Massive Experiments and Observational Studies: A Linearithmic Algorithm for Blocking/Matching/Clustering

Jasjeet S. Sekhon

UC Berkeley

June 21, 2016

Massive Experiments

- Measuring human activity has generated massive datasets with granular information that can be used for personalization of treatments, creating markets, modeling behavior
- Many inferential issues: e.g., unknown sampling frames, heterogeneity, targeting optimal treatments, many false positives
- Some traditional experimental design methods have become computationally infeasible—e.g., blocking
- Blocking: create strata and then randomize within strata
- Polynomial time solution not quick enough. Linearithmic is survivable.
- Algorithm can also be used for post-stratification, matching, and clustering

Causal inference in large samples

Big Data does not solve key inferential problems

- For experiments:
  - Need to adjust experiments to examine sub-groups and to increase power: With Great Power Comes Small Effect Sizes: 1e-9
  - Issues of randomization bias: poor external validity

- For observational studies:
  - Covariate balance is not a function of the sample size.
  - The fundamental problem of causal inference is not solved just by a large N

A New Blocking/Matching/Clustering Method

The method minimizes the pair-wise Maximum Within-Block Distance: λ

- Any valid distance metric (must satisfy the triangle inequality)
- Ensures good covariate balance by design
- Works for any number of treatments and any minimum number of observations per block
- It is fast: $O(n \log n)$ expected time
- It is memory efficient: $O(n)$ storage
- Approximately optimal: $\leq 4 \times \lambda$
- Special cases
  1. with one covariate: λ
  2. with two covariates: $\leq 2 \times \lambda$

Some Current blocking approaches

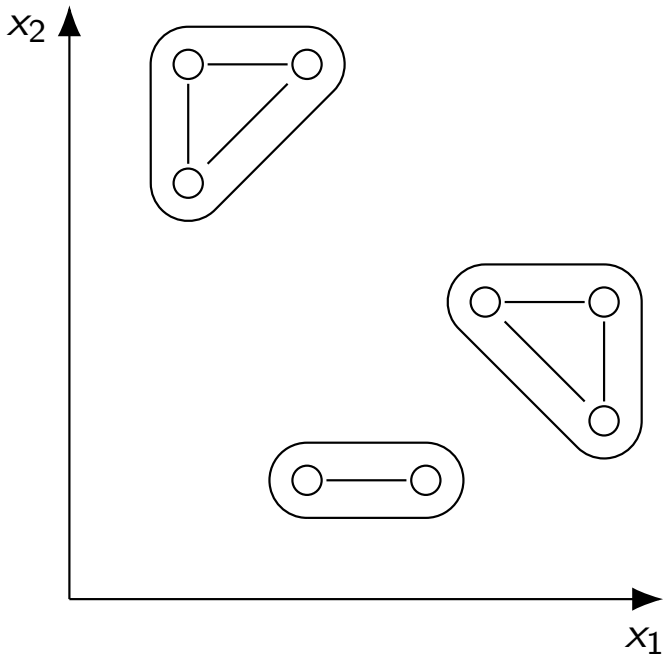
Blocking, Matching, Clustering is a NP hard problem in general

- Optimal Multivariate Matching Before Randomization
  - No efficient way to extend approach to more than two treatment categories
  - Even for two treatment categories, doesn't scale well

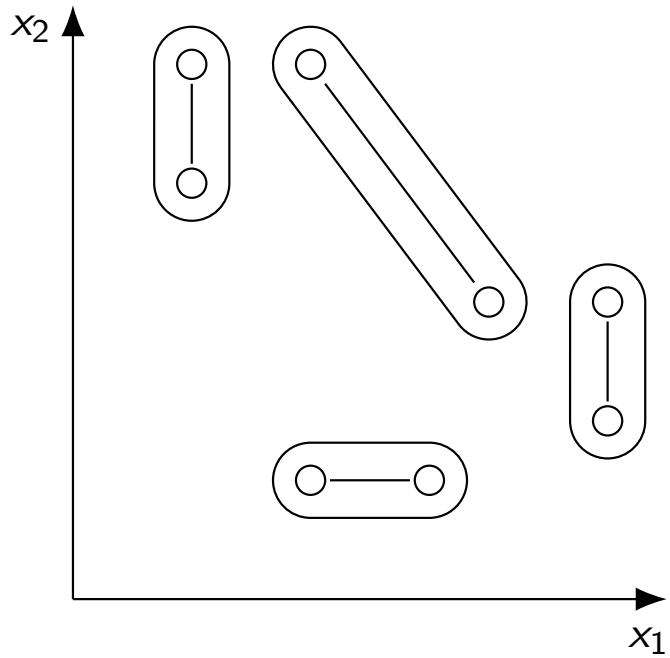
- Matched-pairs blocking: Pair "most-similar" units together. For each pair, randomly assign one unit to treatment, one to control
  - Natural clustering in the data ignored
  - Cannot estimate conditional variances
  - Difficulty with treatment effect heterogeneity

Threshold blocking: relaxing the block structure

Threshold blocking



Fixed-sized blocking



An Advantage

Theorem

For all samples, all objective functions and all desired block sizes, the optimal threshold blocking is always weakly better than the optimal fixed-sized blocking.

- Proof: interpret blocking as an non-linear integer programming problem.
  - The search set of threshold blocking is a superset of fixed-sized blocking

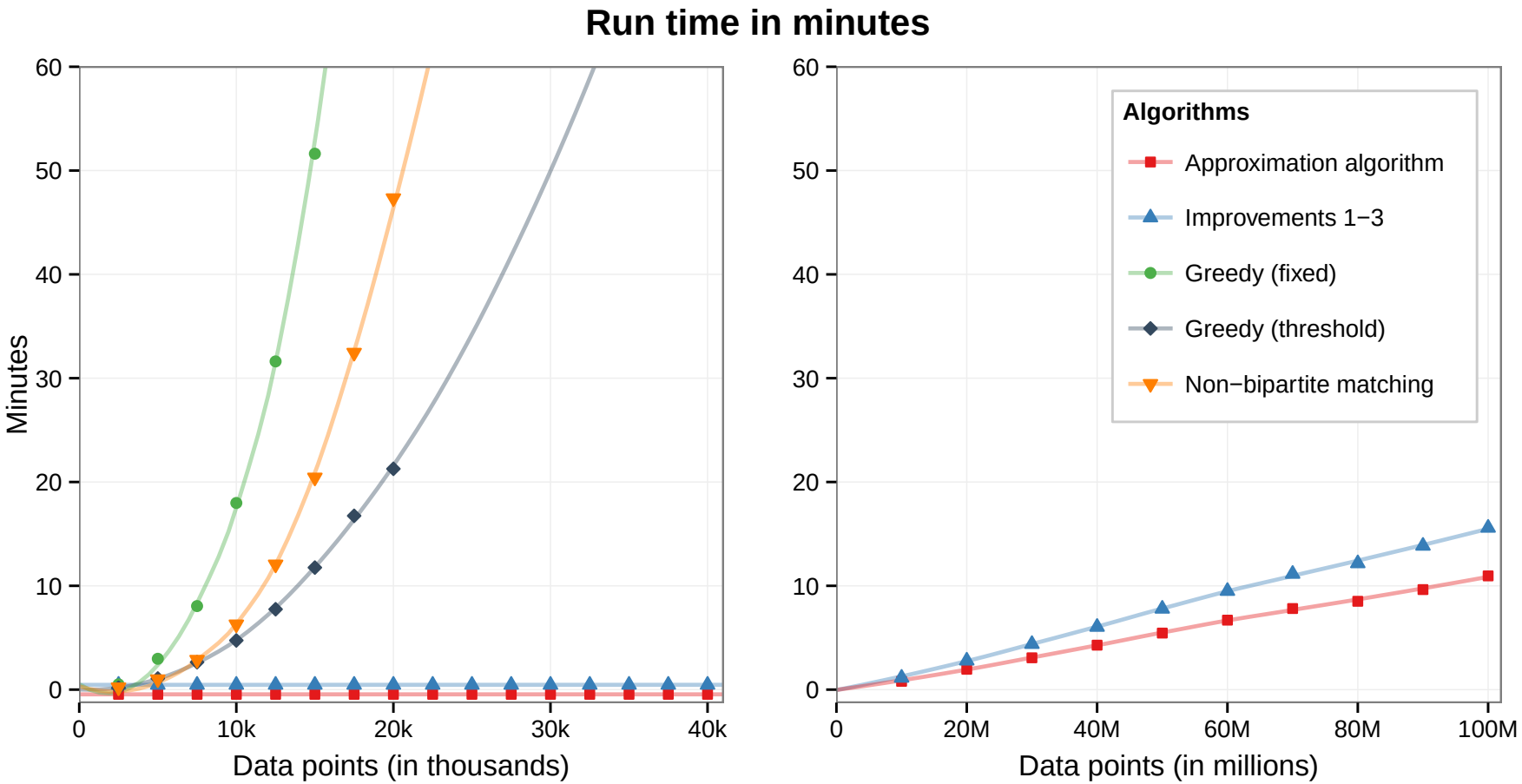
But there are problems

- Problem 1: the theorem is for the objective function used to construct the blocks
  - Might not be the quantity of true interest
- Problem 2: larger search set $\Rightarrow$ much harder optimization problem
  - No help to us if we cannot find the optimum

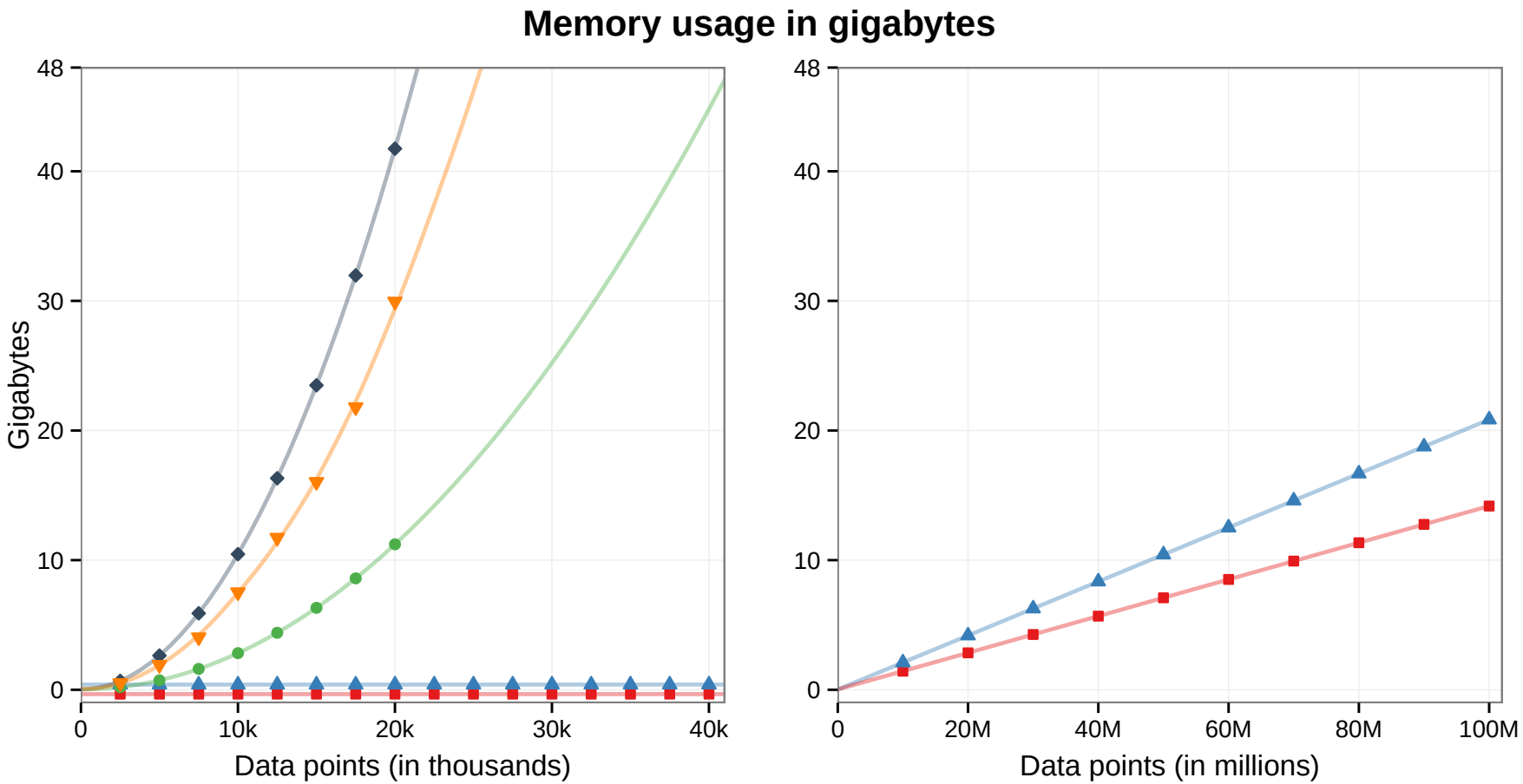
Table: # unique blockings (block size = 2)

# units	Fixed-sized	Threshold
8	105	715
10	945	17,722
12	10,395	580,317
14	135,135	24,011,157
16	2,027,025	1,216,070,380
18	34,459,425	73,600,798,037
20	654,729,075	5.2×10^{12}

The AppOpt algorithm



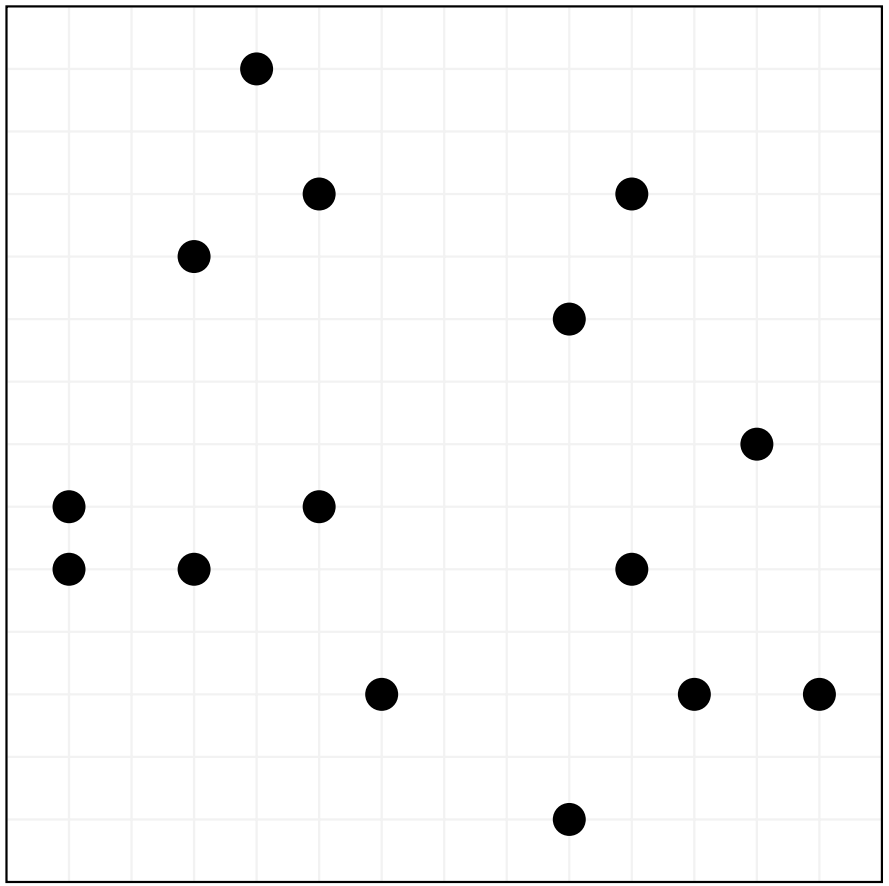
The AppOpt algorithm



The AppOpt algorithm

Input:

- Units' covariates
- Distance metric
- Minimum block size: $k = 2$



Procedure:

1. A undirected complete graph with distances as edge weights
2. Find $(k-1)$-nearest neighbor graph
3. Construct the second power of NNG
4. Find a maximal independent set (seeds)
5. Form blocks with the seeds and their neighbors in NNG
6. Assign remaining units to a block containing any neighbor

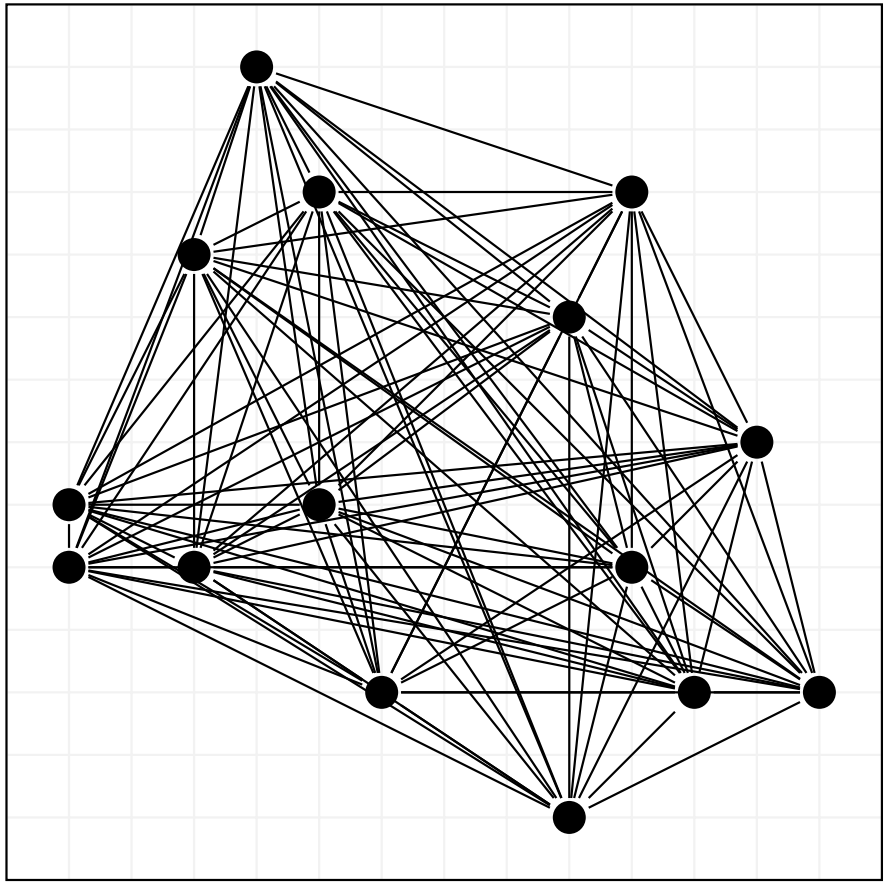
The AppOpt algorithm

Input:

- Units' covariates
- Distance metric
- Minimum block size: $k = 2$

Procedure:

1. A undirected complete graph with distances as edge weights
2. Find $(k-1)$-nearest neighbor graph
3. Construct the second power of NNG
4. Find a maximal independent set (seeds)
5. Form blocks with the seeds and their neighbors in NNG
6. Assign remaining units to a block containing any neighbor



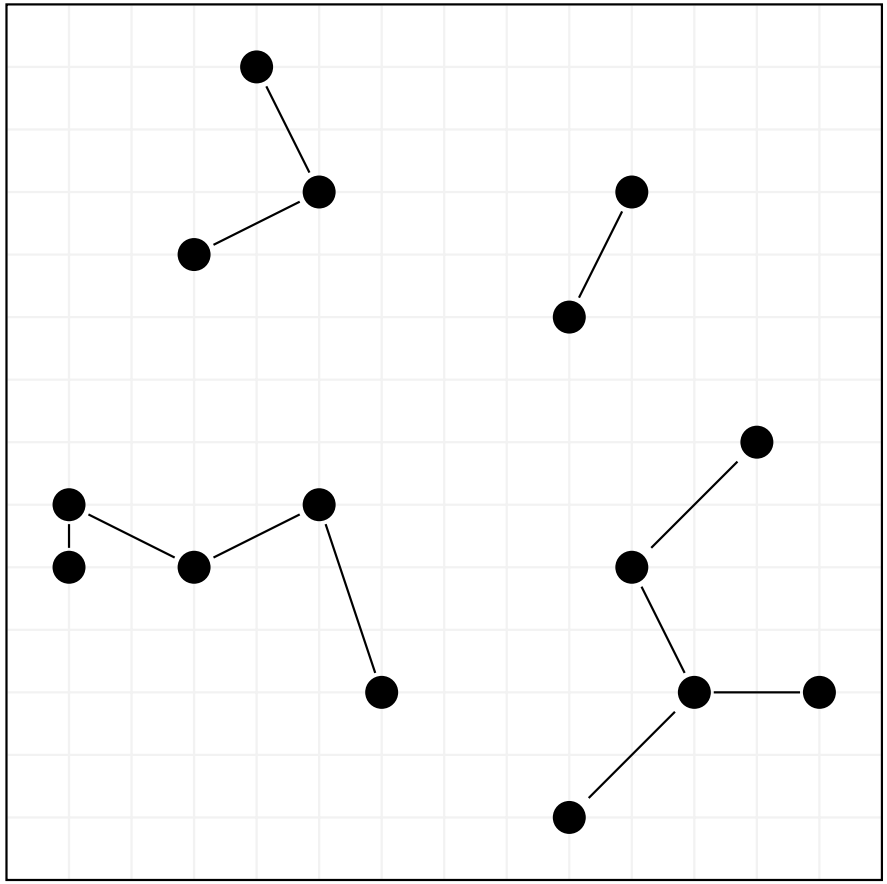
The AppOpt algorithm

Input:

- Units' covariates
- Distance metric
- Minimum block size: $k = 2$

Procedure:

1. A undirected complete graph with distances as edge weights
2. Find $(k-1)$-nearest neighbor graph
3. Construct the second power of NNG
4. Find a maximal independent set (seeds)
5. Form blocks with the seeds and their neighbors in NNG
6. Assign remaining units to a block containing any neighbor



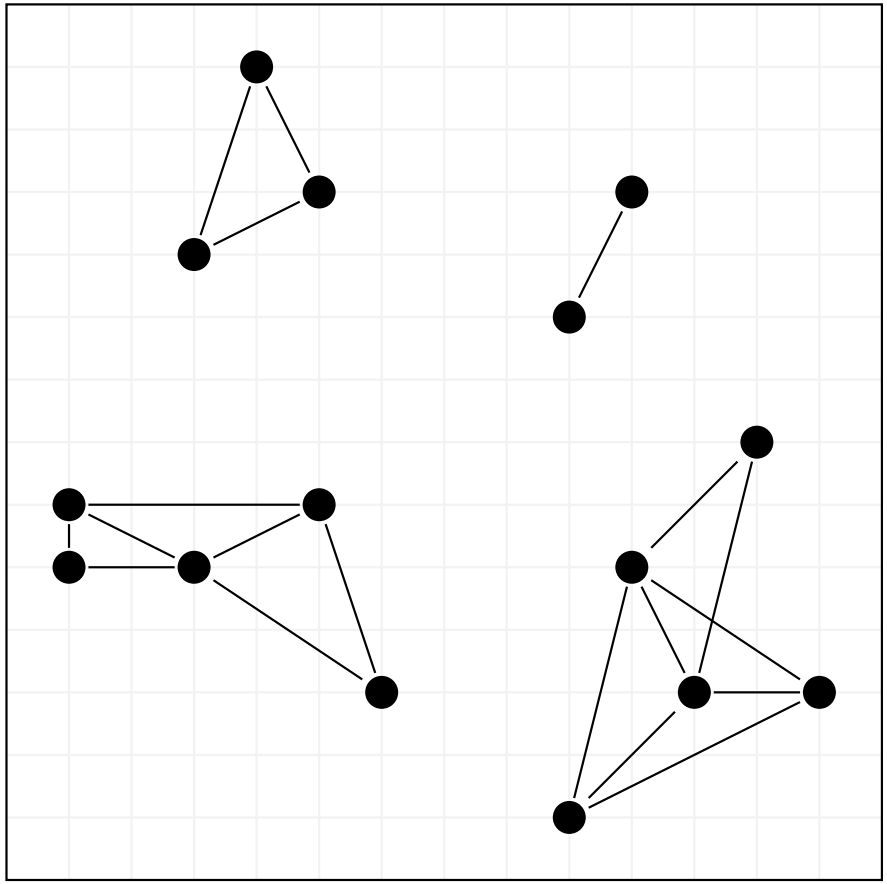
The AppOpt algorithm

Input:

- Units' covariates
- Distance metric
- Minimum block size: $k = 2$

Procedure:

1. A undirected complete graph with distances as edge weights
2. Find $(k-1)$-nearest neighbor graph
3. Construct the second power of NNG
4. Find a maximal independent set (seeds)
5. Form blocks with the seeds and their neighbors in NNG
6. Assign remaining units to a block containing any neighbor



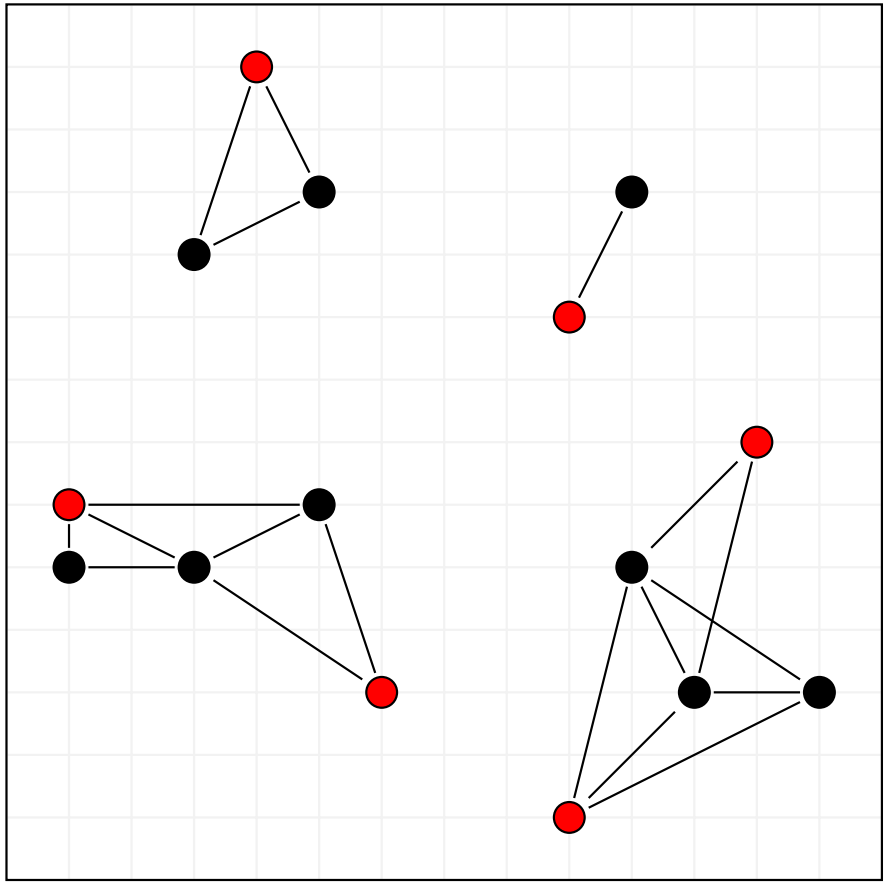
The AppOpt algorithm

Input:

- Units' covariates
- Distance metric
- Minimum block size: $k = 2$

Procedure:

1. A undirected complete graph with distances as edge weights
2. Find $(k-1)$-nearest neighbor graph
3. Construct the second power of NNG
4. Find a maximal independent set (seeds)
5. Form blocks with the seeds and their neighbors in NNG
6. Assign remaining units to a block containing any neighbor



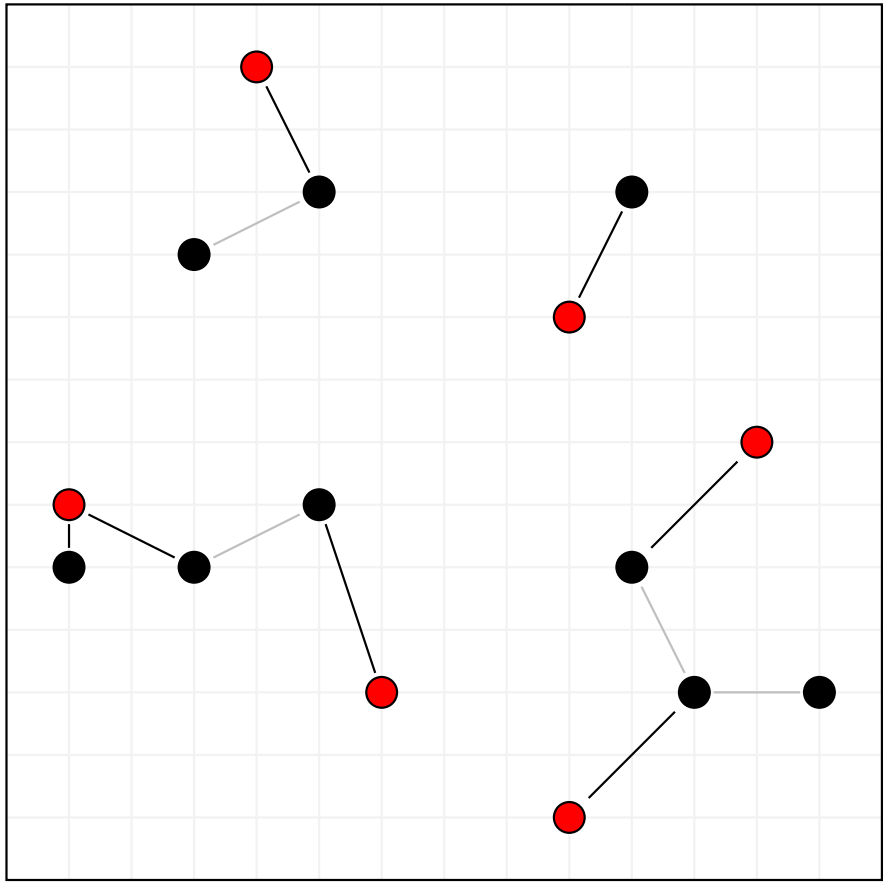
The AppOpt algorithm

Input:

- Units' covariates
- Distance metric
- Minimum block size: $k = 2$

Procedure:

1. A undirected complete graph with distances as edge weights
2. Find $(k-1)$-nearest neighbor graph
3. Construct the second power of NNG
4. Find a maximal independent set (seeds)
5. Form blocks with the seeds and their neighbors in NNG
6. Assign remaining units to a block containing any neighbor



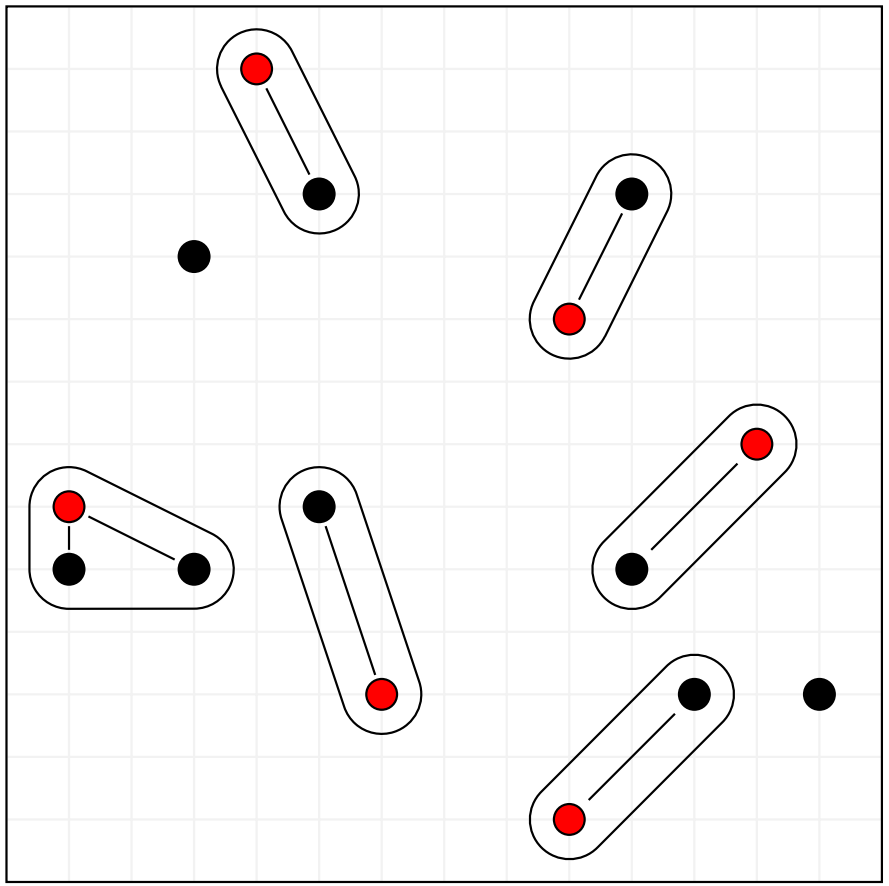
The AppOpt algorithm

Input:

- Units' covariates
- Distance metric
- Minimum block size: $k = 2$

Procedure:

1. A undirected complete graph with distances as edge weights
2. Find $(k-1)$-nearest neighbor graph
3. Construct the second power of NNG
4. Find a maximal independent set (seeds)
5. Form blocks with the seeds and their neighbors in NNG
6. Assign remaining units to a block containing any neighbor



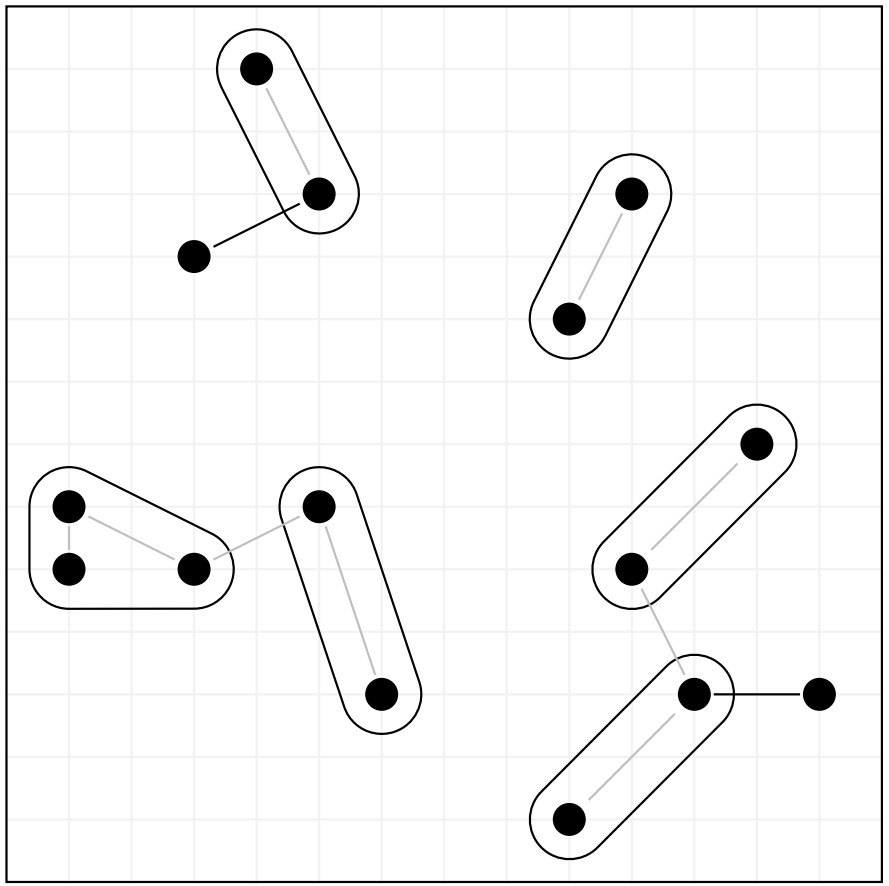
The AppOpt algorithm

Input:

- Units' covariates
- Distance metric
- Minimum block size: $k = 2$

Procedure:

1. A undirected complete graph with distances as edge weights
2. Find $(k-1)$-nearest neighbor graph
3. Construct the second power of NNG
4. Find a maximal independent set (seeds)
5. Form blocks with the seeds and their neighbors in NNG
6. Assign remaining units to a block containing any neighbor



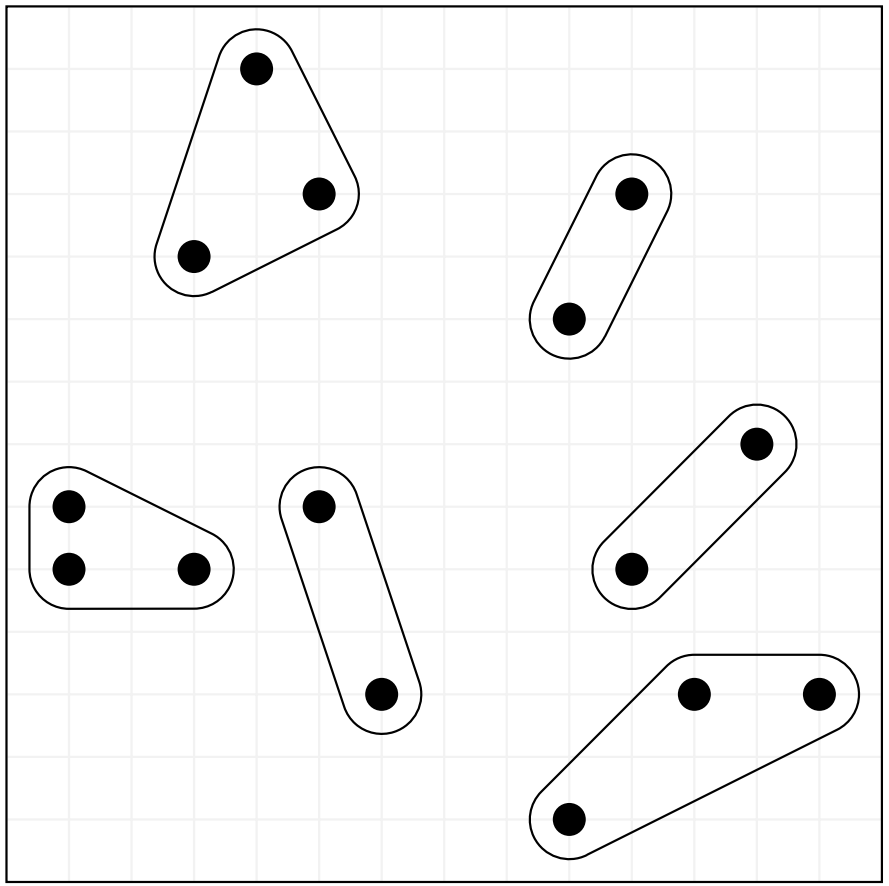
The AppOpt algorithm

Input:

- Units' covariates
- Distance metric
- Minimum block size: $k = 2$

Procedure:

1. A undirected complete graph with distances as edge weights
2. Find $(k-1)$-nearest neighbor graph
3. Construct the second power of NNG
4. Find a maximal independent set (seeds)
5. Form blocks with the seeds and their neighbors in NNG
6. Assign remaining units to a block containing any neighbor



Properties

- Unless $P = NP$, no polynomial-time $(2 - \epsilon)$ approximation algorithm exists
- Validity: the blocking algorithm produces a threshold blocking: $\mathbf{b}_{alg} \in \mathbf{B}_k$
- Approximate optimality: blocking algorithm is a 4-approximation algorithm:

$$\max_{ij \in E(\mathbf{b}_{alg})} c_{ij} \leq 4\lambda.$$

- Local approximate optimality: Let $\mathbf{b}_{sub} \subseteq \mathbf{b}_{alg}$ be any subset of blocks from a blocking constructed by the algorithm. Define $V_{sub} = \bigcup_{V_x \in \mathbf{b}_{sub}} V_x$ as the set of all vertices contained in the blocks of $\mathbf{b}_{sub}$. Let λ_{sub} denote the maximum edge cost in an optimal blocking of V_{sub}. The subset of blocks is an approximately optimal blocking of V_{sub}:

$$\max_{ij \in E(\mathbf{b}_{sub})} c_{ij} \leq 4\lambda_{sub}.$$

Summary

- Closer to clustering than traditional blocking/matching methods
- Key property is obtained indirectly: we are not directly optimizing the objective function
- Fast algorithm:
  - NNG plus $O(d^0 kn)$ time and $O(d^0 kn)$ space
  - K-d trees NN: $O(2^d kn \log n)$ expected time, $O(2^d kn^2)$ worst time, and $O(kn)$ storage
  - Compare with bipartite, network flow methods:
    - e.g., Derigs: $O(n^3 \log n + dn^2)$ worst time and $O(d^0 n^2)$ space
- A key preprocessing step: feature selection

Observational Data

- The usual assumption: selection on observables
- Covariate balance is not a function of the sample size

$\Rightarrow$ We need matching methods also in large samples

What is matching?

- Matching is a non-parametric method that creates balanced samples:
  - Construct matched groups (MG) of similar units
  - Re-weight units so that each treatment condition is equally "big" in each MG

  $\Rightarrow$ As the MGs are approximately balanced, so will the re-weighted sample

Inspiration from two extremes

- **Fast method:** Greedy NN-matching without replacement.
  - **Sequentially** matches each treated unit to its nearest unmatched control to form **pairs**.

- **Well-performing method:** Optimal full matching.
  - Finds the **best** matching subject to **only** the design constraints.

What is generalized full matching (GFM)?

- **GFM extends full matching to a more general setting.**

- In a study with **two** treatment conditions, a **full matching** satisfies:
  1. Each unit must be assigned to a matched group.
  2. Each group must contain at least **one** treated and **one** control.

What is generalized full matching (GFM)?

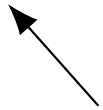
- **GFM extends full matching to a more general setting.**

- In a study with **two** treatment conditions, a **full matching** satisfies:
  1. Each unit must be assigned to a matched group.
  2. Each group must contain at least **one** treated and **one** control.

- In a study with k treatment conditions, a **GFM** satisfies:
  1. Each unit must be assigned to a matched group.
  2. Each group must contain at least τ_i units for each treatment $i \in \{1, \cdots, k\}$.
  3. Each group must contain at least τ_A units in total.

Matching as optimization

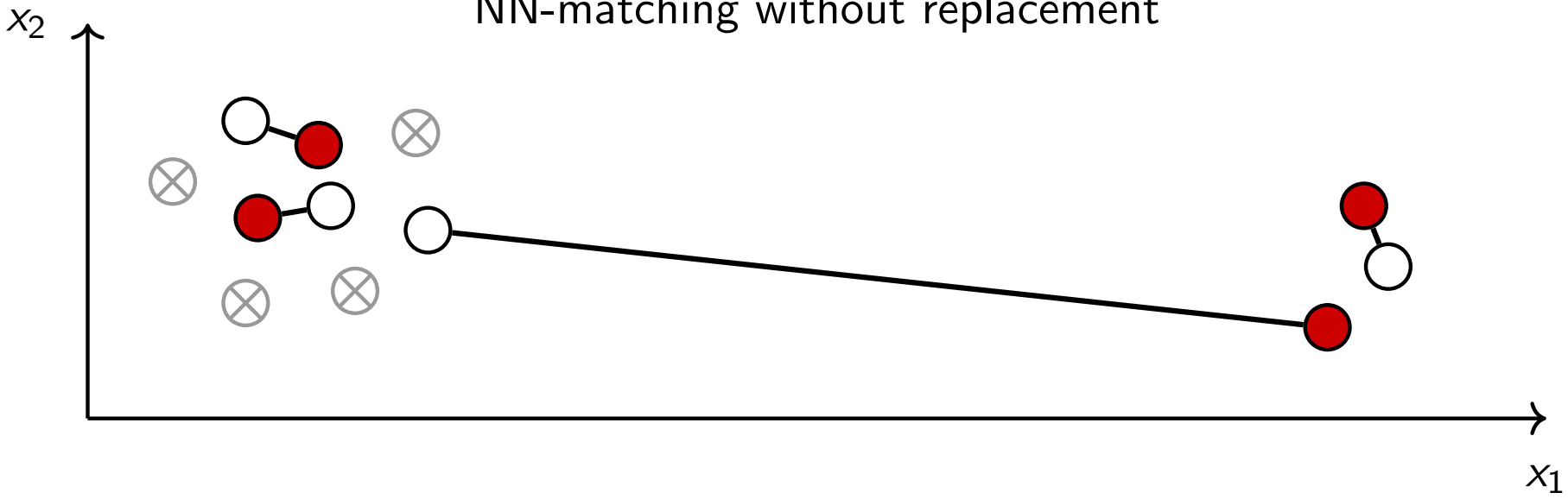
- **Three properties of a good matching method:**
  - Constructs matched groups with units that are similar to each other.
  - Groups conform to a desired structure (e.g., one unit of each treatment).
  - There is a way to construct the groups.



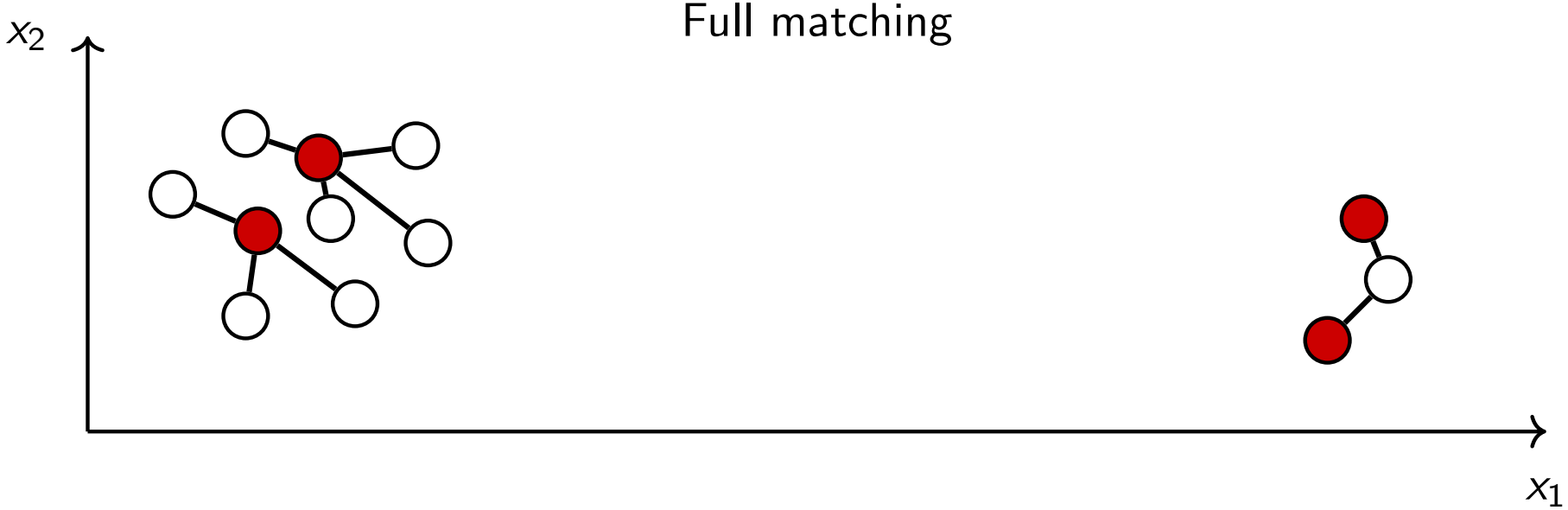

x_2
x_1


x_2
x_1

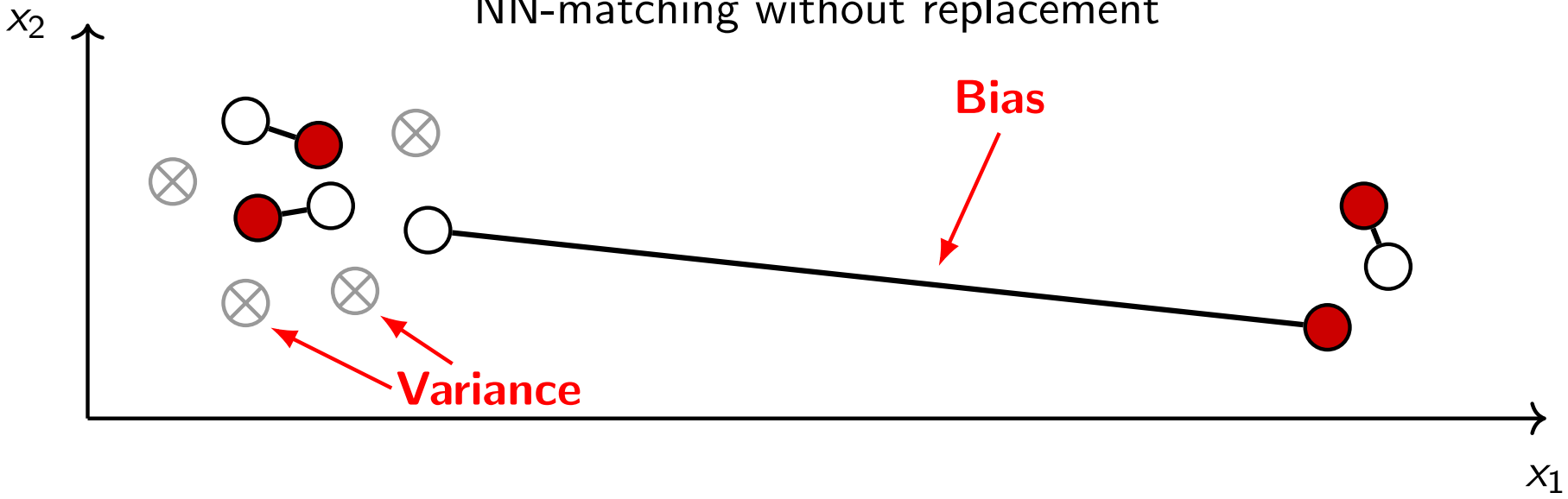
NN-matching without replacement



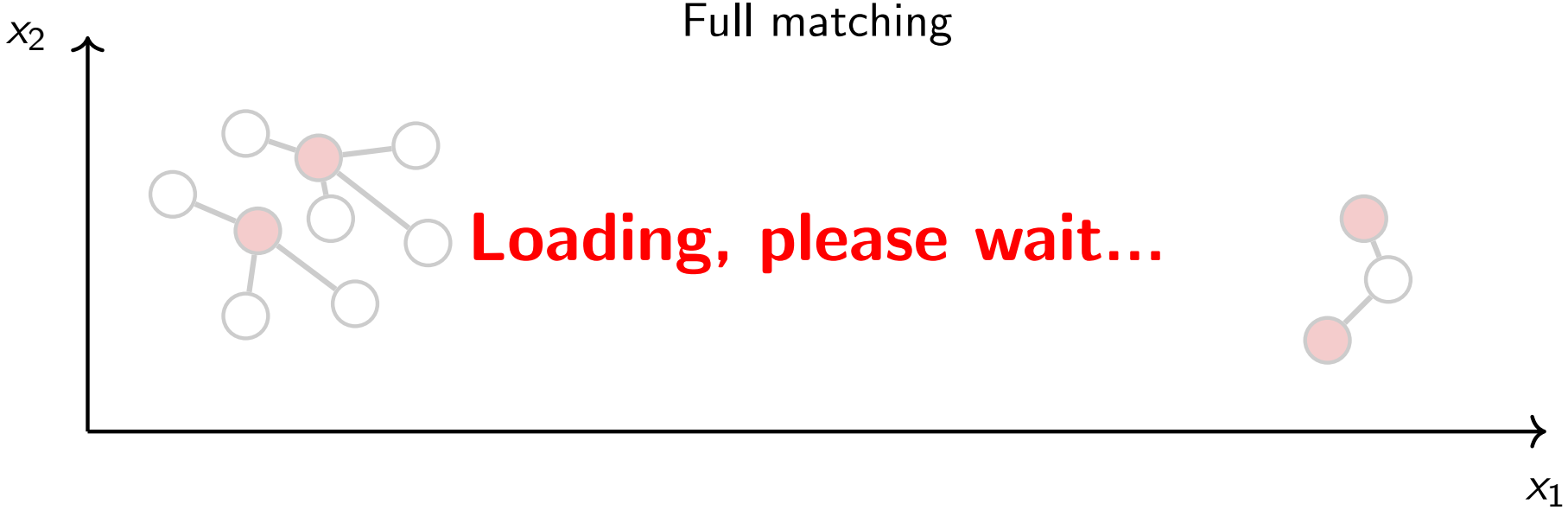
Full matching



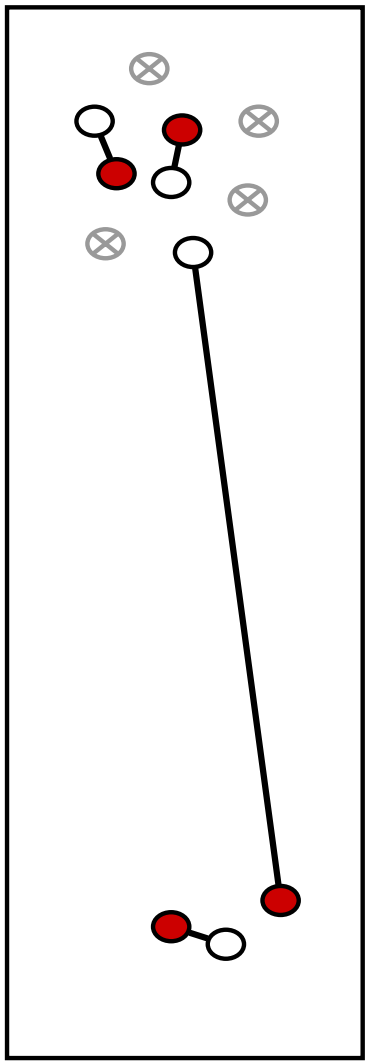
NN-matching without replacement



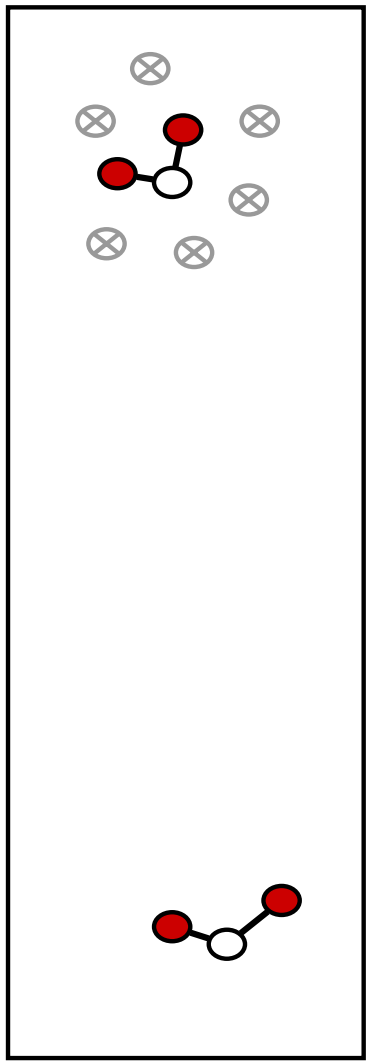
Full matching



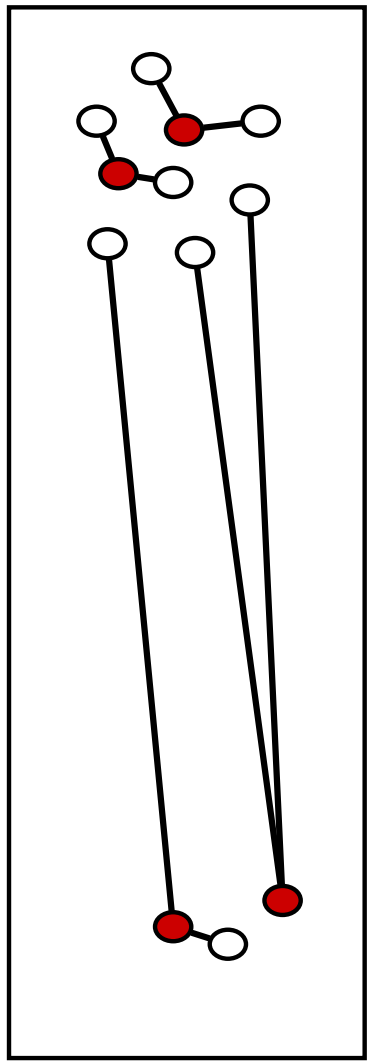
Without replacement



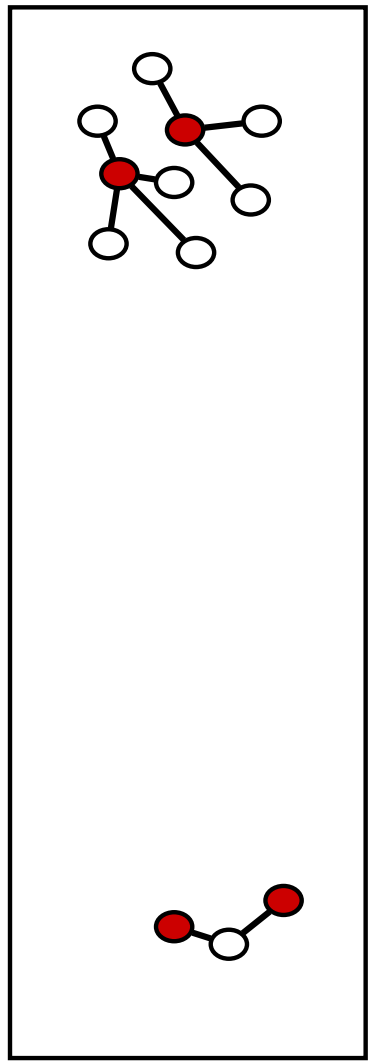
With replacement



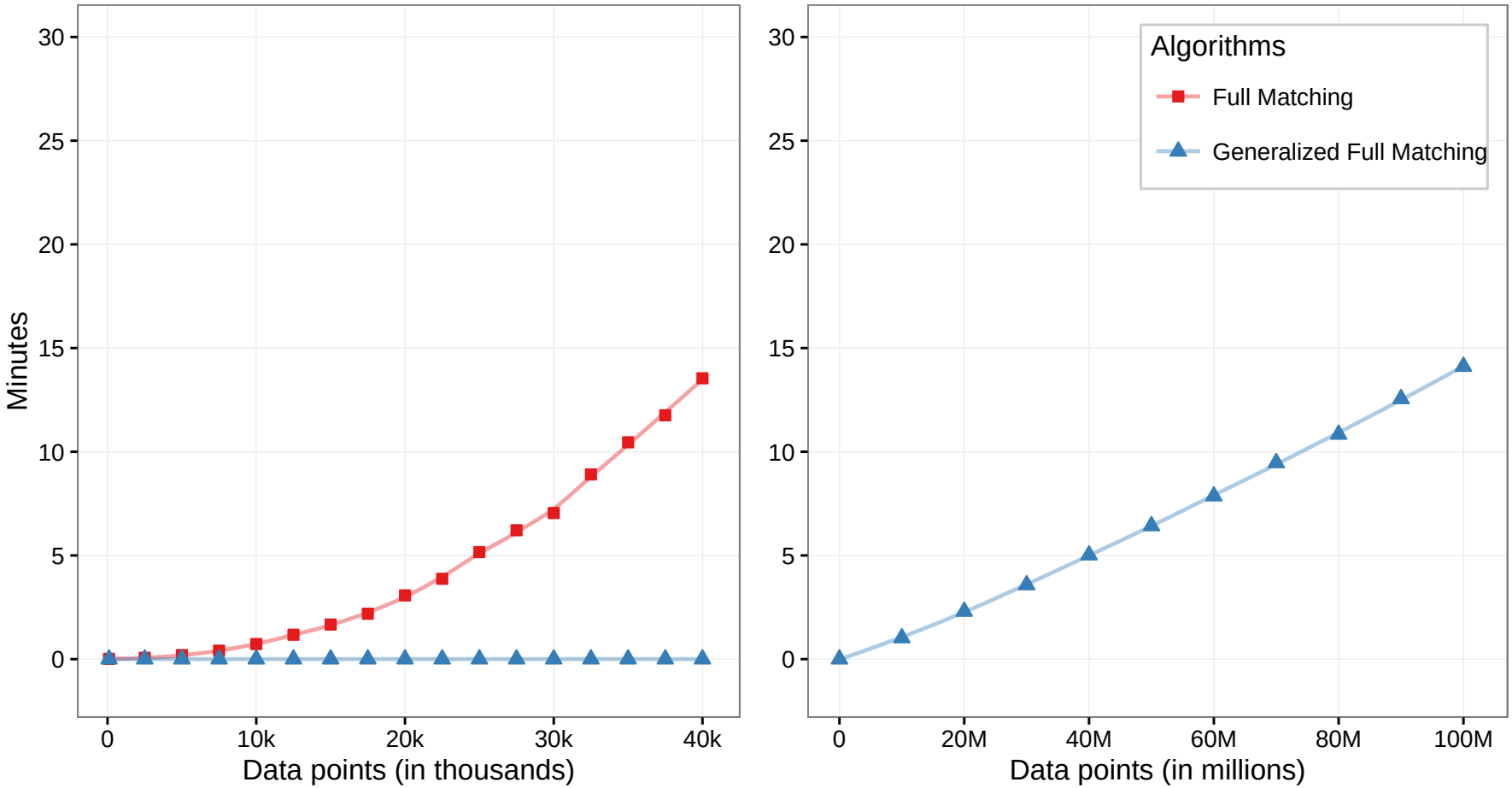
1:k-matching



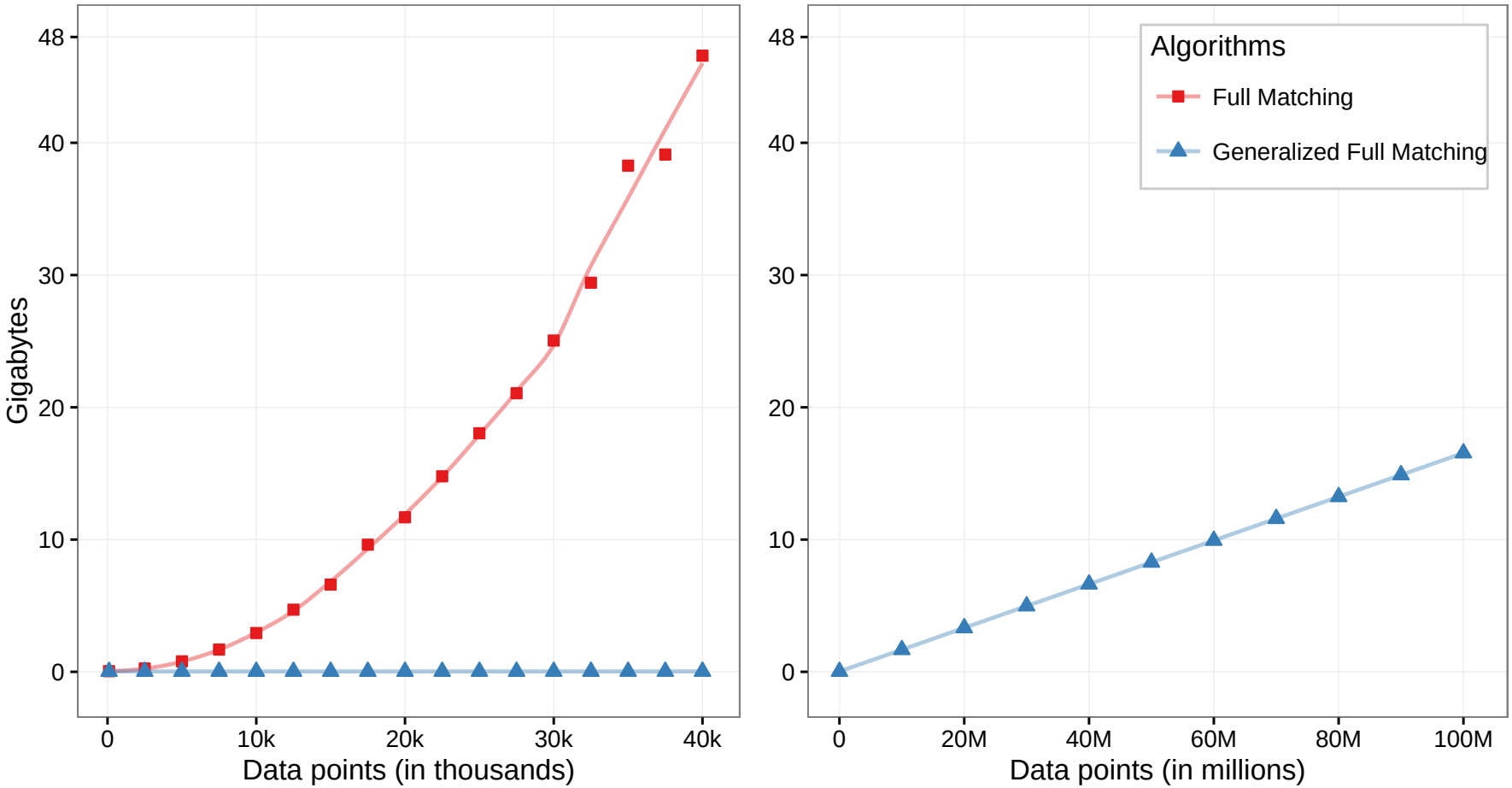
Full matching



Matching algorithm



Matching algorithm



Clustering

- Follows in a similar way to the blocking/matching case
- One selects the minimum number of observations in a cluster; not the number of clusters
- Allows one to use clustering as a data reduction step for further analysis—e.g., by ML estimation methods

Summary

- Existing methods don't work well in large samples
- Applied used created hacked heuristic algorithms, which have no proven properties and often have worse computational performance
- Have ignored estimation issues in this talk: in practice they are important

Joint Work with Michael J. Higgins and Fredrick Sävje