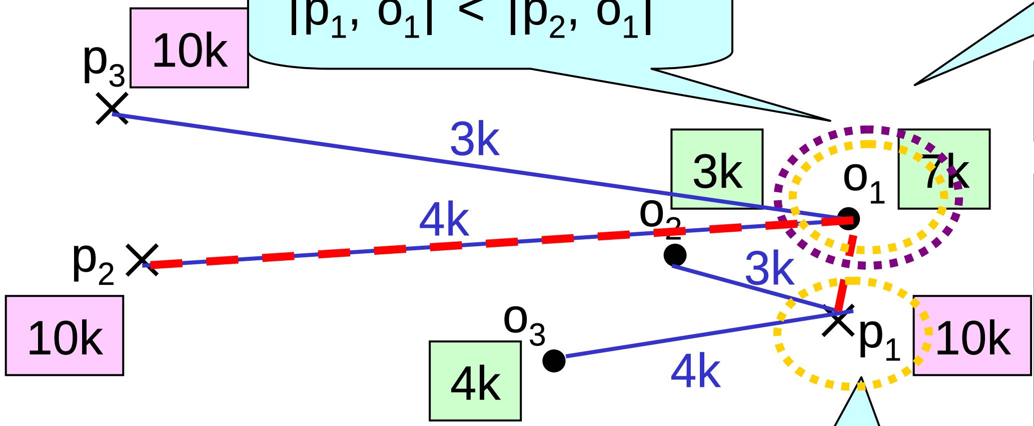
Weighted SPM

$P = \{p_1, p_2, p_3\}$ Polling places

$O = \{o_1, o_2, o_3\}$ Residential estates

How can we perform an assignment between P and O?

$|p_1, o_1| < |p_2, o_1|$




First, we consider an assignment A.

(p, o) is a **dangling pair** if

1. |p, o| < the distance between o and some of its partners in A
2. |p, o| < the distance between p and some of its partners in A

$(p_1, o_1)$ is a **dangling pair**.

# 2. Problem

Spatial matching (SPM)

**Problem:** to find an assignment between P and O with the consideration of the population of $p_i \in P$ and the capacity of $o_j \in O$.

Weighted SPM

$P = \{p_1, p_2, p_3\}$ Polling places

$O = \{o_1, o_2, o_3\}$ Residential estates

If the assignment A does NOT contain any dangling pair, then the assignment is **fair**.

How can we perform

?

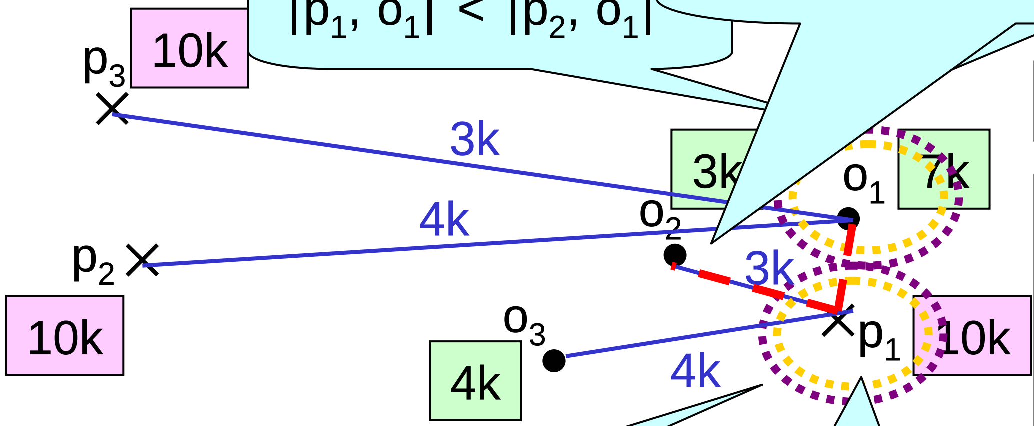
This assignment is NOT fair because we find a dangling pair.

First, we consider an assignment A.

(p, o) is a **dangling pair** if

1. |p, o| < the distance between o and some of its partners in A
2. |p, o| < the distance between p and some of its partners in A

$|p_1, o_1| < |p_2, o_1|$




$|p_1, o_1| < |p_1, o_2|$

$(p_1, o_1)$ is a **dangling pair**.

Spatial matching (SPM)

**Problem:** to find an assignment between P and O with the consideration of the population of $p_i \in P$ and the capacity of $o_j \in O$.

# 2. Problem

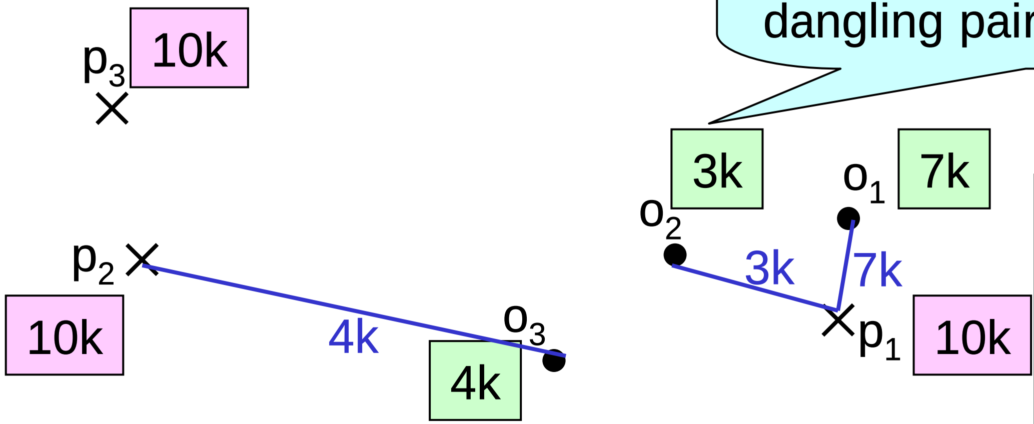
Weighted SPM

$P = \{p_1, p_2, p_3\}$ Polling places

$O = \{o_1, o_2, o_3\}$ Residential estates

If the assignment A does NOT contain any dangling pair, then the assignment is **fair**.

This assignment is fair because we cannot find a dangling pair.

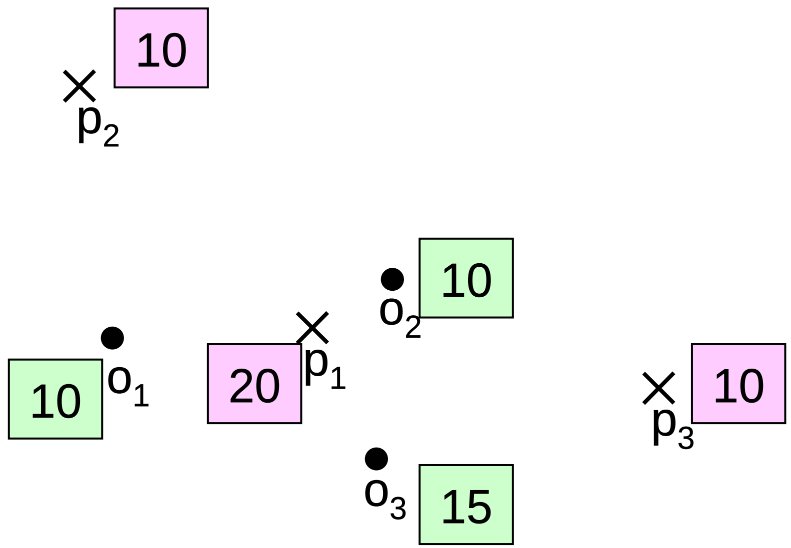



(p, o) is a **dangling pair** if

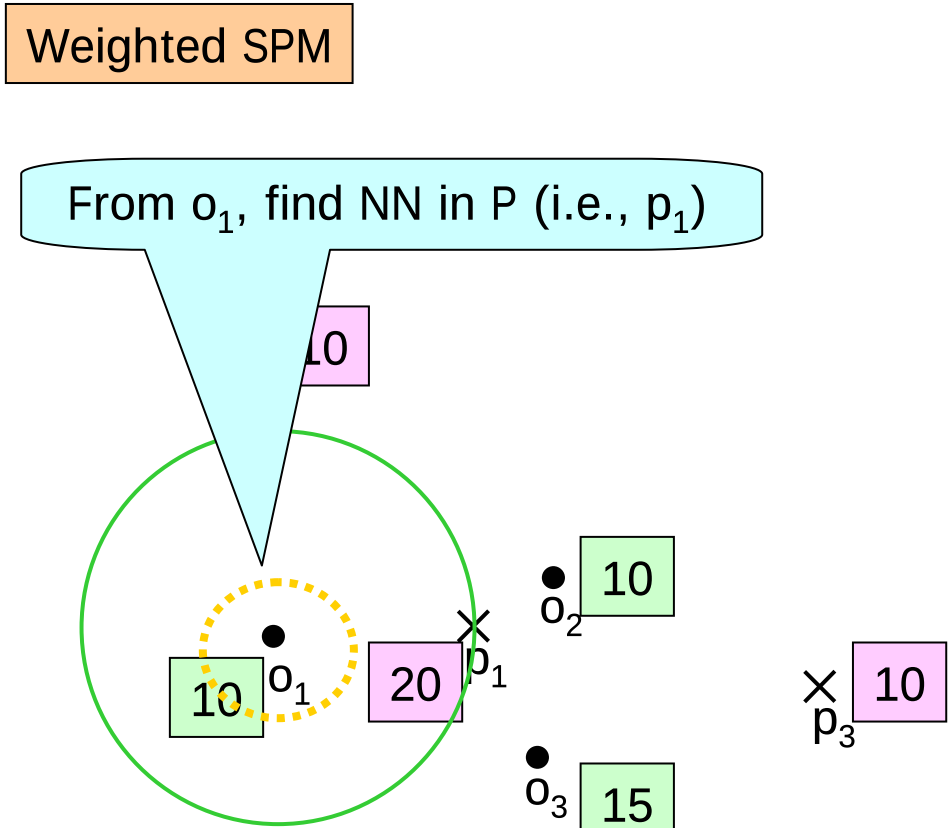
1. |p, o| < the distance between o and some of its partners in A
2. |p, o| < the distance between p and some of its partners in A

# 3.2 Algorithm Weighted Chain

Weighted SPM



# 3.2 Algorithm Weighted Chain



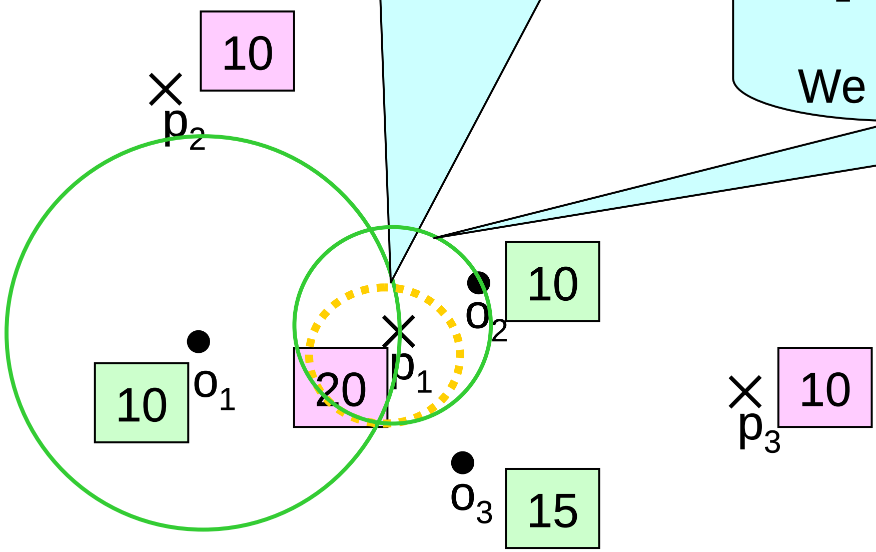
# 3.2 Algorithm Weighted Chain

Weighted SPM

From $p_1$, find NN in O (i.e., $o_2$)

Since $o_2$ is NOT equal to $o_1$,
($p_1$, $o_1$) is not a pair of mutual NN.

We need to continue the process.



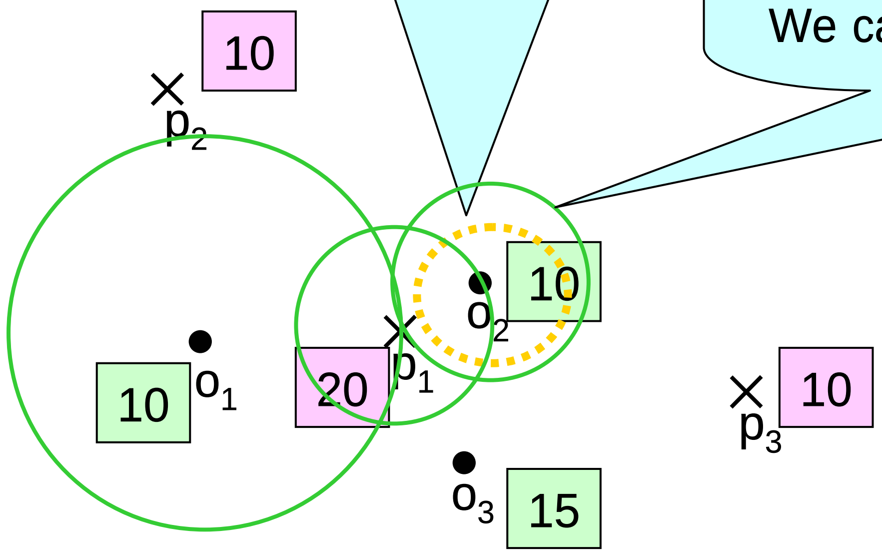
# 3.2 Algorithm Weighted Chain

Weighted SPM

From $o_2$, find NN in P (i.e., $p_1$)

Now, we find a pair of mutual NN ($p_1$, $o_2$).

We can remove ($p_1$, $o_2$, 10).




Assignment =
{ ($p_1$, $o_2$, 10)
}

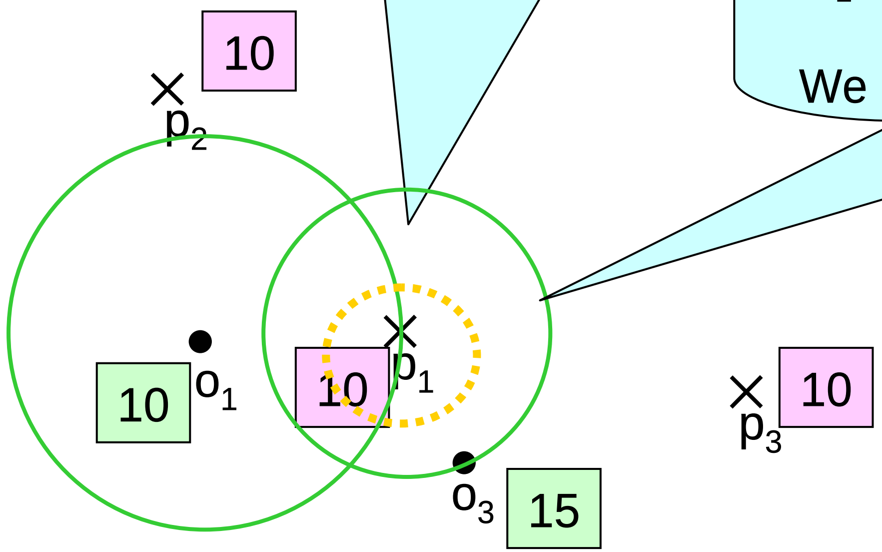
# 3.2 Algorithm Weighted Chain

Weighted SPM

From $p_1$, find NN in O (i.e., $o_3$)

Since $o_3$ is NOT equal to $o_1$,
($p_1$, $o_1$) is not a pair of mutual NN.

We need to continue the process.




Assignment =
{ ($p_1$, $o_2$, 10)
}

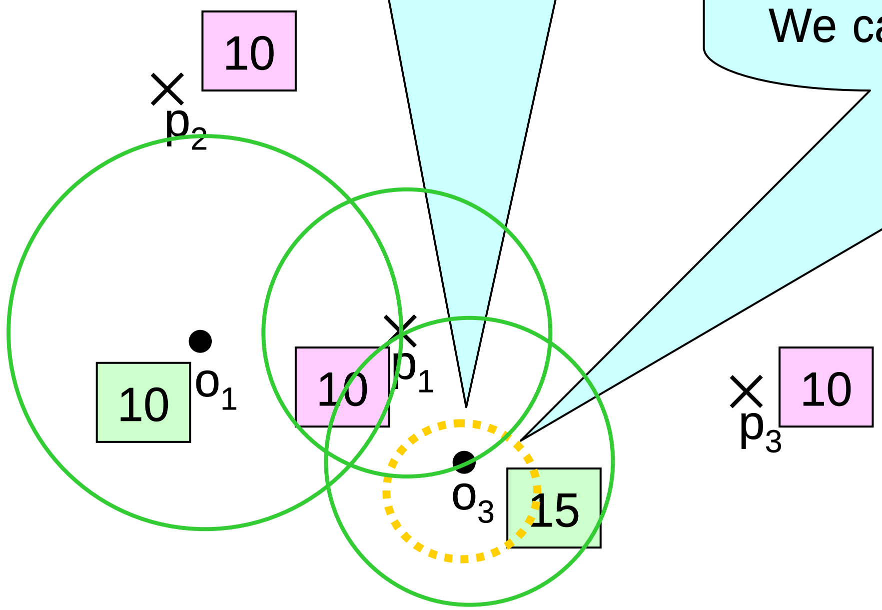
# 3.2 Algorithm Weighted Chain

Weighted SPM

From $o_3$, find NN in P (i.e., $p_1$)

Now, we find a pair of mutual NN ($p_1$, $o_3$).

We can remove ($p_1$, $o_3$, 10).

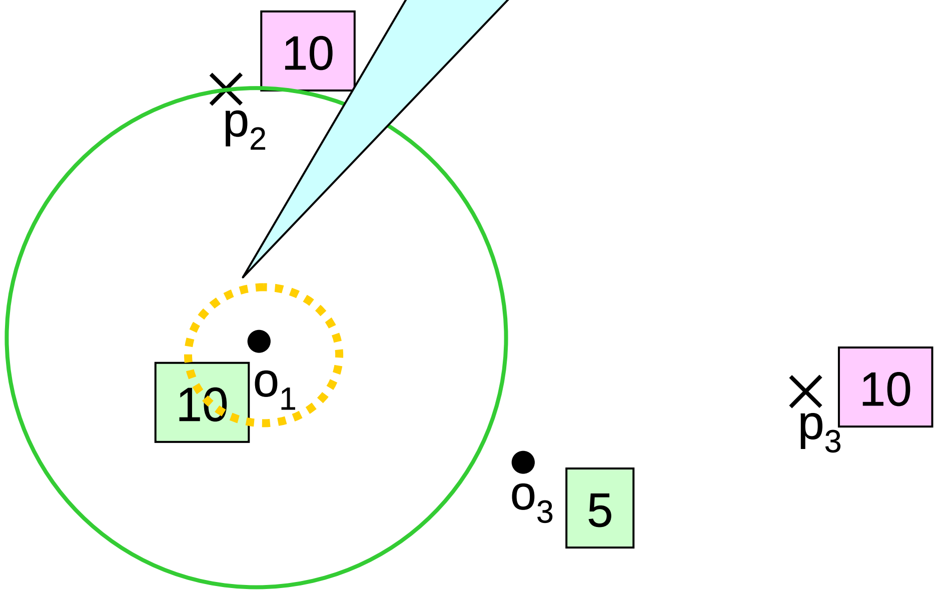



Assignment =
{ ($p_1$, $o_2$, 10), ($p_1$, $o_3$, 10)
}

# 3.2 Algorithm Weighted Chain

Weighted SPM

From $o_1$, find NN in P (i.e., $p_2$)




Assignment =
{ $(p_1, o_2, 10)$, $(p_1, o_3, 10)$
}

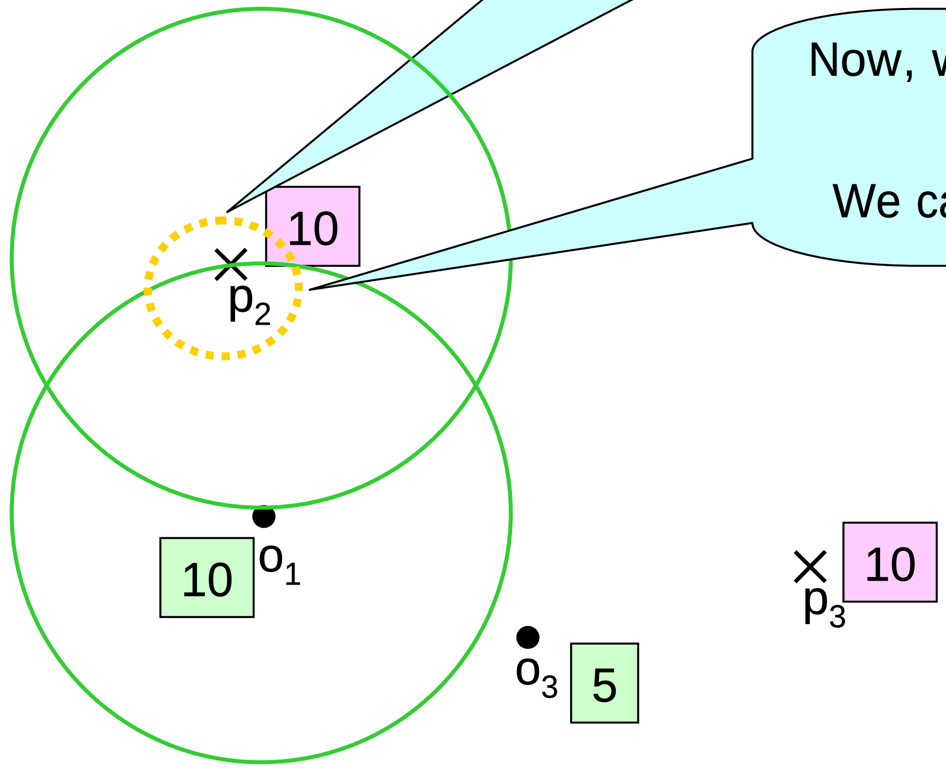
# 3.2 Algorithm Weighted Chain

Weighted SPM

From $p_2$, find NN in O (i.e., $o_1$)

Now, we find a pair of mutual NN ($p_2$, $o_1$).

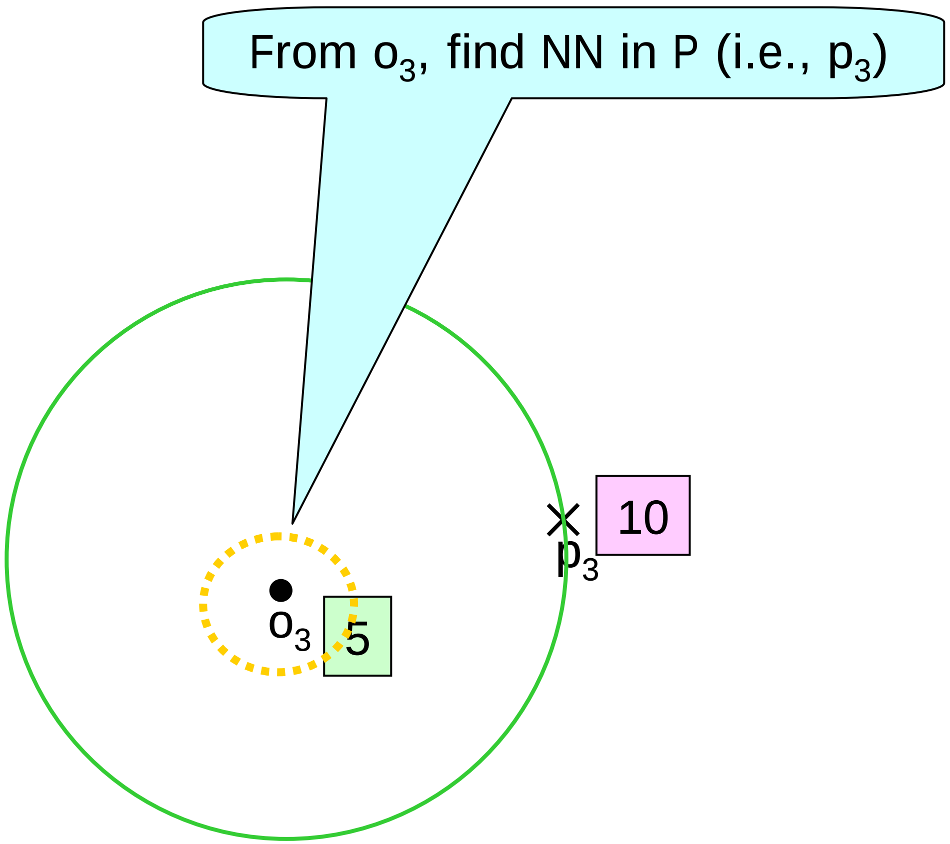
We can remove ($p_2$, $o_1$, 10).




Assignment =
{ ($p_1$, $o_2$, 10), ($p_1$, $o_3$, 10),
($p_2$, $o_1$, 10) }

# 3.2 Algorithm Weighted Chain

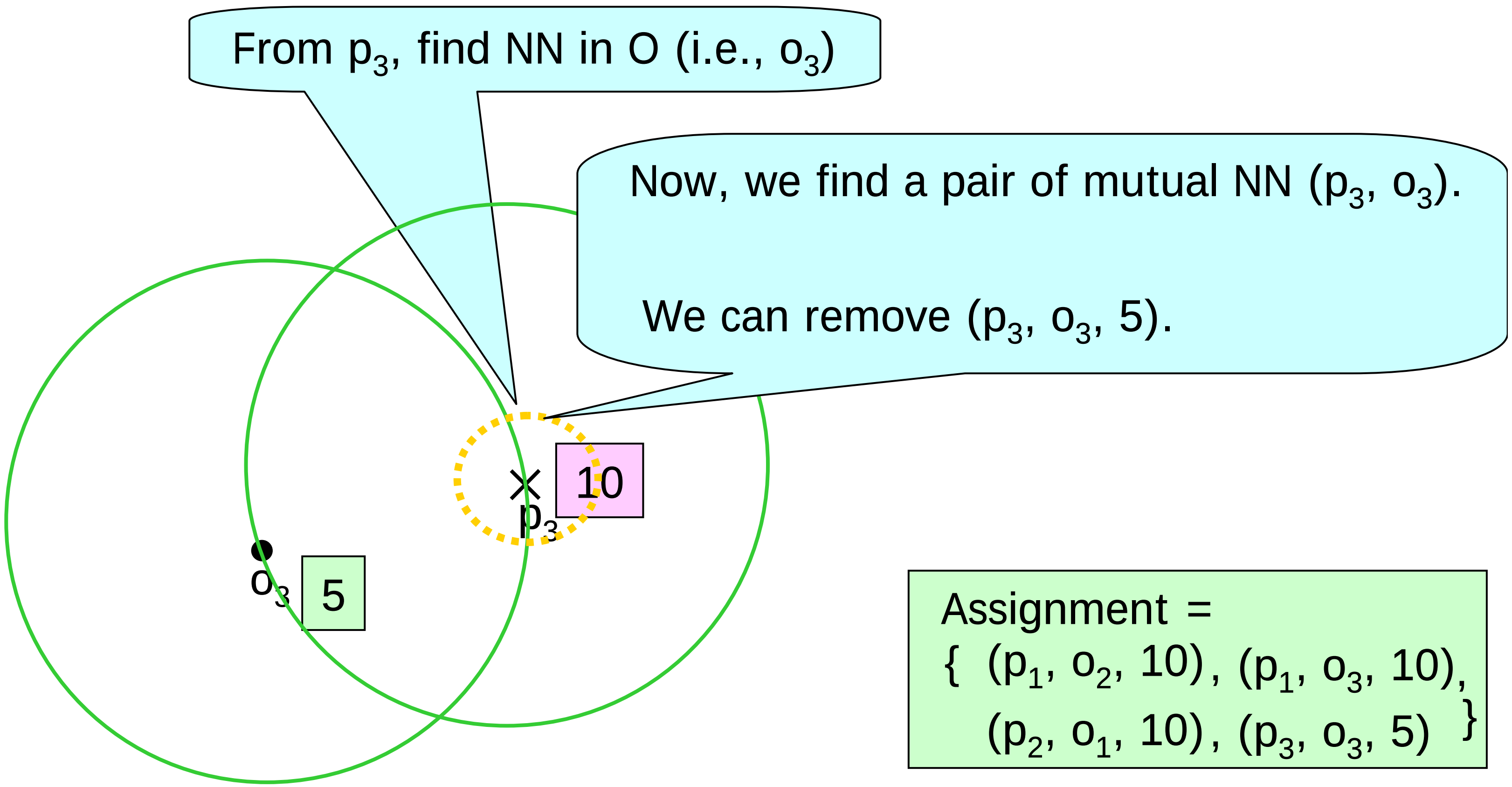
Weighted SPM




Assignment =
{ $(p_1, o_2, 10)$, $(p_1, o_3, 10)$,
$(p_2, o_1, 10)$ }

# 3.2 Algorithm Weighted Chain

Weighted SPM




Assignment =
{ ($p_1$, $o_2$, 10), ($p_1$, $o_3$, 10),
($p_2$, $o_1$, 10), ($p_3$, $o_3$, 5) }