Advanced Algorithms for Geo-Information Systems

WiSe 2024/25

Many-to-many Polygon Matching

Alexander Naumann
Institut für Geodäsie und Geoinformation
Universität Bonn

Motivation: Why Polygon Matching?

source: Google Maps

Motivation: Why Polygon Matching?

source: Google Maps

OSM

Motivation: Why Polygon Matching?

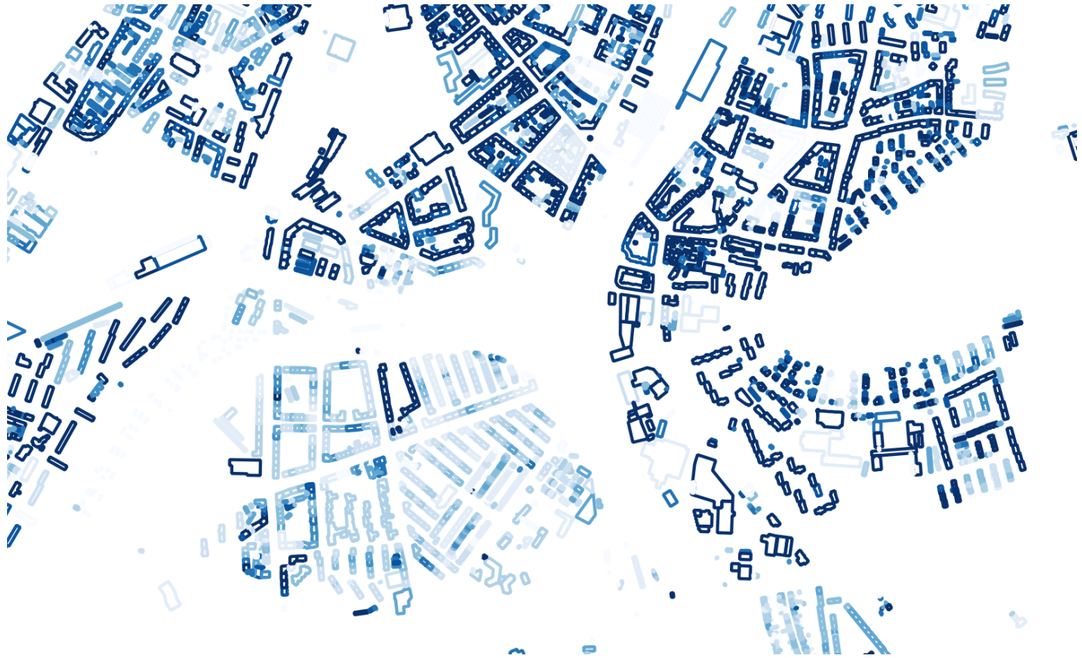
source: Google Maps

OSM ALKIS

Motivation: Why Polygon Matching?

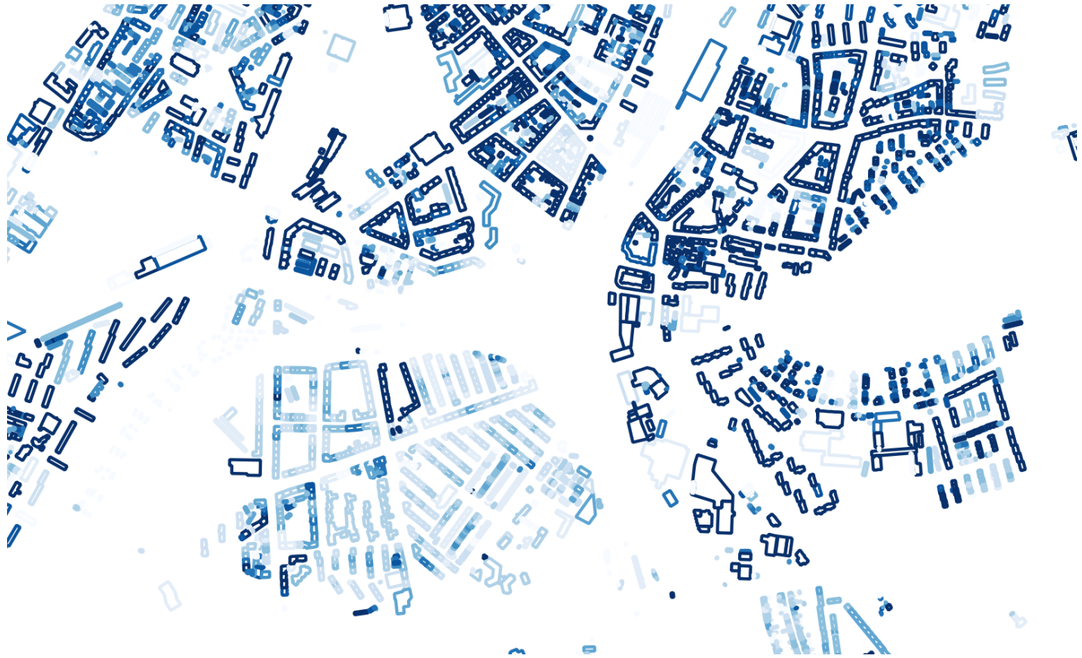
Motivation: Why Polygon Matching?

- **Evaluating the Completeness/Quality**

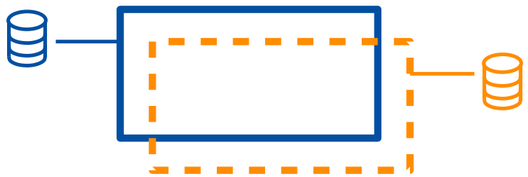


Motivation: Why Polygon Matching?

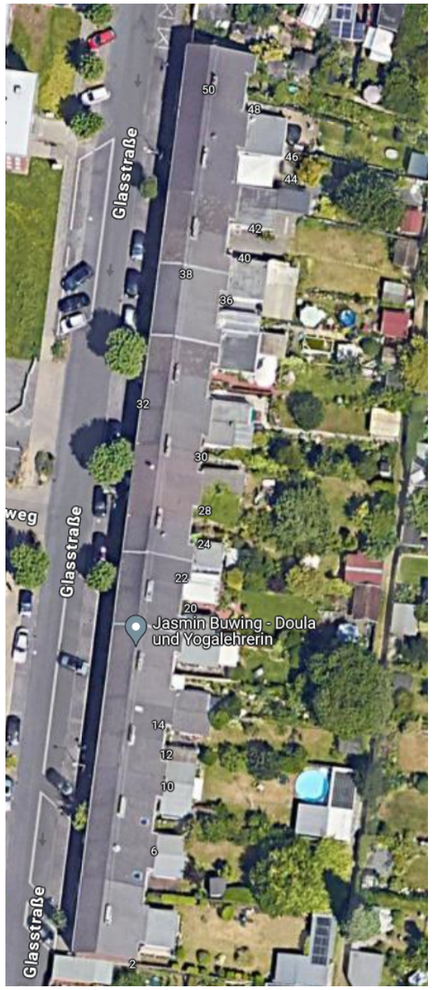
- **Evaluating the Completeness/Quality**



- **Integrating data**



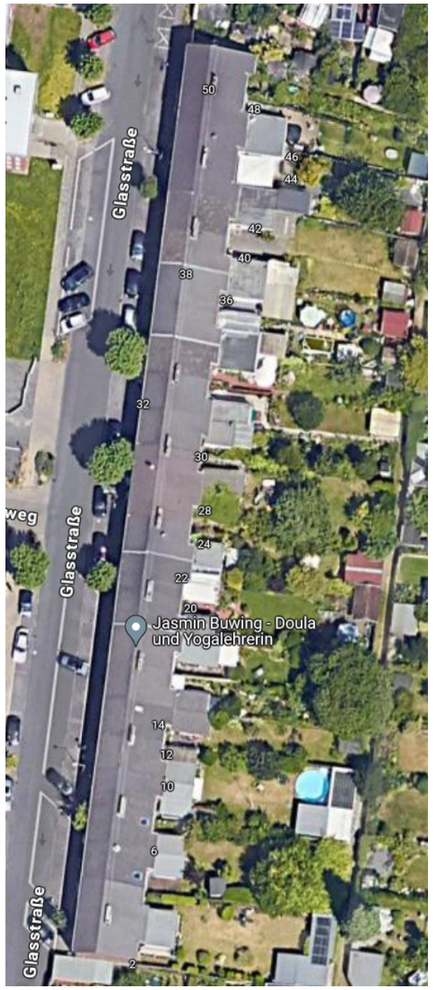
Why does 1 : 1 (or 1 : n) not suffice?



Cologne (Porz)

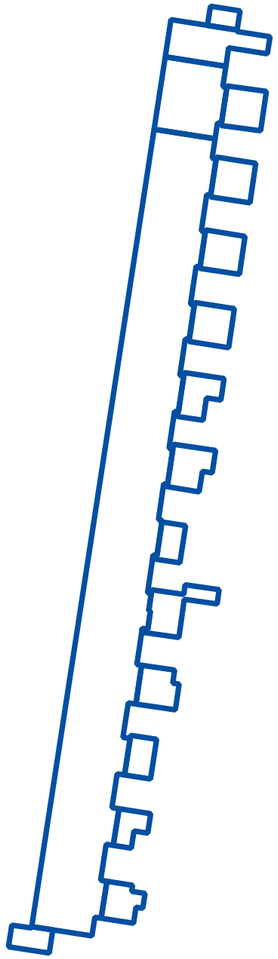
source: Google Maps

Why does 1 : 1 (or 1 : n) not suffice?



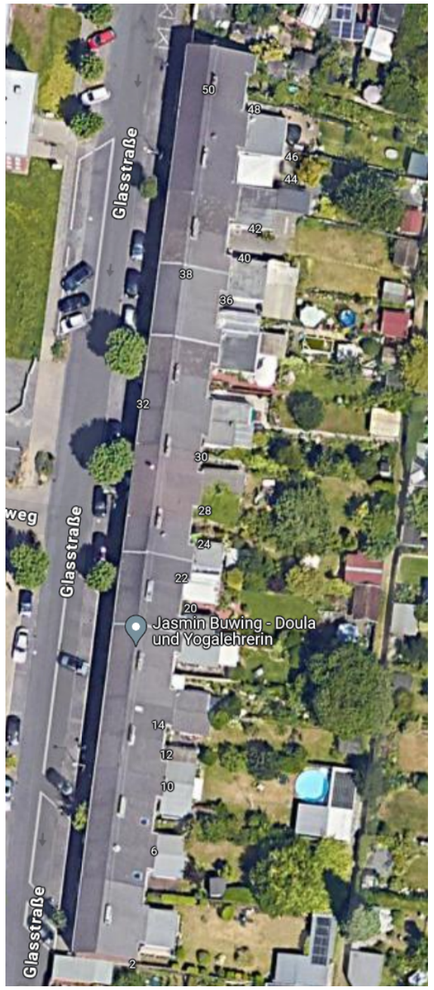
Cologne (Porz)

source: Google Maps



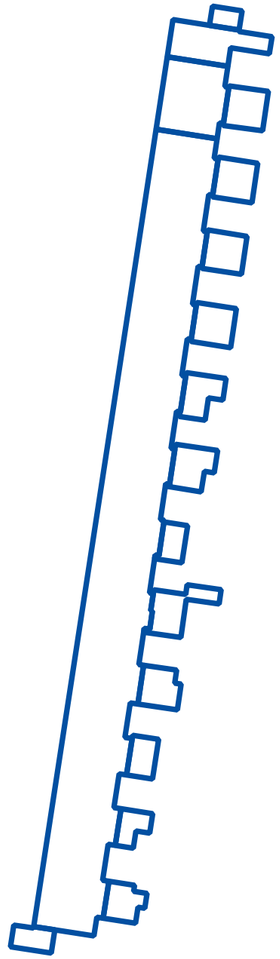
ALKIS

Why does 1 : 1 (or 1 : n) not suffice?

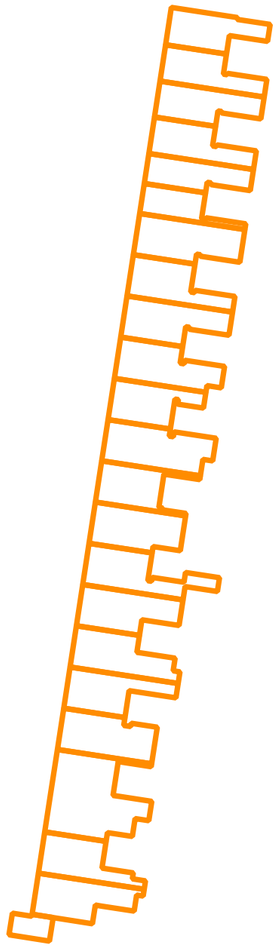


Cologne (Porz)

source: Google Maps



ALKIS



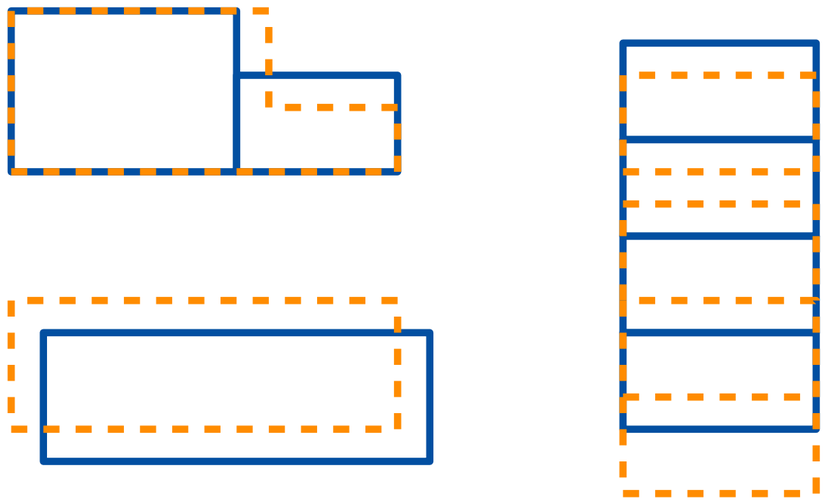
OSM

Terminology - Input

- two sets of polygons B_1 and B_2
- polygons of each B_i are interior-disjoint

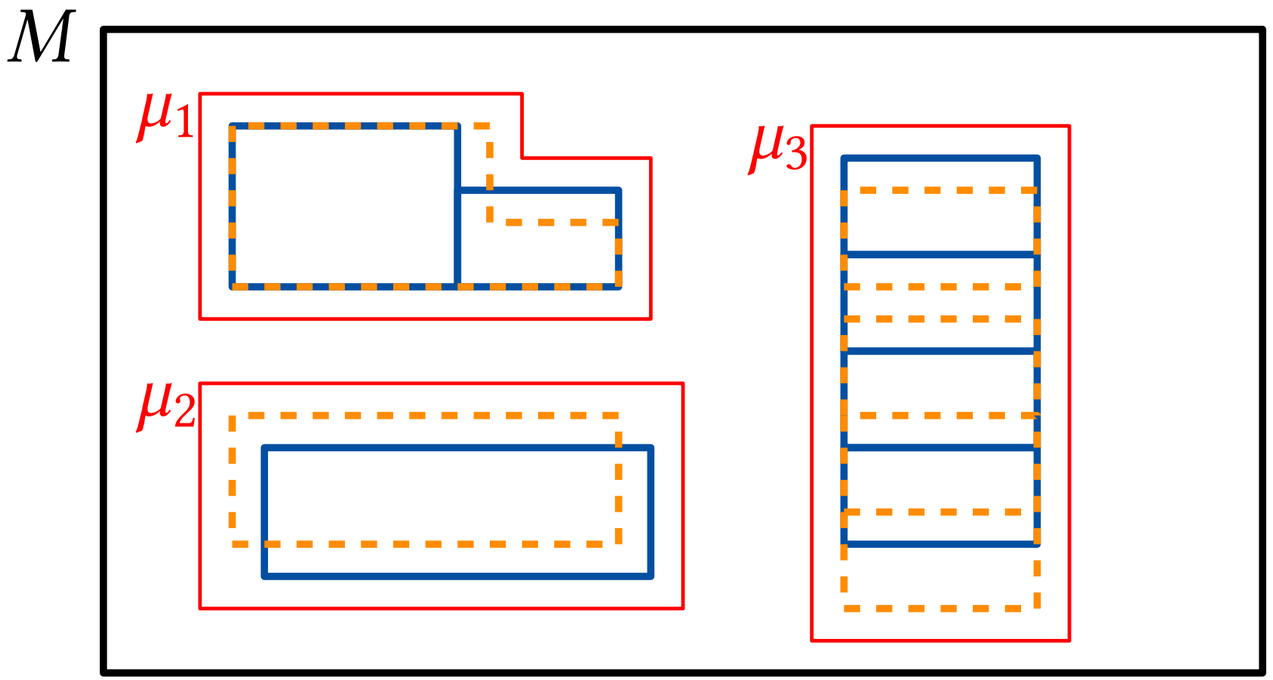
Terminology - Input

- two sets of polygons B_1 and B_2
- polygons of each B_i are interior-disjoint



Terminology - Matching

- For $P_\mu \subseteq B_1,\ Q_\mu \subseteq B_2$, we call a *match* $\mu = \{P_\mu, Q_\mu\}$.
- A *matching* $M = \{\mu_1, ..., \mu_n\}$ includes every polygon of B_1, B_2 at most once.



Related Work - Heuristics

- Fan et al. (2014):
  - let $G_{>0.3}$ be a graph with one node per polygon in B_1, B_2
  - an edge $e = \{p, q\}$ with $p \in B_1$, $q \in B_2$ exists iff:

$$\frac{\mathrm{I}(p, q)}{\min(\mathrm{area}(p), \mathrm{area}(q))} > 0.3$$

  - return connected components of $G_{>0.3}$ as matches m

Related Work - Heuristics

- Fan et al. (2014):
  - return connected components of $G_{>0.3}$ as matches m

Related Work - Heuristics

- Fan et al. (2014):
  - return connected components of $G_{>0.3}$ as matches m

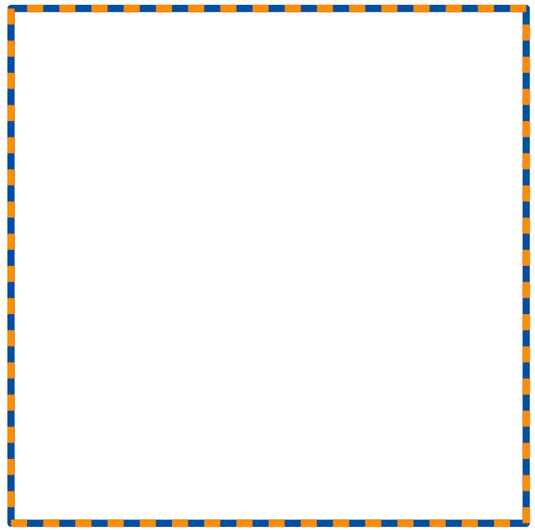
- **Disadvantage**: only evaluated on single polygon level!

Related Work - Heuristics

- Fan et al. (2014):
  - return connected components of $G_{>0.3}$ as matches m

- **Disadvantage**: only evaluated on single polygon level!

1 : 1

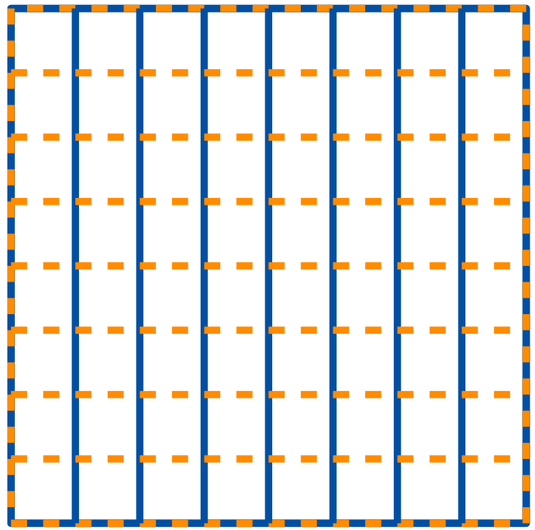


Related Work - Heuristics

- Fan et al. (2014):
  - return connected components of $G_{>0.3}$ as matches m

- **Disadvantage**: only evaluated on single polygon level!

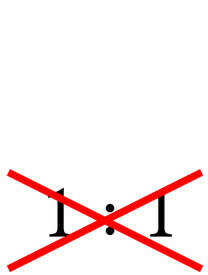
~~1:1~~

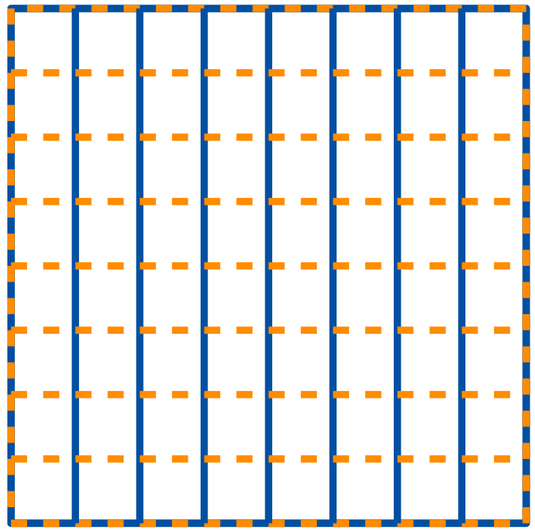


Related Work - Heuristics

- Fan et al. (2014):
  - return connected components of $G_{>0.3}$ as matches m

- **Disadvantage**: only evaluated on single polygon level!



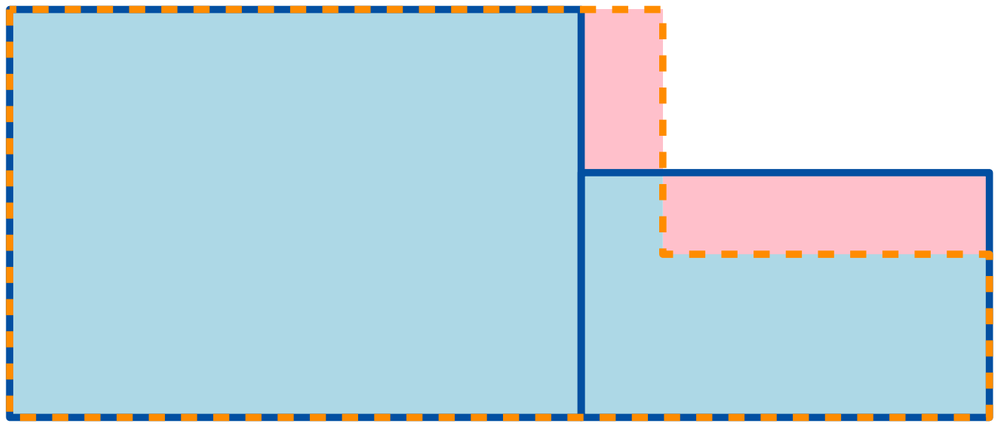


Our Contribution:
Optimal Many-To-Many Matching

The Jaccard Index

- We will make use of the Jaccard Index (aka Intersection over Union / IoU), which for two sets of polygons P, Q is defined as:

$$\mathrm{IoU}(P,Q) = \frac{\mathrm{I}(P,Q)}{\mathrm{U}(P,Q)} = \frac{\mathrm{area}(P \cap Q)}{\mathrm{area}(P \cup Q)} \tag{5}$$



Problem Formulation

- Let $\lambda \in [0, 1]$. Find a matching M^* that fulfills:

$$M^* = \text{argmax}_{M \text{ is matching}} \sum_{\mu \in M} (\text{IoU}(P_\mu, Q_\mu) - \lambda) \tag{1}$$

Problem Formulation

- Let $\lambda \in [0, 1]$. Find a matching M^* that fulfills:

$$M^* = \text{argmax}_{M \text{ is matching}} \sum_{\mu \in M} (\text{IoU}(P_\mu, Q_\mu) - \lambda) \tag{2}$$

- The decision-variant of MATCHING is NP-complete. (We proved that PARTITION $\subseteq_p$ MATCHING)

Problem Formulation

- Let $\lambda \in [0, 1]$. Find a matching M^* that fulfills:

$$M^* = \text{argmax}_{M \text{ is matching}} \sum_{\mu \in M} (\text{IoU}(P_\mu, Q_\mu) - \lambda) \tag{4}$$

- The decision-variant of MATCHING is NP-complete. (We proved that PARTITION $\subseteq_p$ MATCHING)
- Why λ?

Problem Formulation

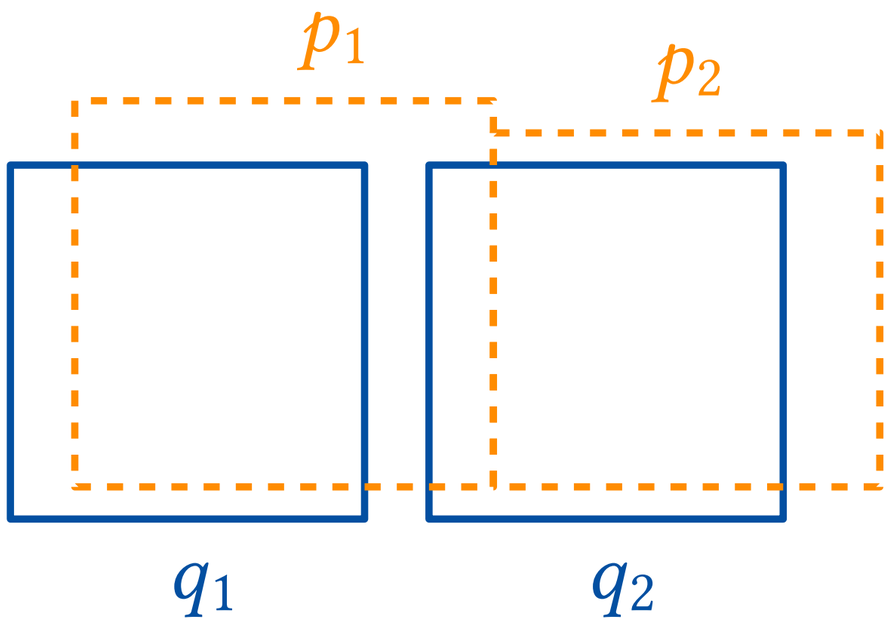
- Let $\lambda \in [0, 1]$. Find a matching M^* that fulfills:

$$M^* = \text{argmax}_{M \text{ is matching}} \sum_{\mu \in M} (\text{IoU}(P_\mu, Q_\mu) - \lambda) \tag{3}$$

- The decision-variant of MATCHING is NP-complete. (We proved that PARTITION $\subseteq_p$ MATCHING)
- Why λ?
  - filter bad matches
  - control matching granularity

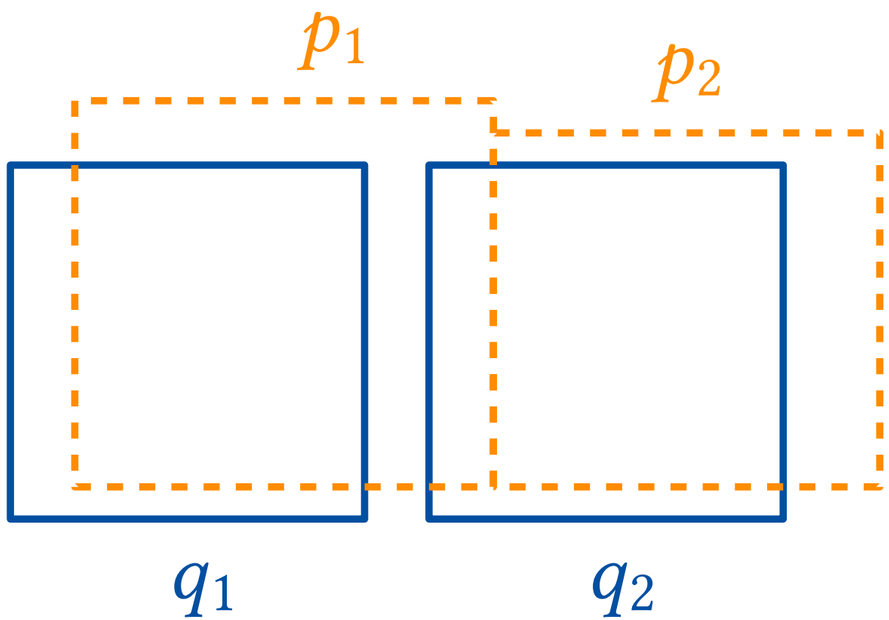
Modeling as a Graph Problem

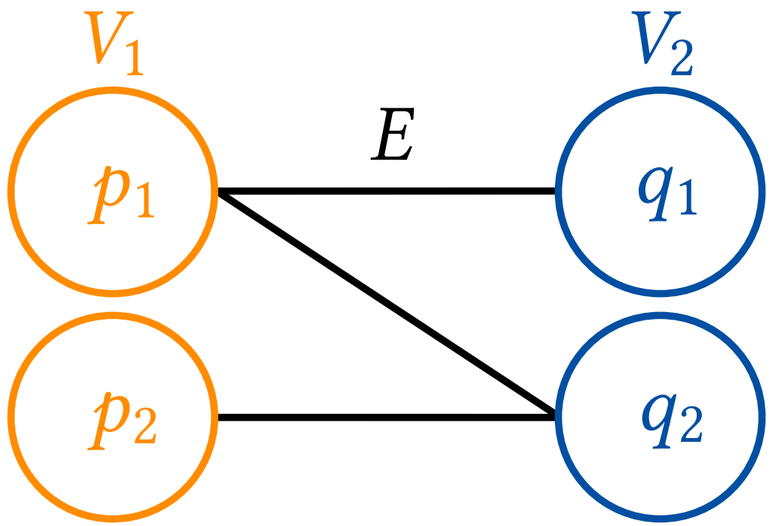
- geometric problem ↔ graph problem
- intersection graph G_{int}:



Modeling as a Graph Problem

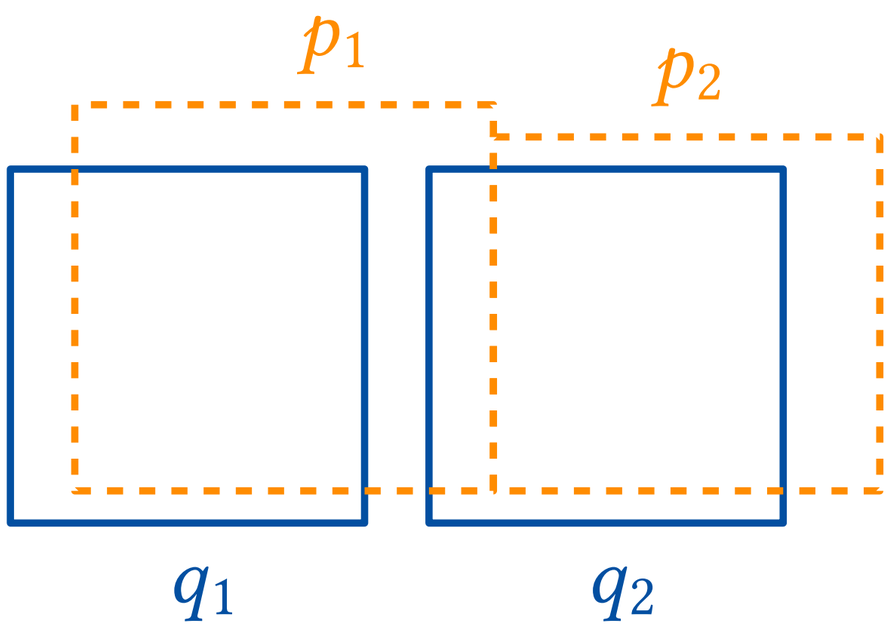
- geometric problem ↔ graph problem
- intersection graph G_{int}:

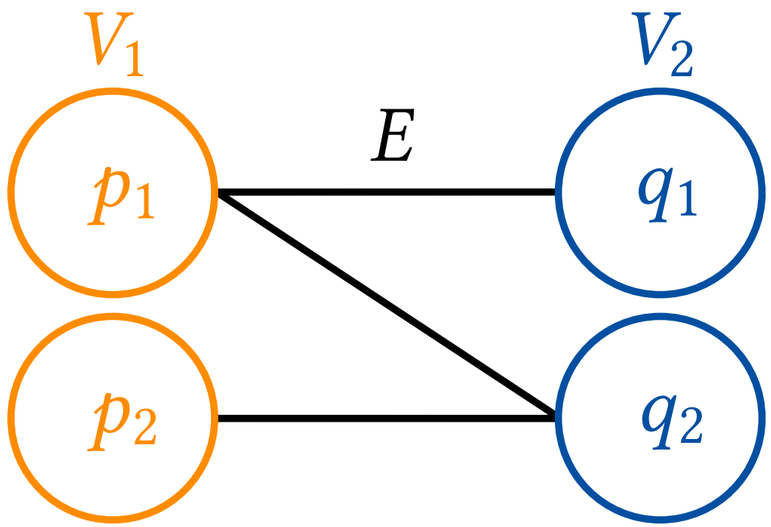




Modeling as a Graph Problem

- geometric problem ↔ graph problem
- intersection graph G_{int}:

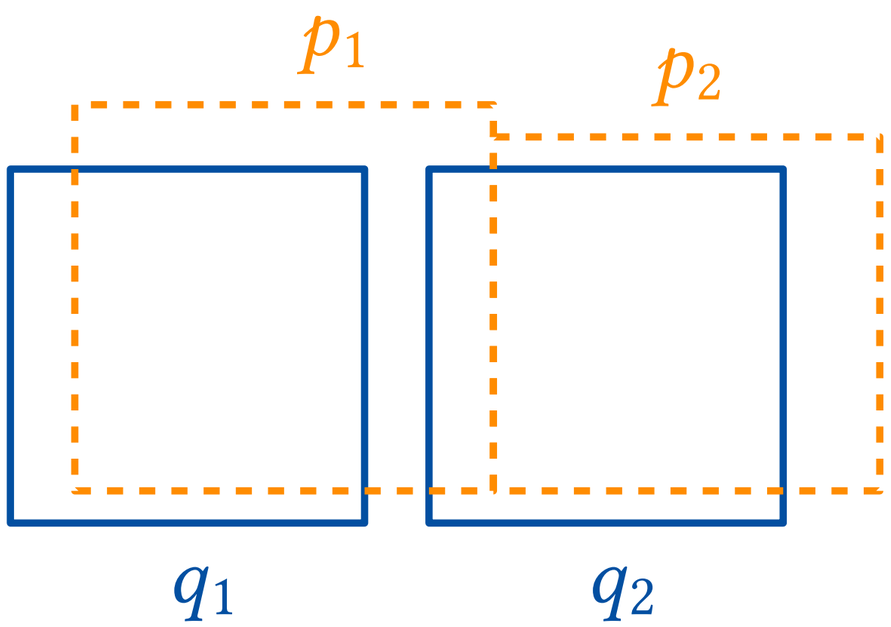


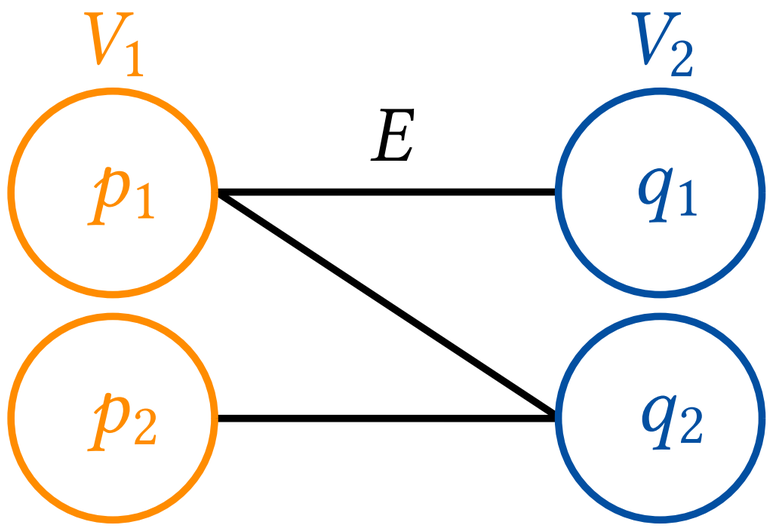


(→ maximum bipartite matching (Keles, Master Thesis 2023))

Modeling as a Graph Problem

- geometric problem ↔ graph problem
- intersection graph G_{int}:



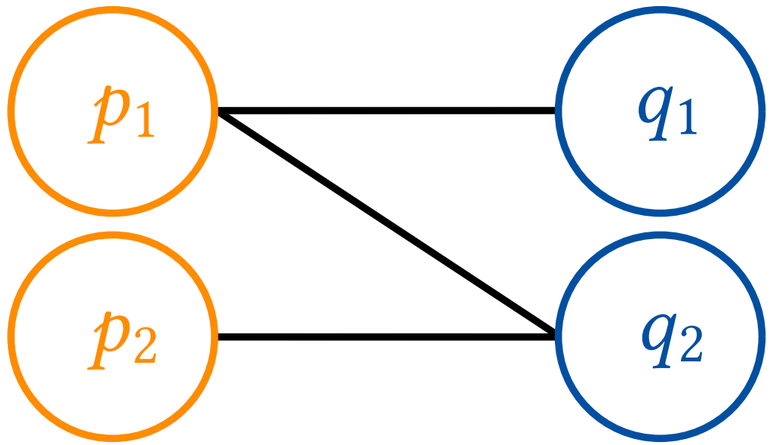


(→ maximum bipartite matching (Keles, Master Thesis 2023))

Problem: This does not suffice to find many-to-many matches!

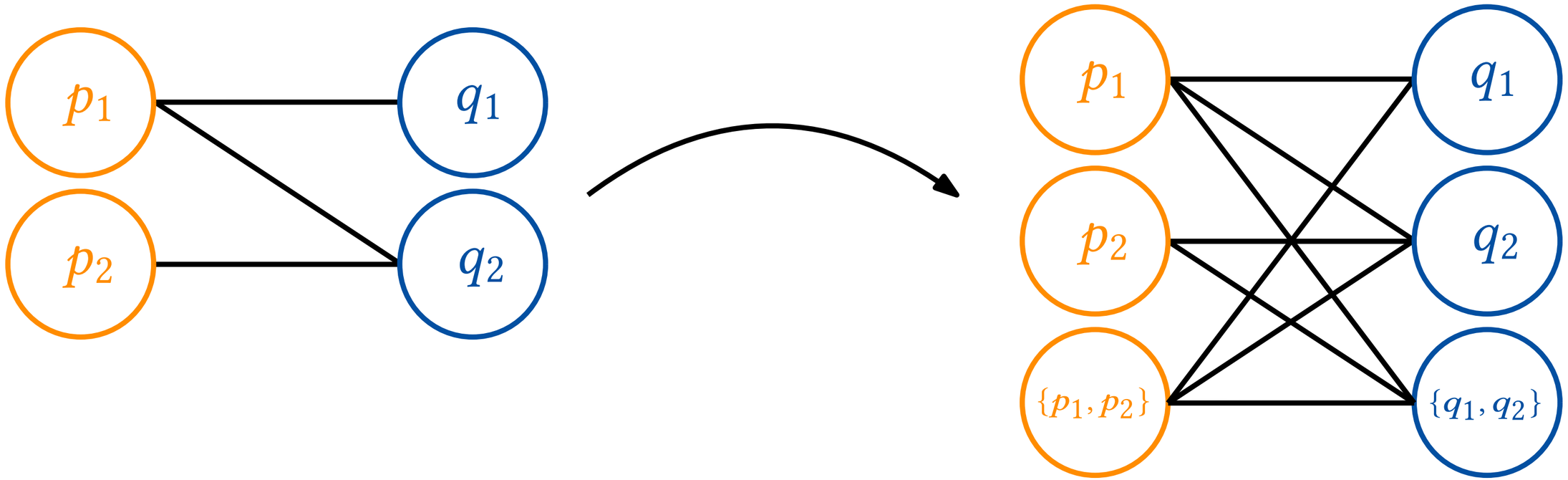
Model - Candidate Graph

- model expansion: a vertex can now also represent a set of polygons. (*cumulative vertex*)
- graph G containing an optimal subset of edges $E^* \subseteq E \leftrightarrow$ *candidate graph*.



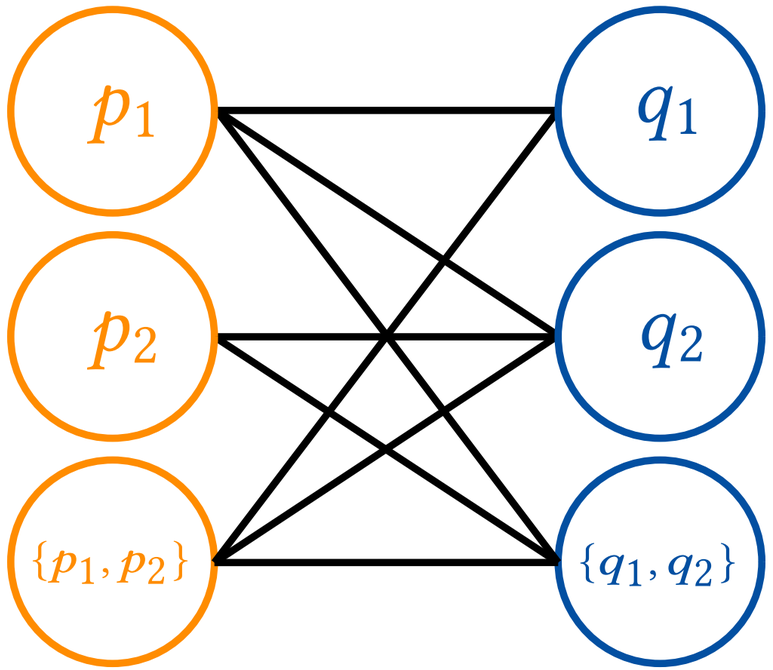
Model - Candidate Graph

- model expansion: a vertex can now also represent a set of polygons. (*cumulative vertex*)
- graph G containing an optimal subset of edges $E^* \subseteq E \leftrightarrow$ *candidate graph.*



Solving the (Graph) Problem

- *constrained maximum bipartite matching* on the candidate graph: no polygon should be matched twice
- can be solved using an Integer Linear Program



Solving the (Graph) Problem

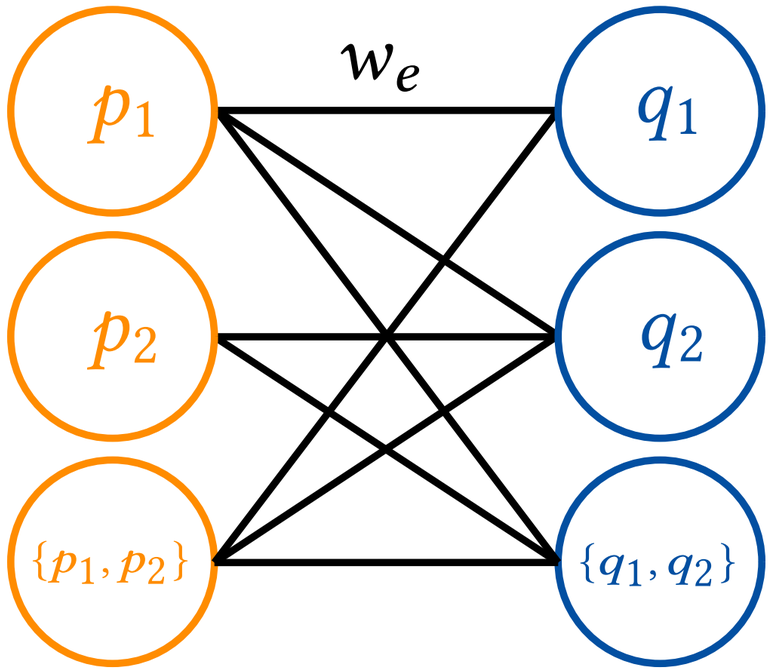
- *constrained maximum bipartite matching* on the candidate graph: no polygon should be matched twice
- can be solved using an Integer Linear Program

$\max \sum_{e \in E}(x_e \cdot w_e)$ s.t.

$$\forall e \in E : x_e \in \{0, 1\}$$

$$\forall p \in B_1 : \sum_{e \in E_p} x_e \leq 1$$

$$\forall q \in B_2 : \sum_{e \in E_q} x_e \leq 1$$



Solving the (Graph) Problem

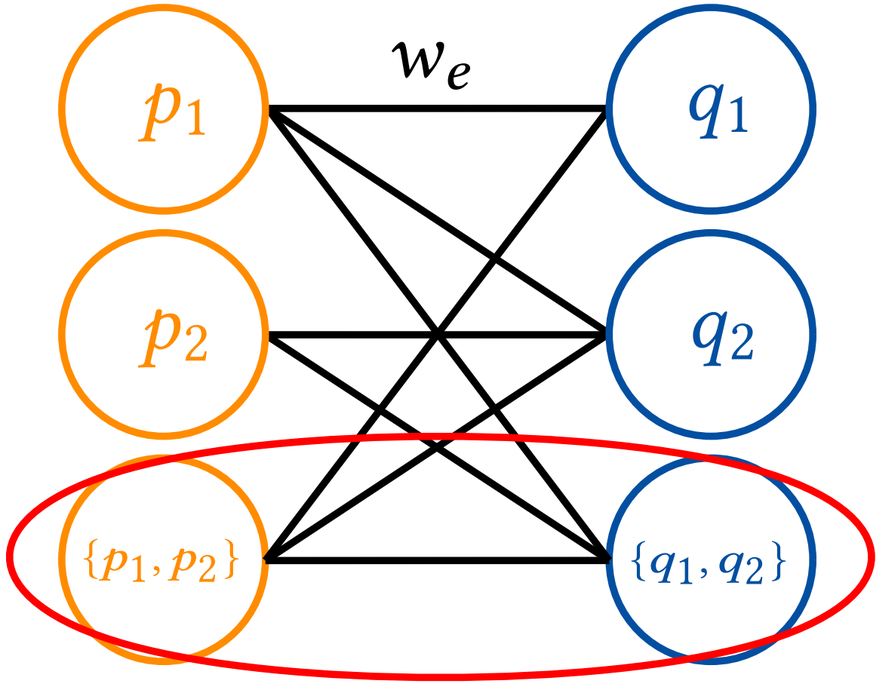
- *constrained maximum bipartite matching* on the candidate graph: no polygon should be matched twice
- can be solved using an Integer Linear Program

$\max \sum_{e \in E}(x_e \cdot w_e)$ s.t.

$$\forall e \in E : x_e \in \{0, 1\}$$

$$\forall p \in B_1 : \sum_{e \in E_p} x_e \leq 1$$

$$\forall q \in B_2 : \sum_{e \in E_q} x_e \leq 1$$



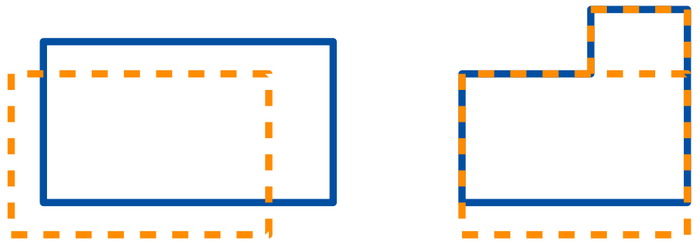
The combinatorial combinations of polygons grow exponentially!

Structural Properties - Lemmas

1. **Match Connectedness**.
2. **Outlier Polygons.**
3. Polygons (not) in the same Match.
4. 1:1 Matches.

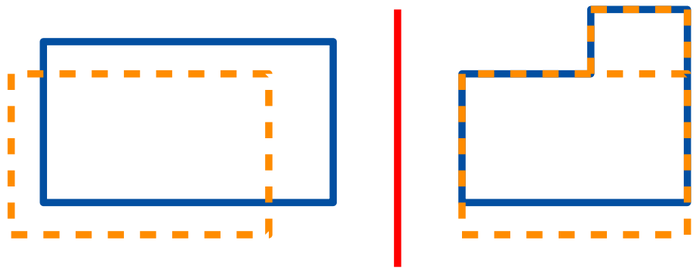
Structural Properties - Match Connectedness

Lemma. Given two sets of polygons B_1, B_2, every matching M^* maximizing $\sigma(M^*)$ exclusively consists of connected matches.



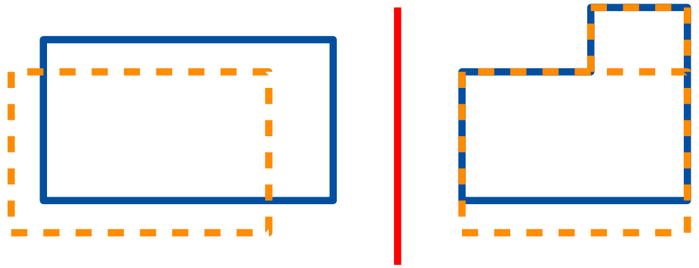
Structural Properties - Match Connectedness

Lemma. Given two sets of polygons B_1, B_2, every matching M^* maximizing $\sigma(M^*)$ exclusively consists of connected matches.



Structural Properties - Match Connectedness

Lemma. Given two sets of polygons B_1, B_2, every matching M^* maximizing $\sigma(M^*)$ exclusively consists of connected matches.



Proof Sketch. Via contradiction: Assume a match $\mu \in M^*$ was disconnected and consists of two disjoint matches μ_1, μ_2.

Case 1. $\sigma(\mu_1) = \sigma(\mu_2)$

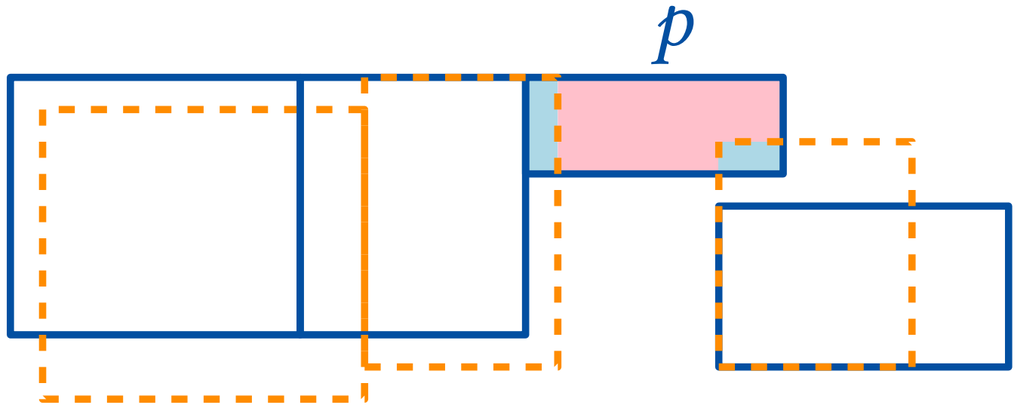
$\rightarrow$ replacing μ by μ_1 and μ_2 would increase $\sigma(M^*)$.

Case 2. w.l.o.g. $\sigma(\mu_1) < \sigma(\mu_2)$

$\rightarrow$ then $\mathrm{IoU}(\mu) < \mathrm{IoU}(\mu_2)$ and replacing μ by μ_2 would increase $\sigma(M^*)$

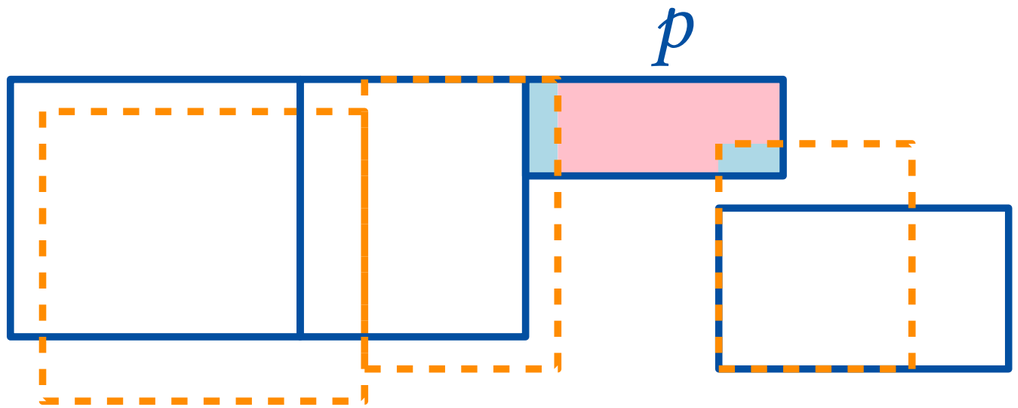
Structural Properties - Outlier Polygons

Lemma. Given two sets of polygons B_1, B_2. Let $p \in B_1$ and let Q be the subset of B_2 that consists of all polygons that intersect p. If $\frac{\mathrm{I}(p,Q)}{\mathrm{area}(p \setminus Q)} < \lambda$, then p is not part of a match in an optimal $m : n$ matching.



Structural Properties - Outlier Polygons

Lemma. Given two sets of polygons B_1, B_2. Let $p \in B_1$ and let Q be the subset of B_2 that consists of all polygons that intersect p. If $\frac{\mathrm{I}(p,Q)}{\mathrm{area}(p \setminus Q)} < \lambda$, then p is not part of a match in an optimal $m : n$ matching.



Proof Idea. It can be shown that every match p could be included in would be improved by removing p from it.

Properties - Recap

In every solution M^* minimizing $\sigma(M^*)$...

1. ... **every match μ is connected.**
2. ... **depending on λ, some polygons are unmatched.**
3. ... if certain conditions hold, two polygons p, q are always (or never) in the same match.
4. ... if certain conditions hold, two polygons p, q always form a $1:1$ -match.

Algorithm

Algorithm

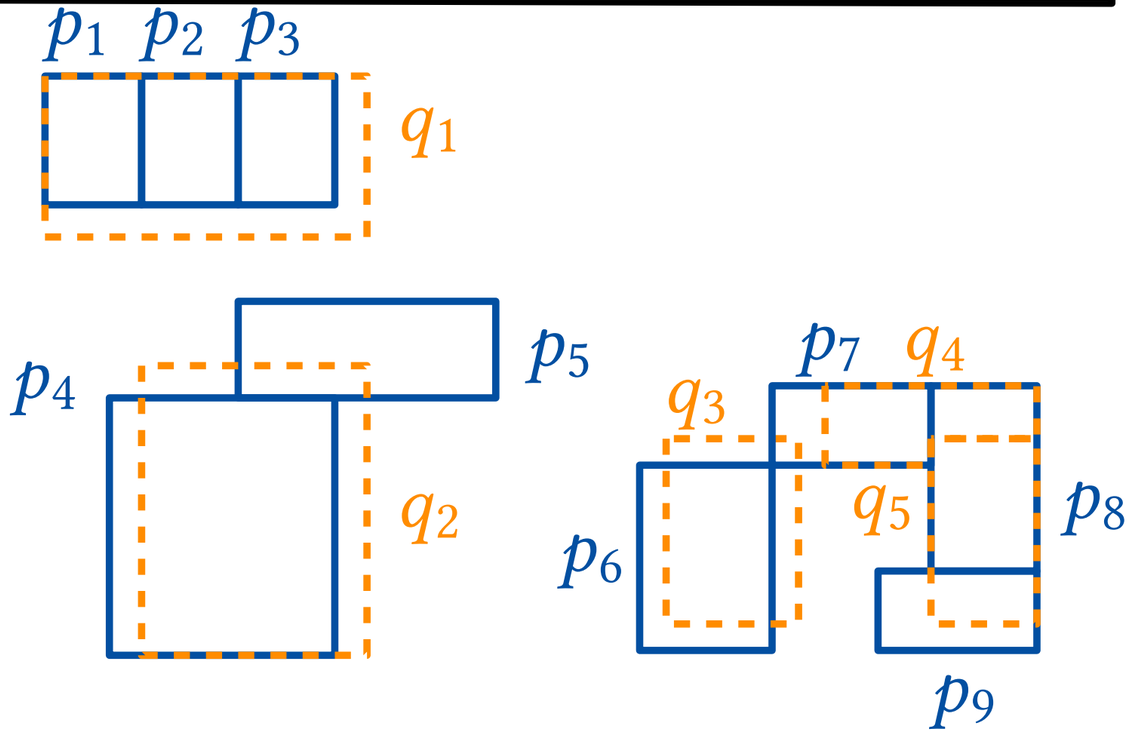
Input interior-disjoint polygon-sets $\{B_1, B_2\}$, threshold λ
Output optimal $m : n$ matching M^*

Algorithm - OPTIMAL M:N MATCHING

1. compute intersection graph w.r.t. B_1, B_2
2. for every connected component c: *(match connectedness)*
   (a) reduce c *(match connectedness, outliers, 1:1 matches)*
   (b) build candidate graph from c *(polys (not) in the same match)*
   (c) solve ILP on the remaining graph

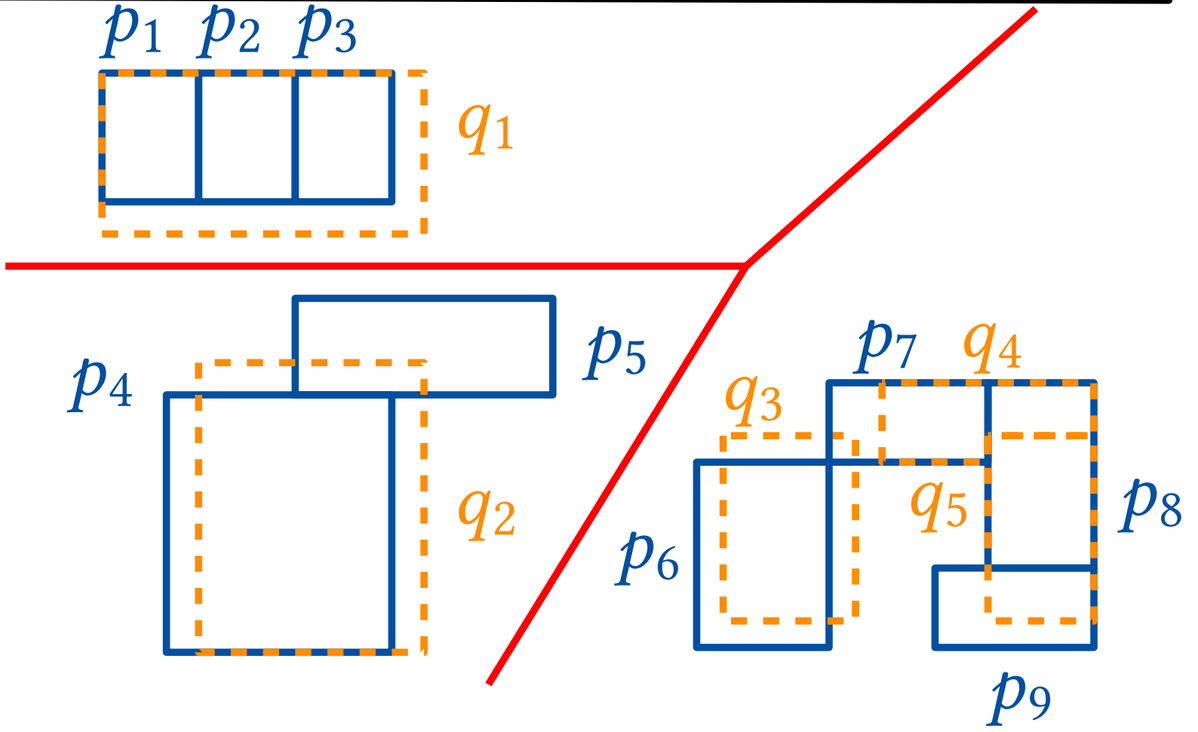
Algorithm - OPTIMAL M:N MATCHING

1. **compute intersection graph w.r.t.** B_1, B_2
2. for every connected component c: *(match connectedness)*
   (a) reduce c *(match connectedness, outliers, 1:1 matches)*
   (b) build candidate graph from c *(polys (not) in the same match)*
   (c) solve ILP on the remaining graph



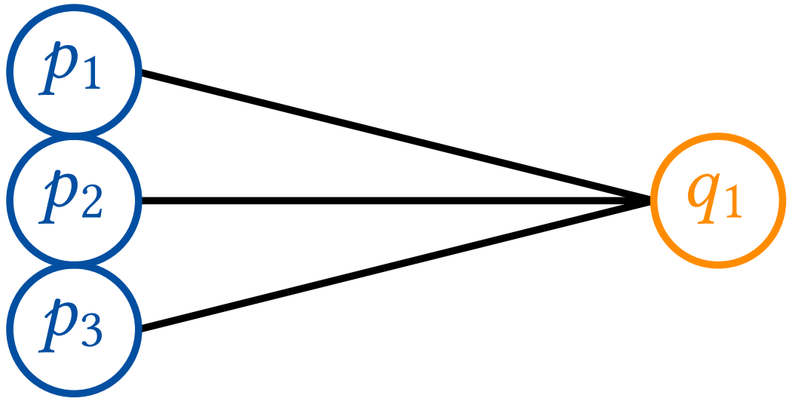
Algorithm - OPTIMAL M:N MATCHING

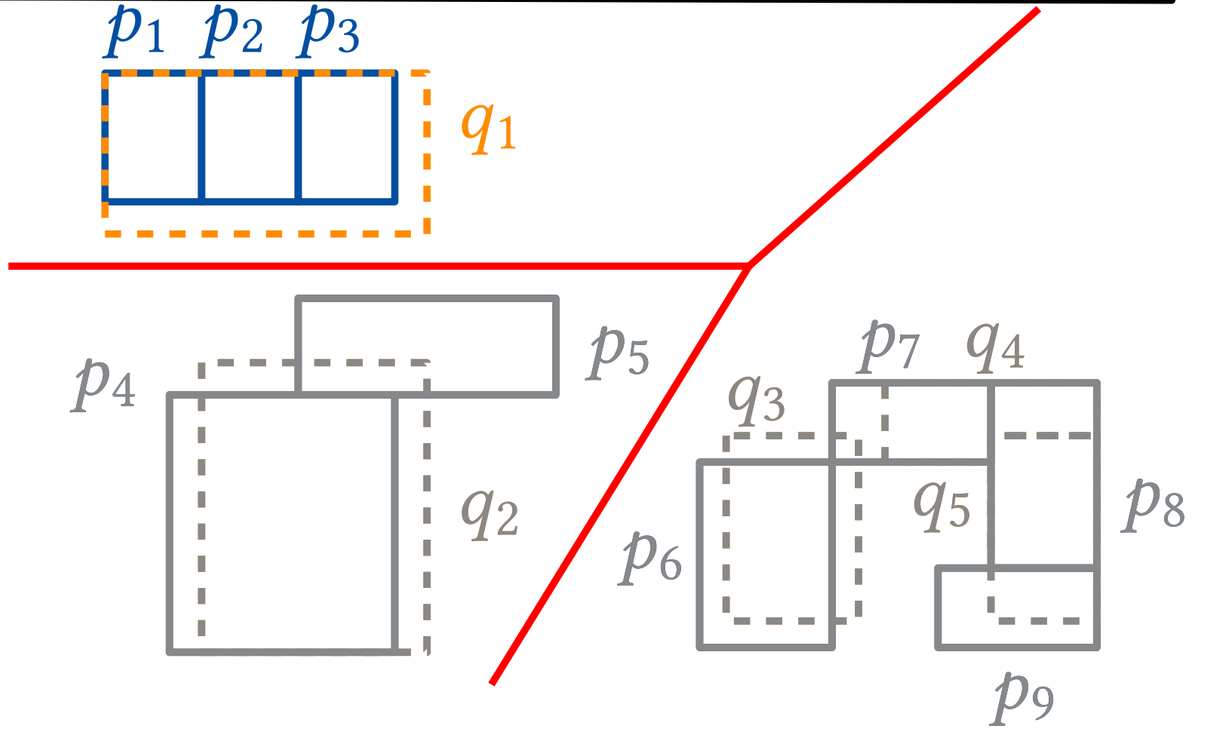
1. compute intersection graph w.r.t. B_1, B_2
2. **for every connected component** c**:** ***(match connectedness)***
   (a) reduce c *(match connectedness, outliers, 1:1 matches)*
   (b) build candidate graph from c *(polys (not) in the same match)*
   (c) solve ILP on the remaining graph



Algorithm - OPTIMAL M:N MATCHING

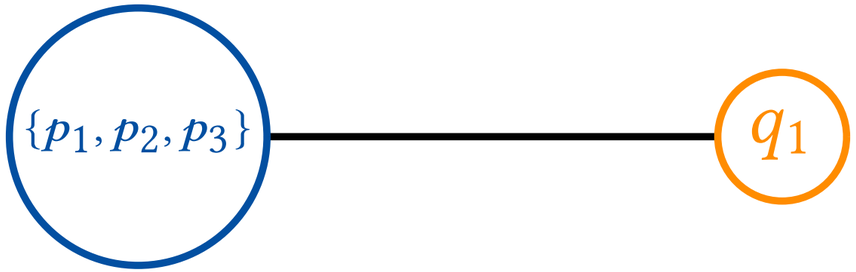
1. compute intersection graph w.r.t. B_1, B_2
2. for every connected component c: *(match connectedness)*
   (a) **reduce** c ***(match connectedness, outliers, 1:1 matches)***
   (b) build candidate graph from c *(polys (not) in the same match)*
   (c) solve ILP on the remaining graph

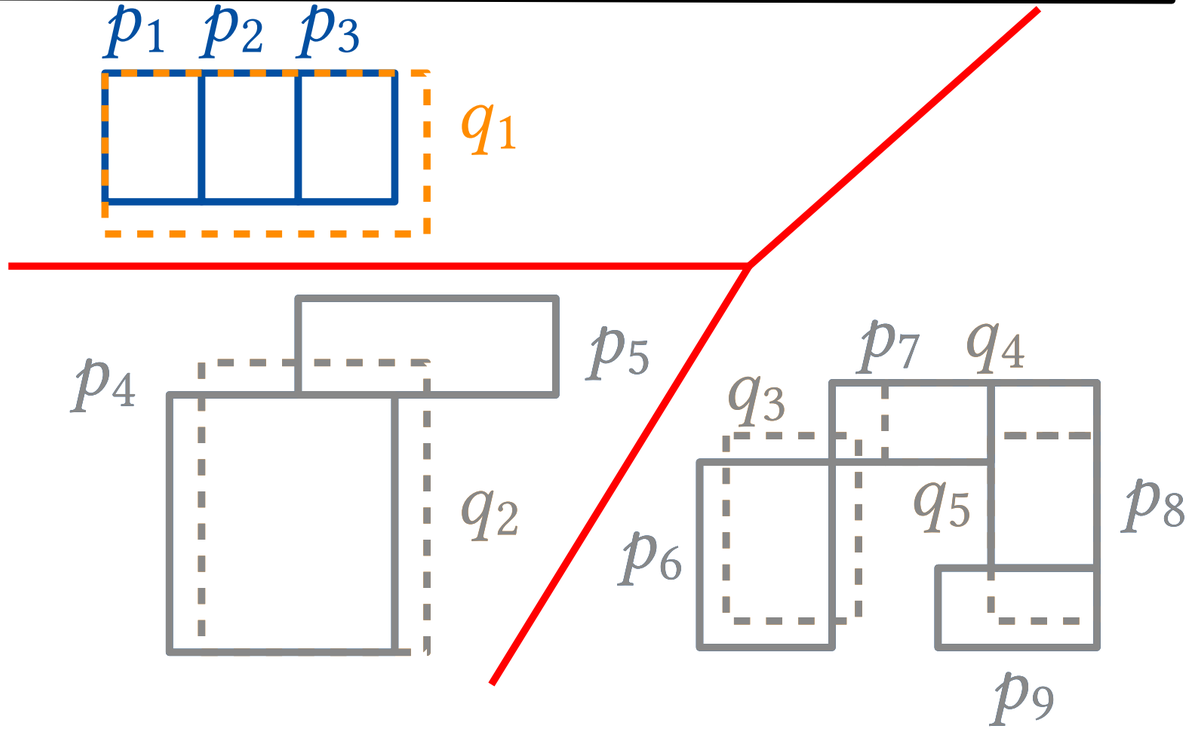




Algorithm - OPTIMAL M:N MATCHING

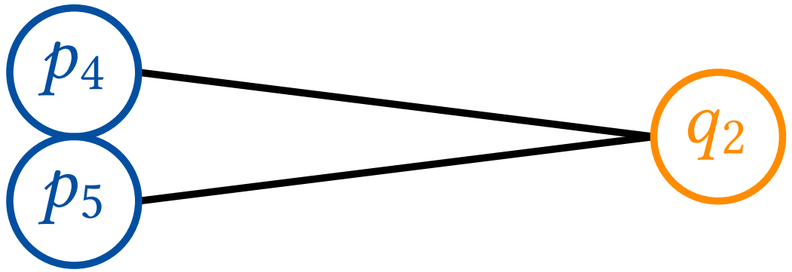
1. compute intersection graph w.r.t. B_1, B_2
2. for every connected component c: *(match connectedness)*
   (a) **reduce** c ***(match connectedness, outliers, 1:1 matches)***
   (b) build candidate graph from c *(polys (not) in the same match)*
   (c) solve ILP on the remaining graph

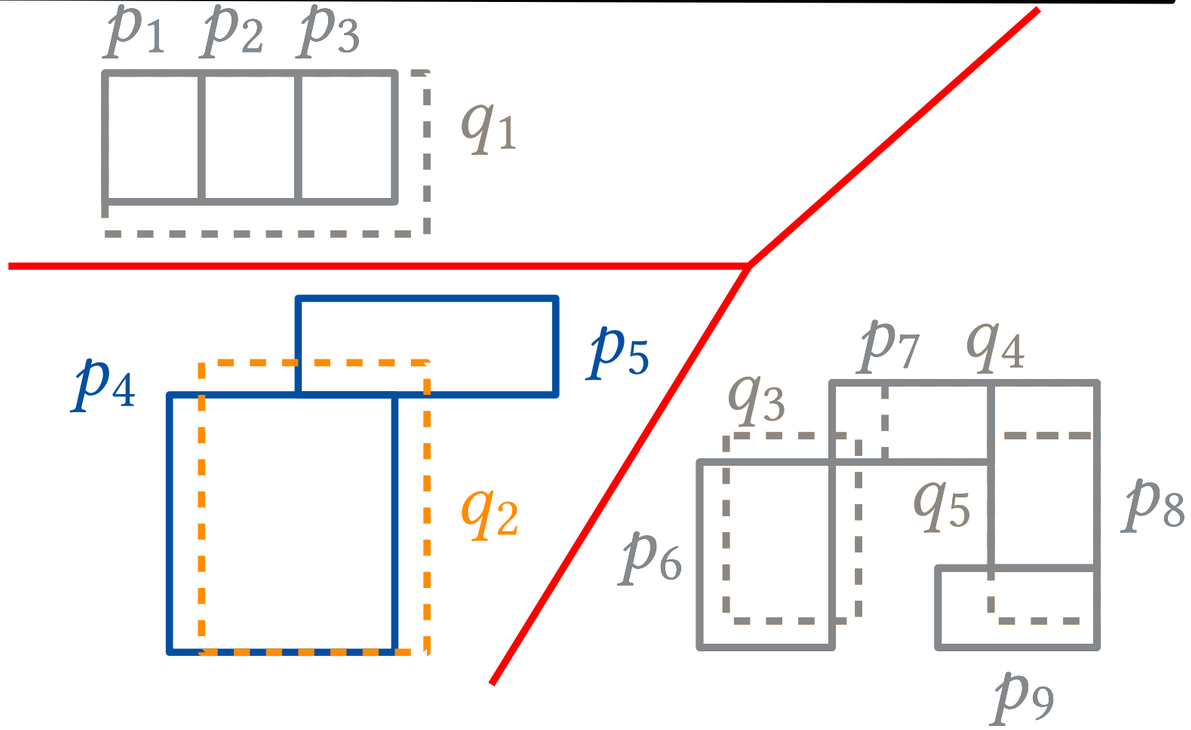




Algorithm - OPTIMAL M:N MATCHING

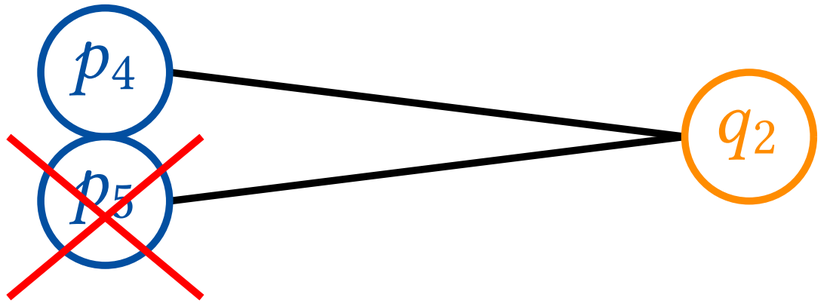
1. compute intersection graph w.r.t. B_1, B_2
2. for every connected component c: *(match connectedness)*
   (a) **reduce** c ***(match connectedness, outliers, 1:1 matches)***
   (b) build candidate graph from c *(polys (not) in the same match)*
   (c) solve ILP on the remaining graph

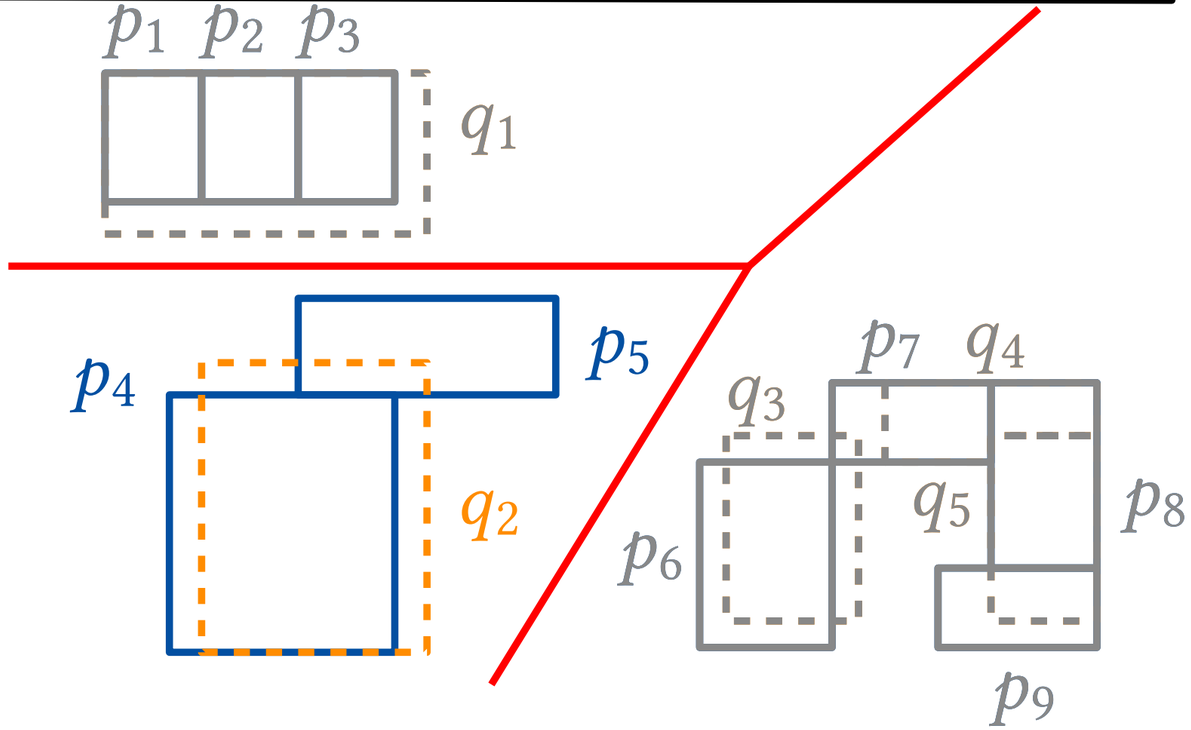




Algorithm - OPTIMAL M:N MATCHING

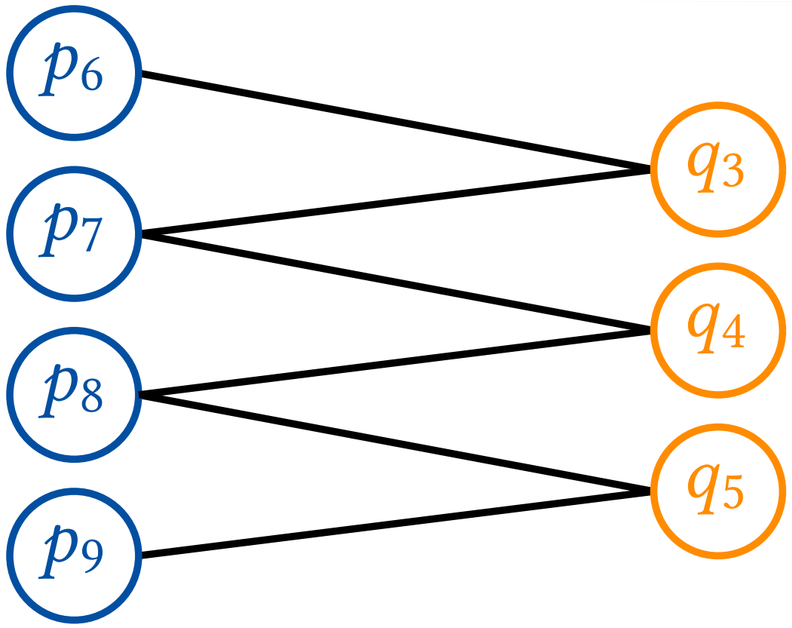
1. compute intersection graph w.r.t. B_1, B_2
2. for every connected component c: *(match connectedness)*
   (a) **reduce** c ***(match connectedness, outliers, 1:1 matches)***
   (b) build candidate graph from c *(polys (not) in the same match)*
   (c) solve ILP on the remaining graph

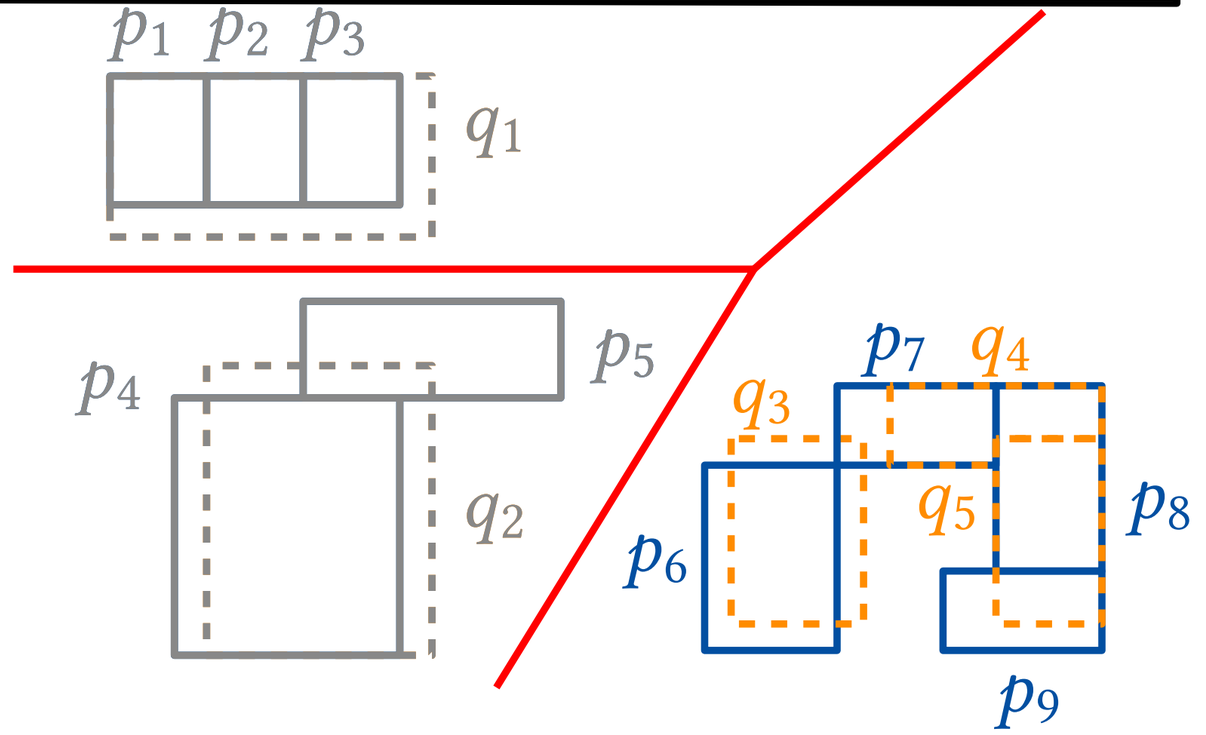




Algorithm - OPTIMAL M:N MATCHING

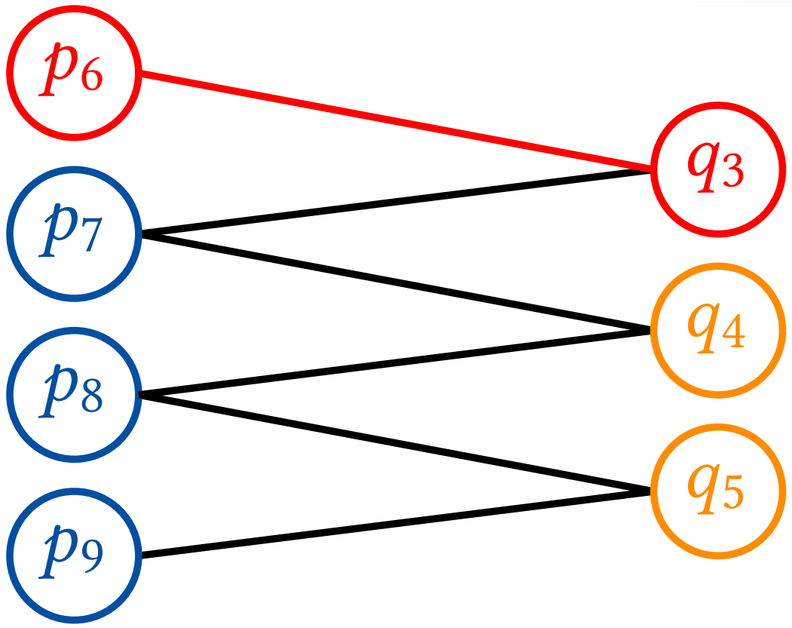
1. compute intersection graph w.r.t. B_1, B_2
2. for every connected component c: *(match connectedness)*
   (a) **reduce** c ***(match connectedness, outliers, 1:1 matches)***
   (b) build candidate graph from c *(polys (not) in the same match)*
   (c) solve ILP on the remaining graph

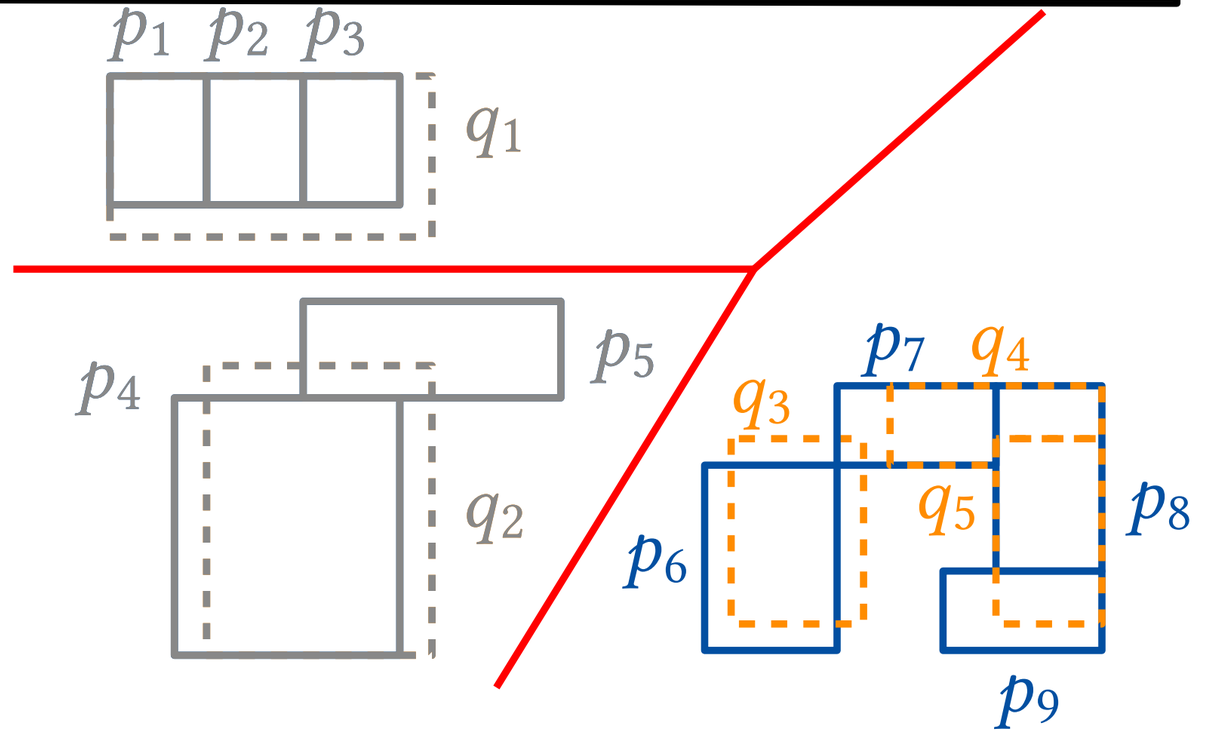




Algorithm - OPTIMAL M:N MATCHING

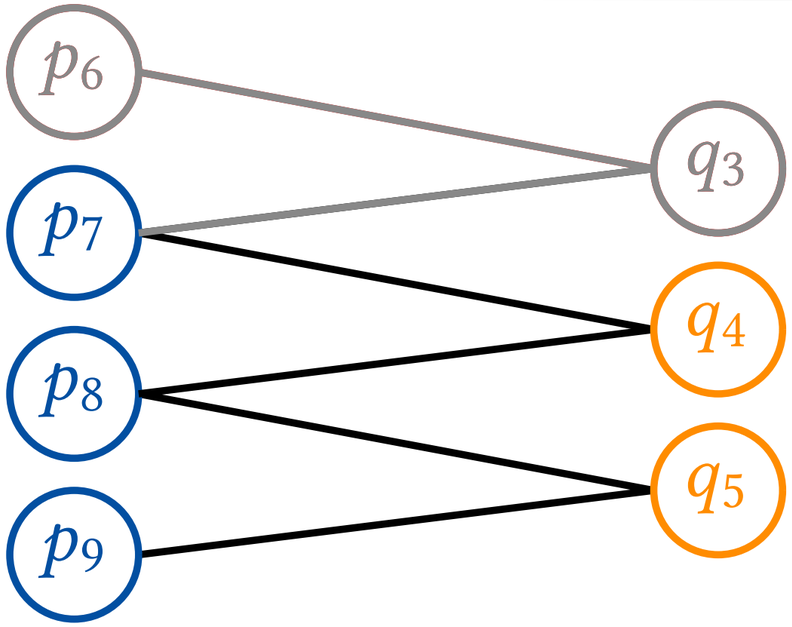
1. compute intersection graph w.r.t. B_1, B_2
2. for every connected component c: *(match connectedness)*
   (a) **reduce** c ***(match connectedness, outliers, 1:1 matches)***
   (b) build candidate graph from c *(polys (not) in the same match)*
   (c) solve ILP on the remaining graph

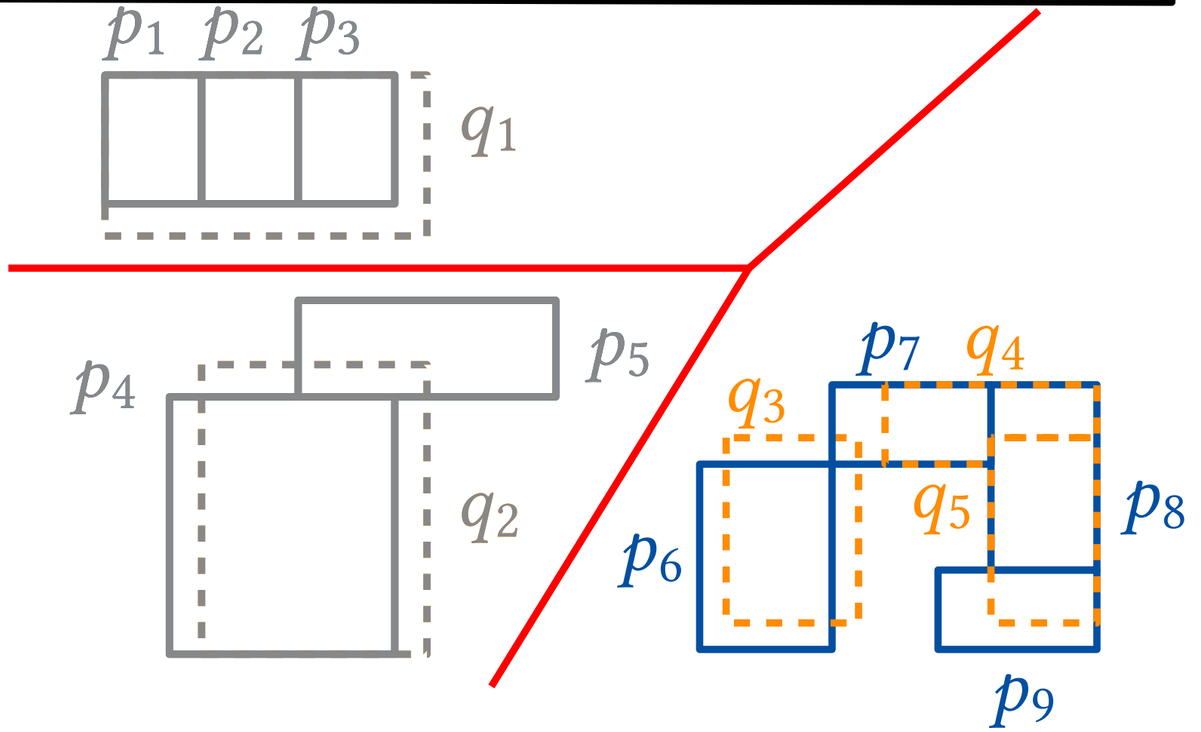




Algorithm - OPTIMAL M:N MATCHING

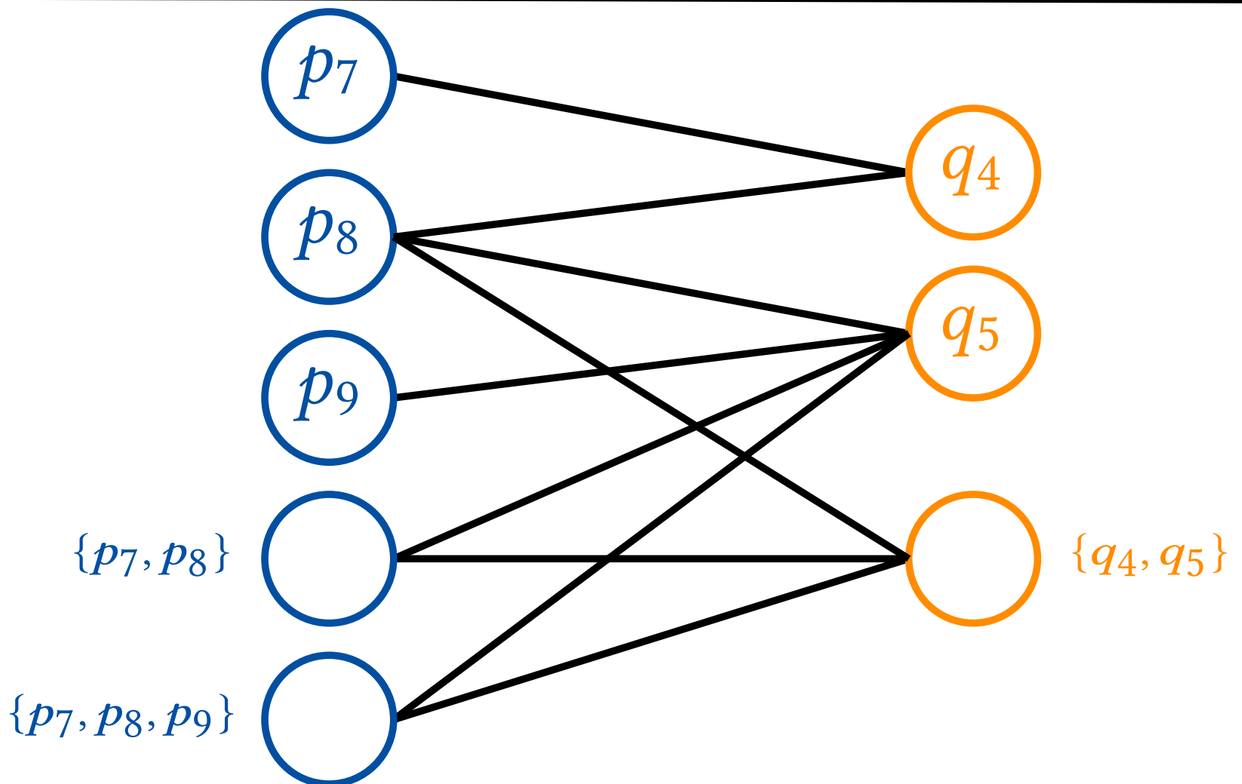
1. compute intersection graph w.r.t. B_1, B_2
2. for every connected component c: *(match connectedness)*
   (a) **reduce** c ***(match connectedness, outliers, 1:1 matches)***
   (b) build candidate graph from c *(polys (not) in the same match)*
   (c) solve ILP on the remaining graph

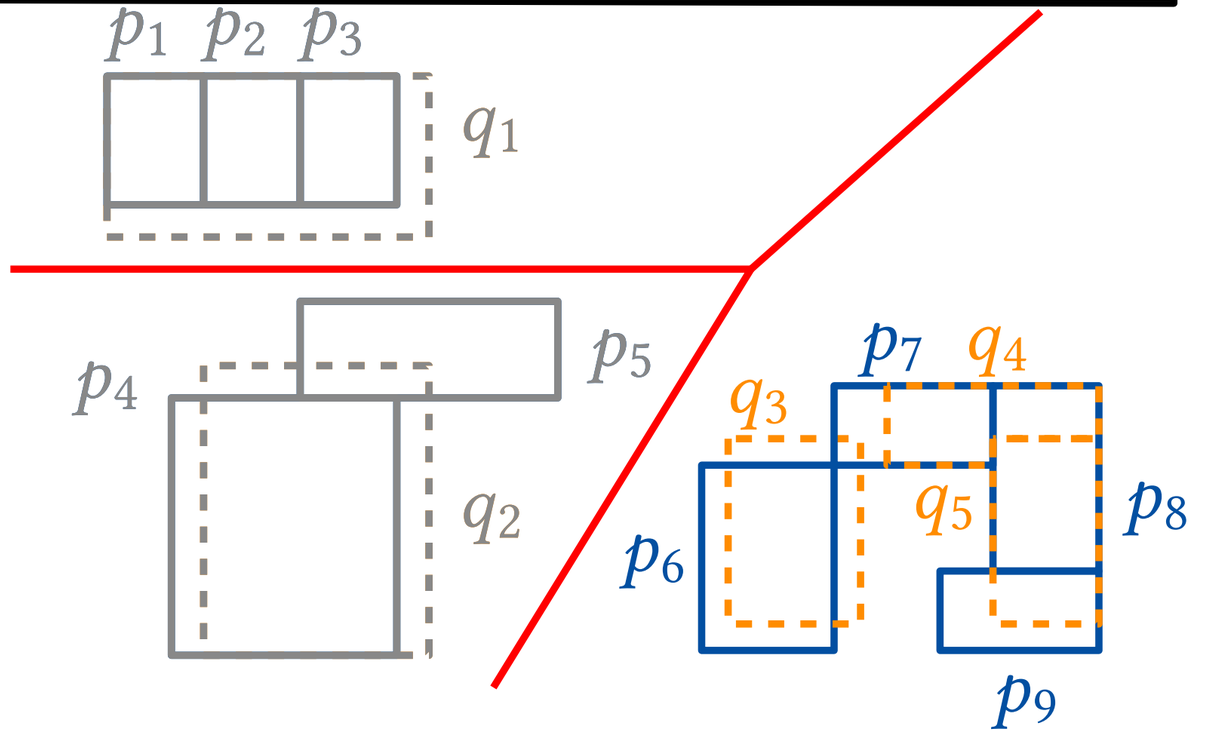




Algorithm - OPTIMAL M:N MATCHING

1. compute intersection graph w.r.t. B_1, B_2
2. for every connected component c: *(match connectedness)*
   (a) reduce c *(match connectedness, outliers, 1:1 matches)*
   (b) **build candidate graph from c *(polys (not) in the same match)***
   (c) solve ILP on the remaining graph





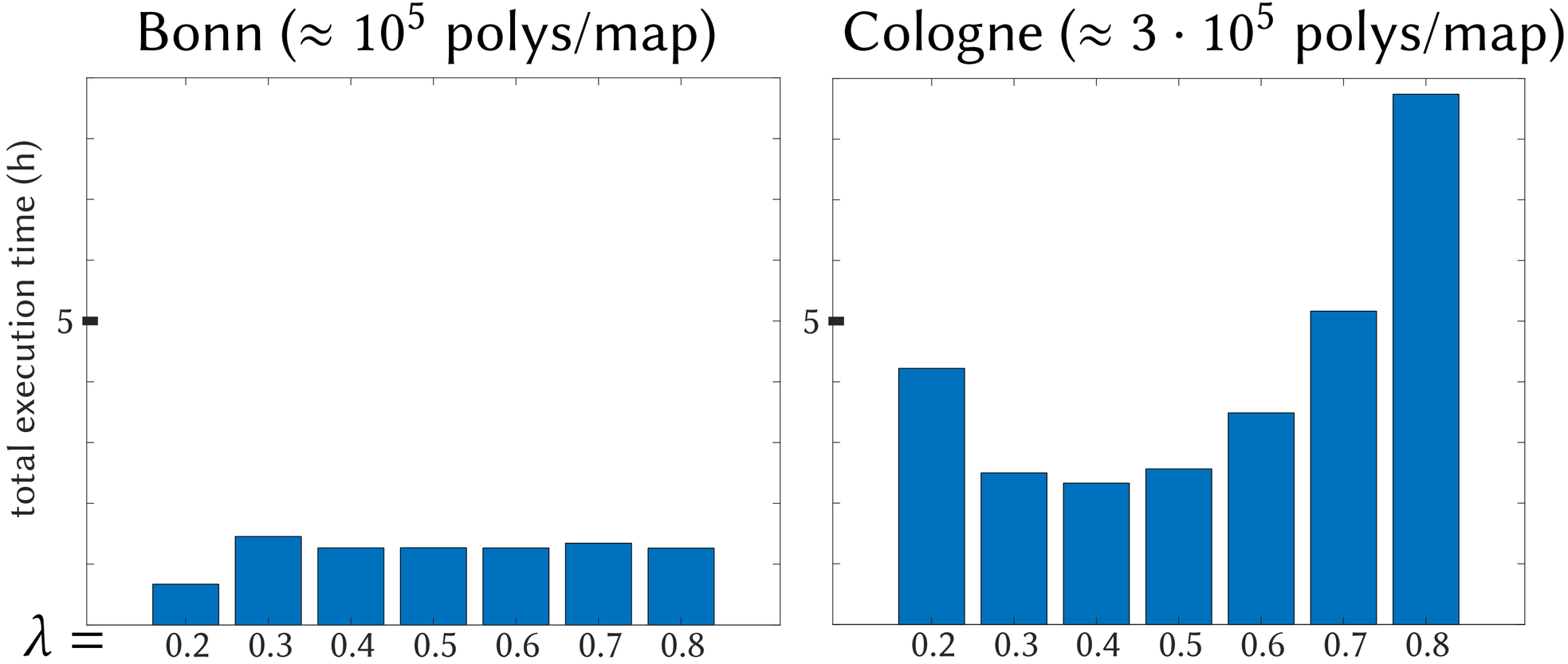
Evaluation

Experimental Evaluation - Running Time

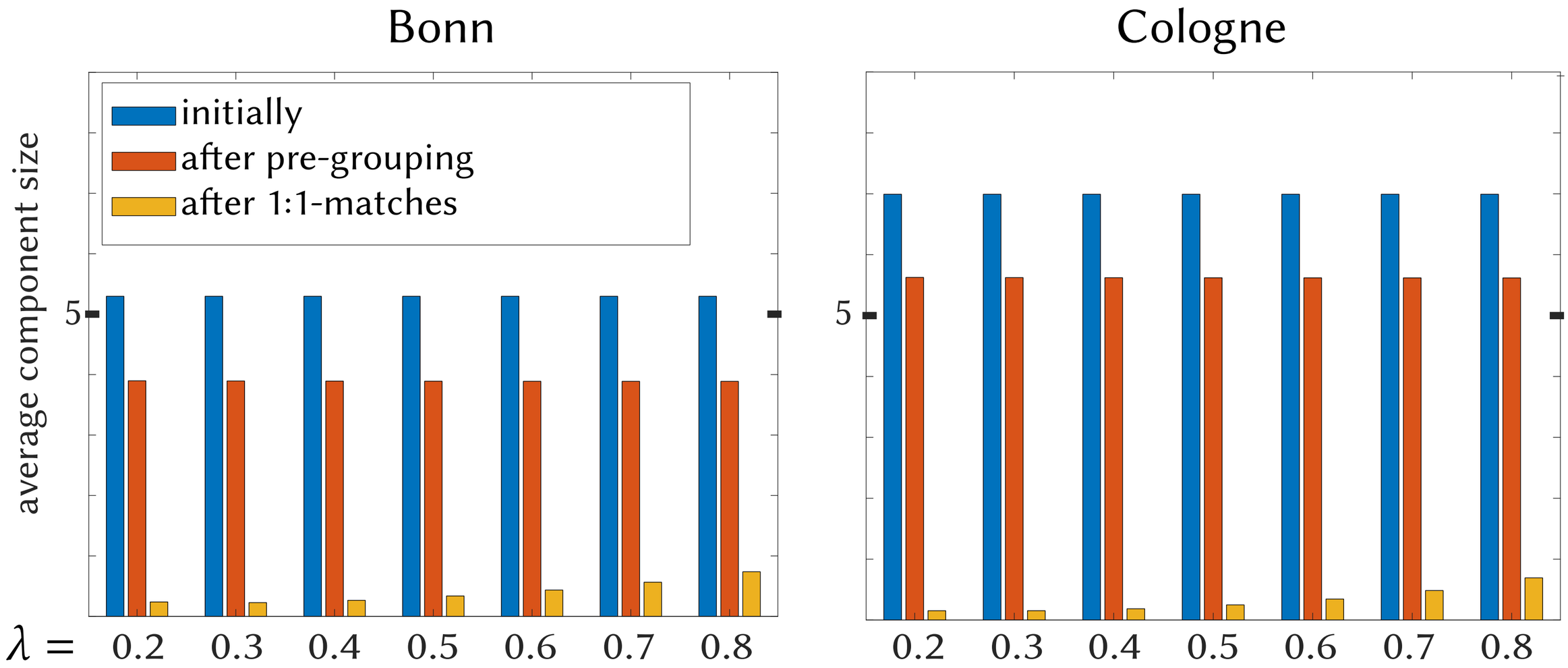
- implementation in C++
- application on cadastral data (ALKIS) vs. volunteered data (OSM)
- time limit of 1000s per connected component

Experimental Evaluation - Running Time

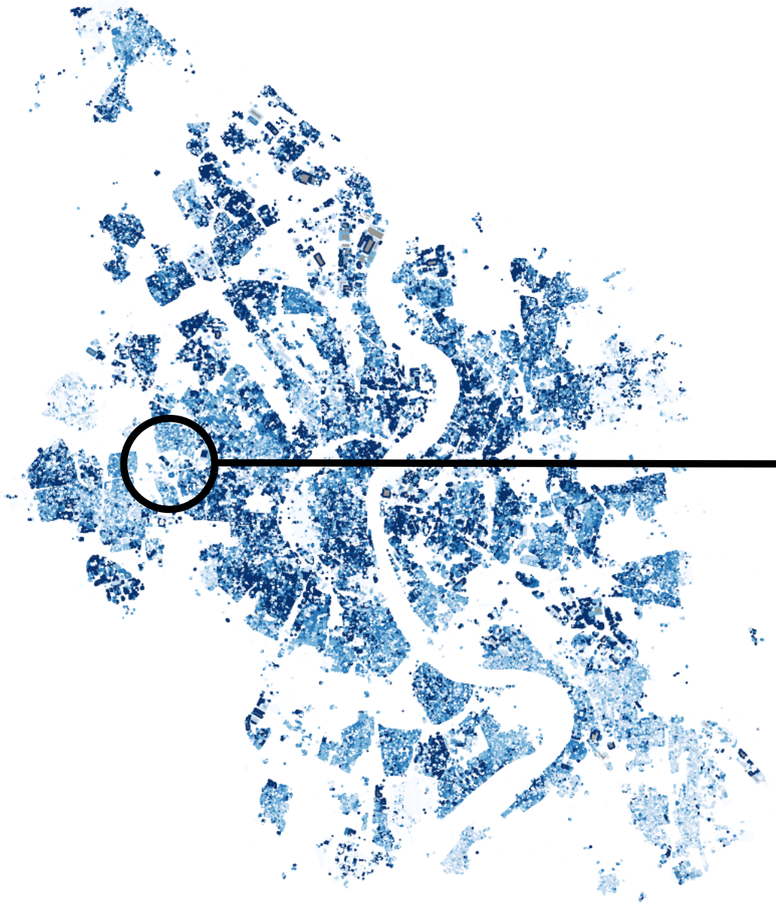
- implementation in C++
- application on cadastral data (ALKIS) vs. volunteered data (OSM)
- time limit of 1000s per connected component

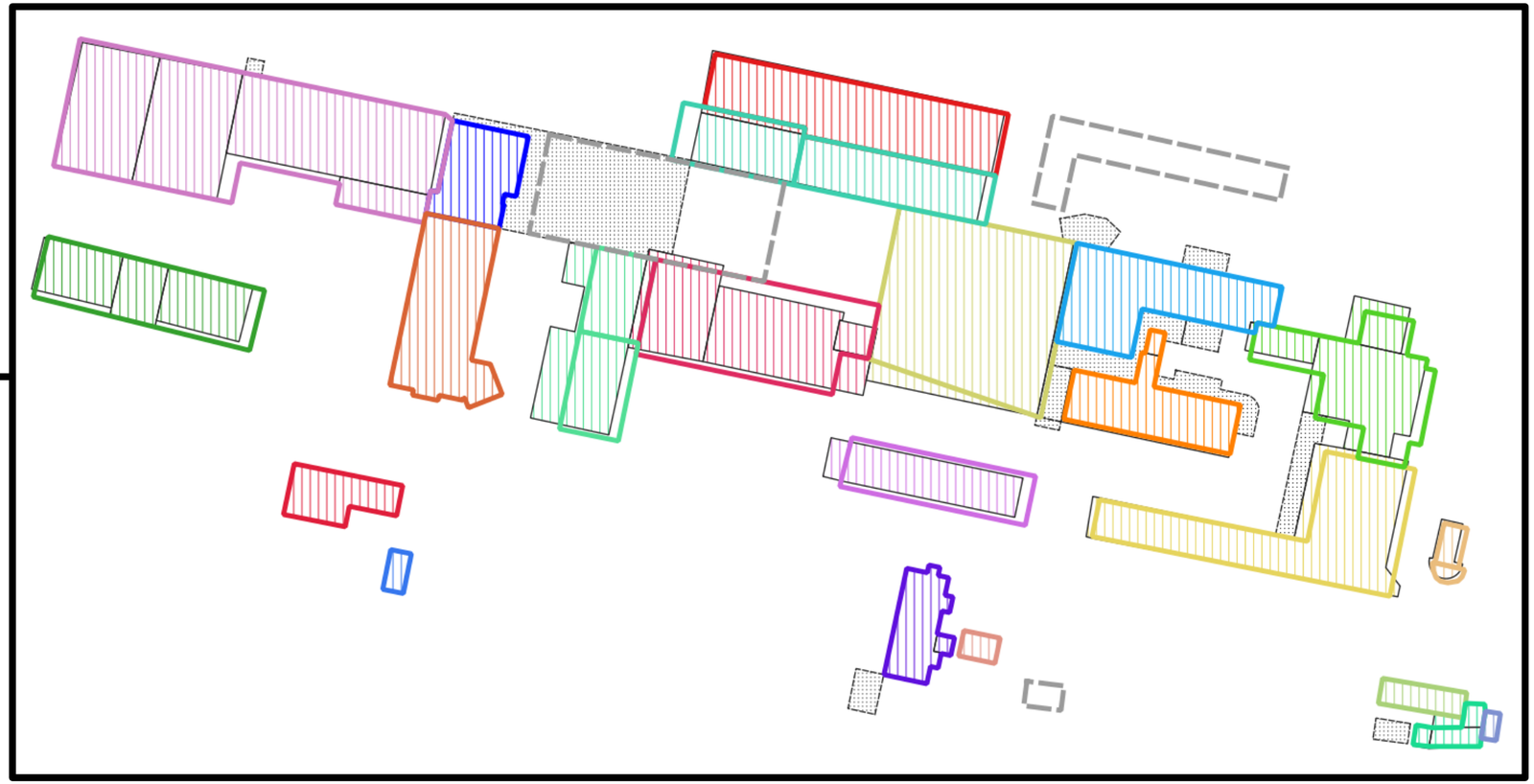


Experimental Evaluation - Precomputing



Experimental Evaluation - Visualization





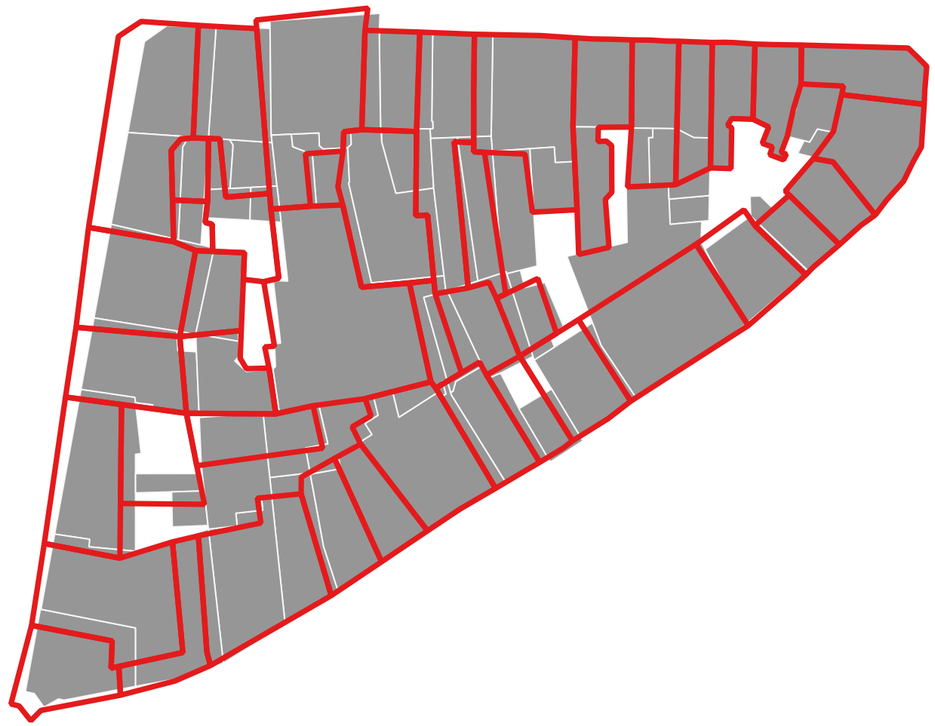
$\lambda = 0.4$

Experimental Evaluation - Limits

- There is a small number of components (up to 12/86596 for cologne) that could not be solved within this time limit, for example:

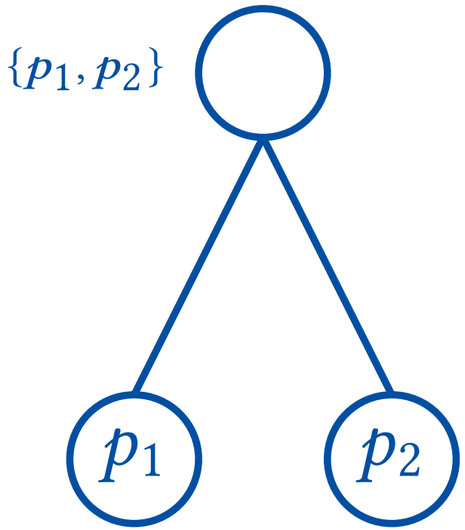
Experimental Evaluation - Limits

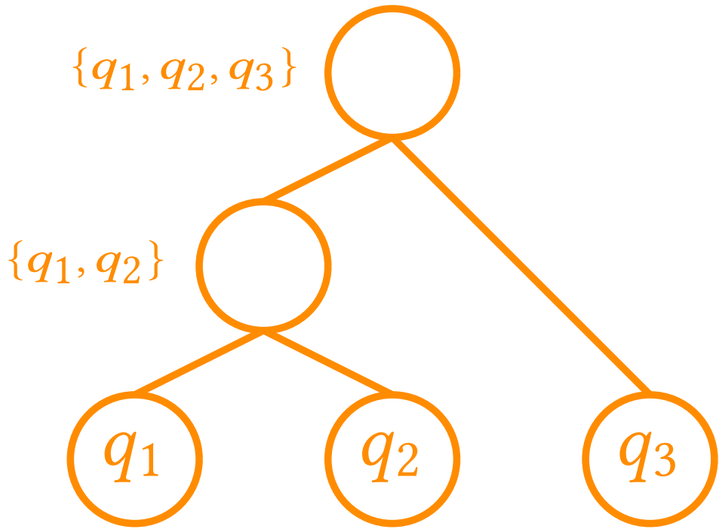
- There is a small number of components (up to 12/86596 for cologne) that could not be solved within this time limit, for example:



Open Questions & Outlook

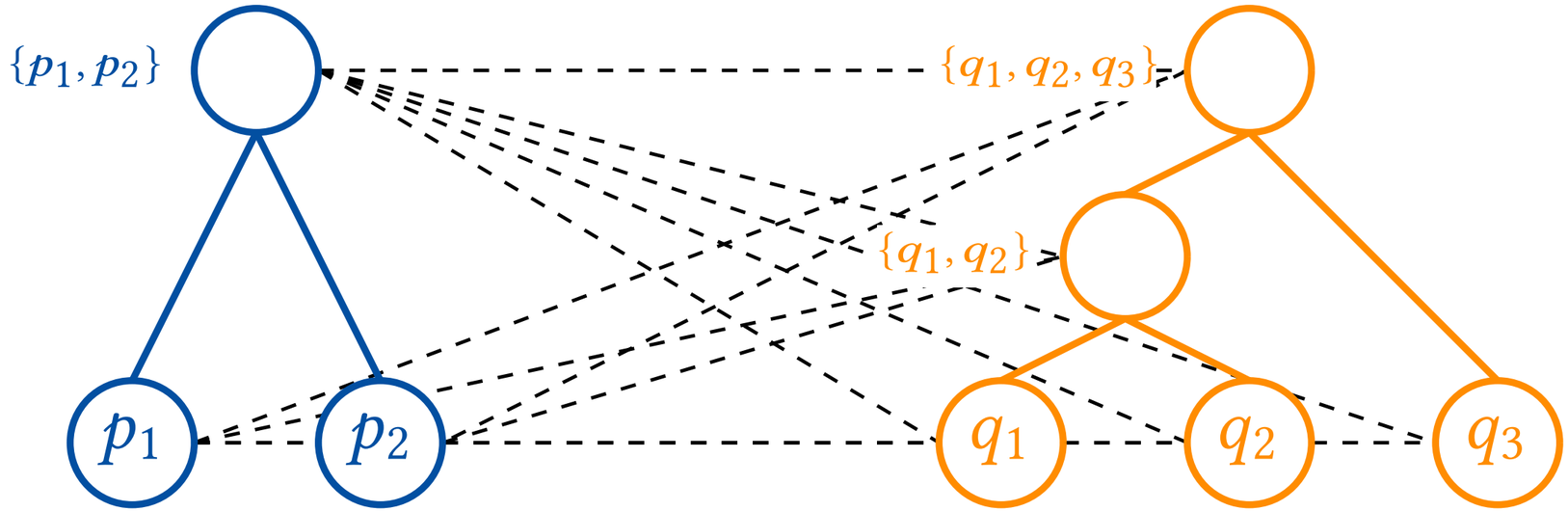
- can we approximate the optimal solution in polynomial time?
- can we meaningfully constraint our solution space?





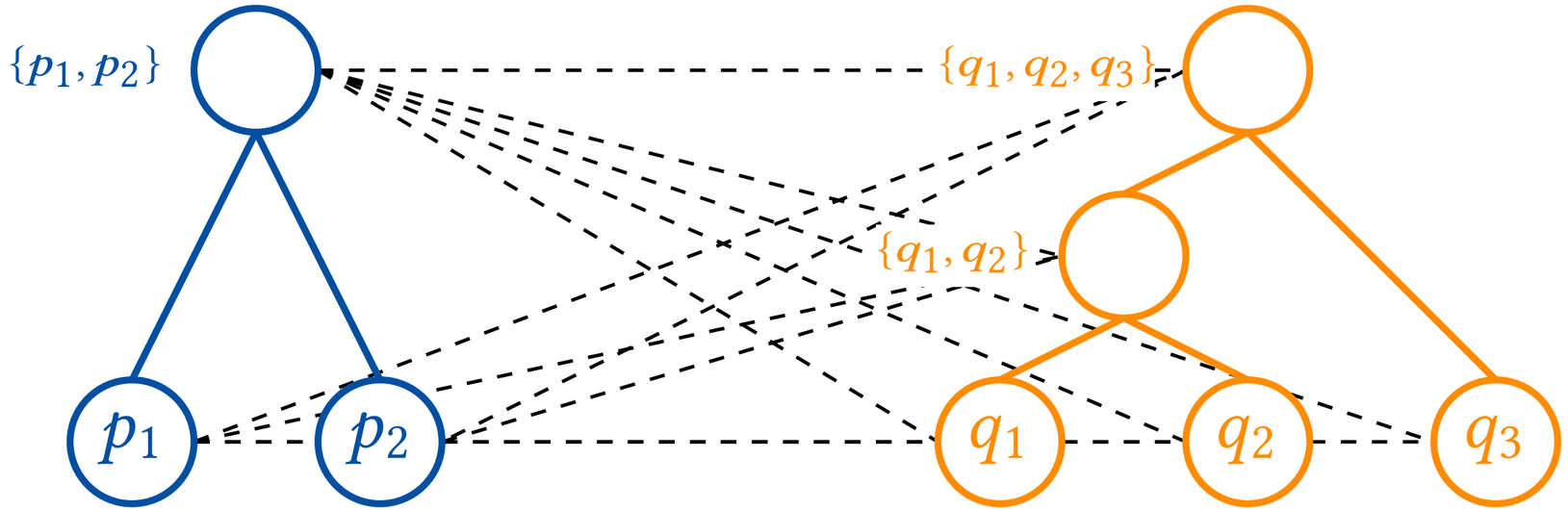
Open Questions & Outlook

- can we approximate the optimal solution in polynomial time?
- can we meaningfully constraint our solution space?



Open Questions & Outlook

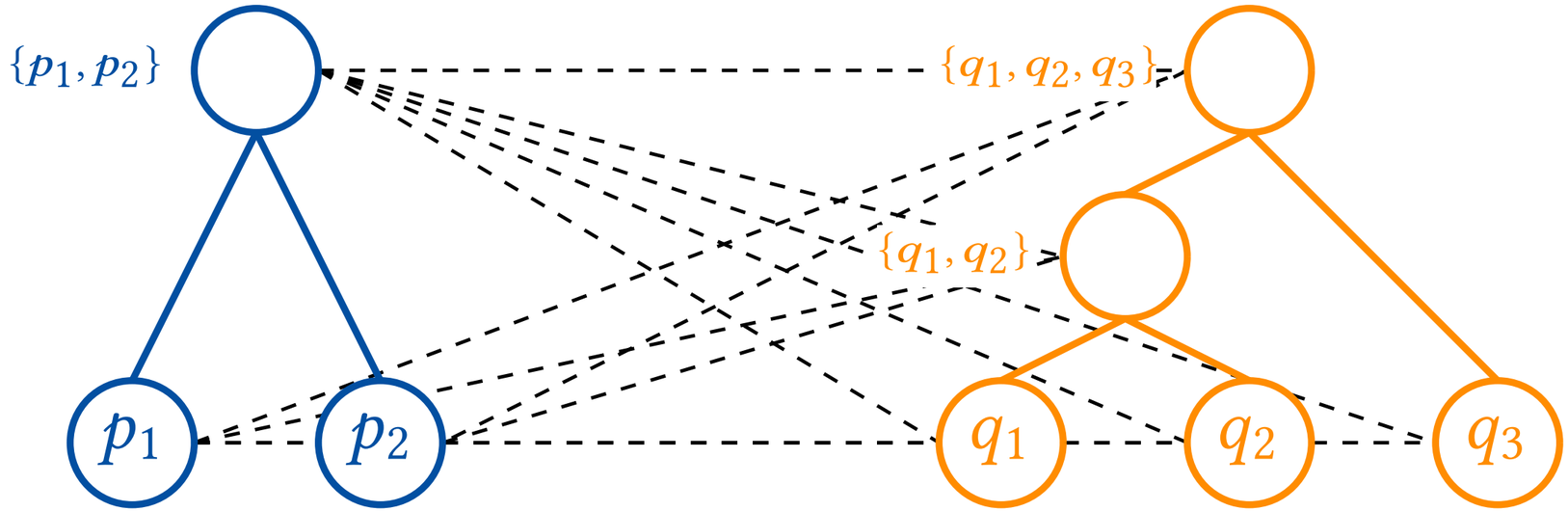
- can we approximate the optimal solution in polynomial time?
- can we meaningfully constraint our solution space?



- *tree-constrained bipartite matching problem.*
- polynomial-time 2-approximation algorithm exists

Open Questions & Outlook

- can we approximate the optimal solution in polynomial time?
- can we meaningfully constraint our solution space?



- *tree-constrained bipartite matching problem.*
- polynomial-time 2-approximation algorithm exists

Thank you!