Generalized full matching

Fredrik Sävje[1] Michael Higgins[2] Jasjeet Sekhon[1]

[1]Departments of Political Science and Statistics, UC Berkeley

[2]Department of Statistics, Kansas State University

July 31, 2016
JSM

Causal inference in large samples

- Massive data sets are great. We can, for example, investigate:
  - Very small, but relevant, treatment effects
  - Conditional effects at a fine-grained level

- **However, "big data" does not solve the fundamental problem of causal inference**

- Matching:
  - Covariate balance is not a function of the sample size

  $\Rightarrow$ We need matching methods in large samples

  $\Rightarrow$ Also needed for complicated designs: multiple treatment groups; heterogenous treatment effects by matched group

  $\Rightarrow$ Can be used for post-stratification because many pipelines do not allow for blocking

What is the problem?

- **Matching problems are NP-Hard in the general case**
- Simplifications are often needed even in smallish samples:
  - Limit the scope to certain designs.
  - Impose simplifying constraints.
  - Use heuristic algorithms.
- Fast methods $\neq$ good methods.
- Can we break this trade-off?

▸ What is matching? ▸ Matching as optimization

Inspiration from two extremes

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

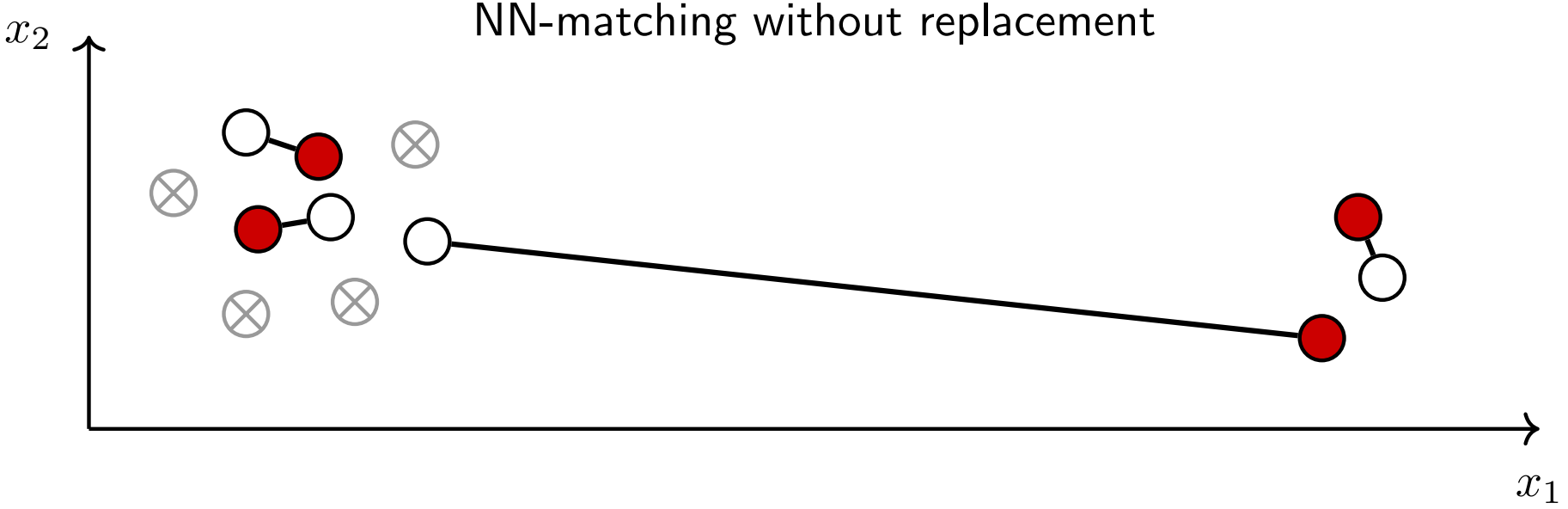
- **Well-performing method:** Optimal full matching.
  - Finds the **best** matching subject to **only** the design constraints.

▸ Illustration

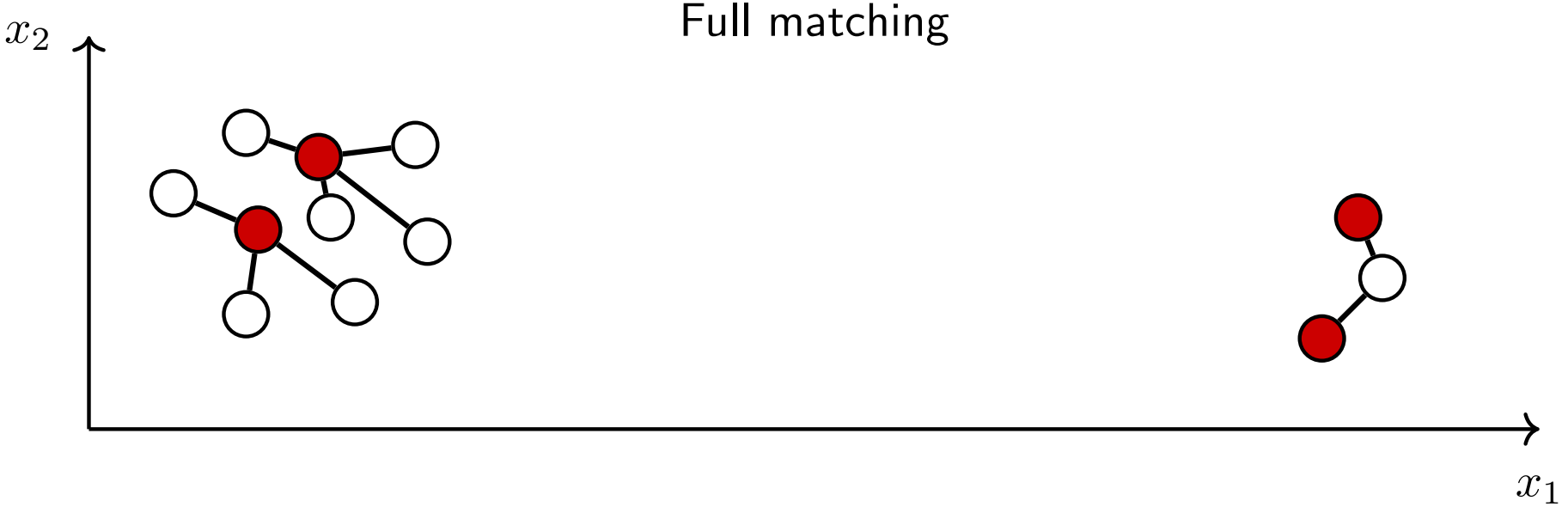

x_2
x_1


x_2
x_1

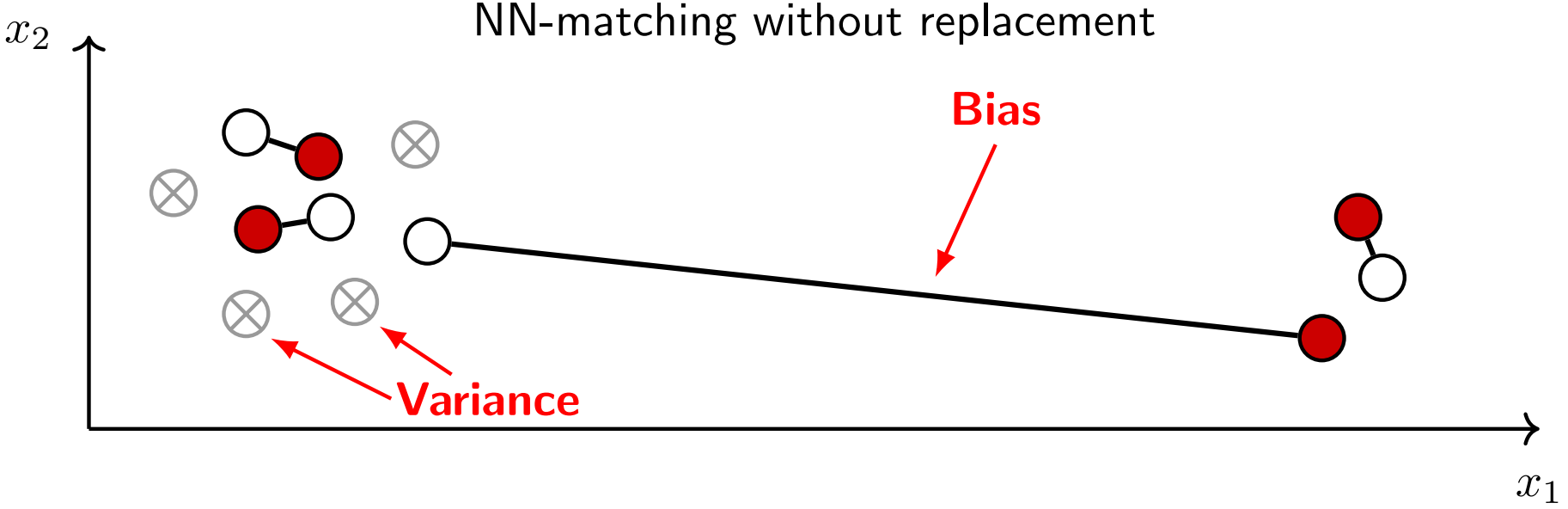
NN-matching without replacement



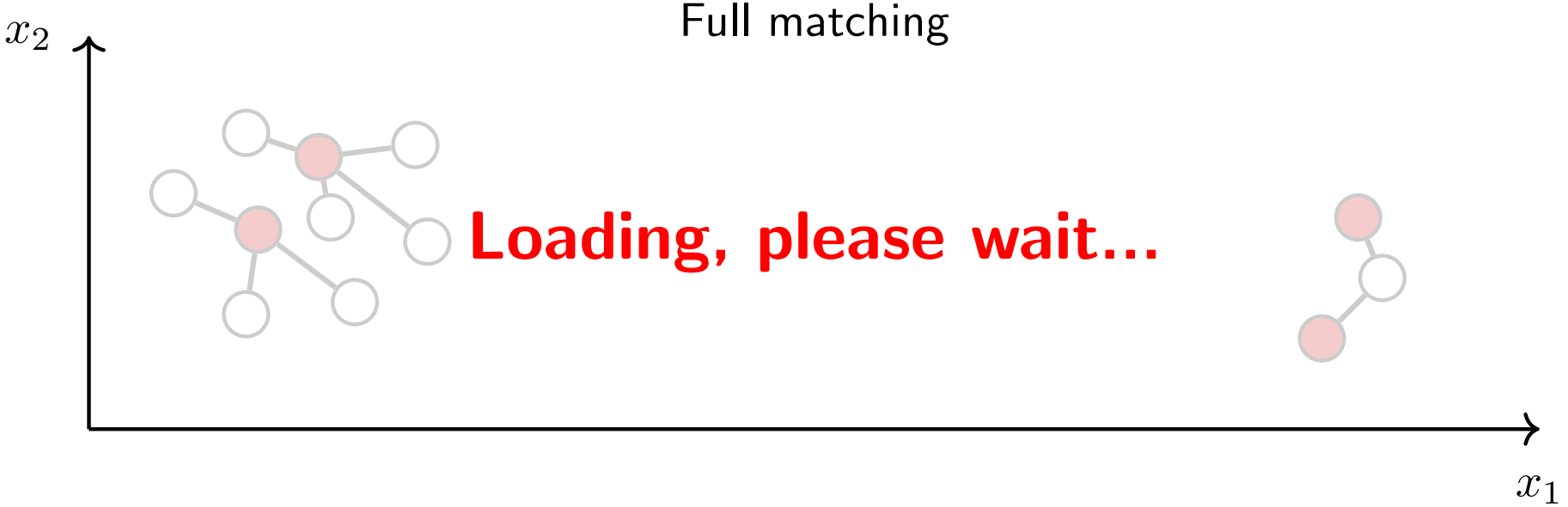
Full matching



NN-matching without replacement



Full matching



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.

* Not original definition, but equivalent.

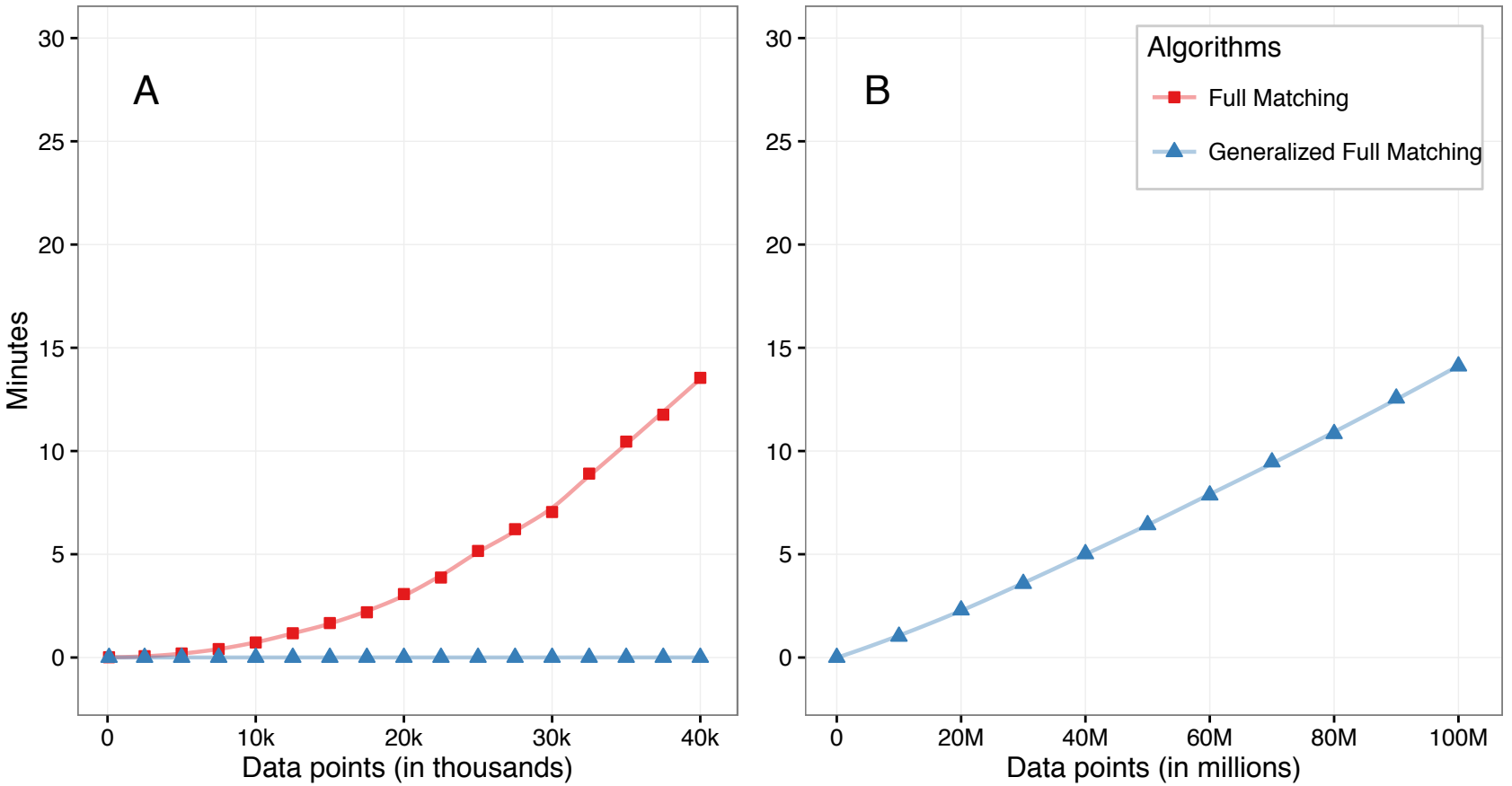
The GFM algorithm

- Minimizes the maximum within-group dissimilarity subject to the design constraints.

- Builds on an idea from Higgins, Sävje & Sekhon (2016):
  - Simplify partitioning problems by pruning irrelevant information.

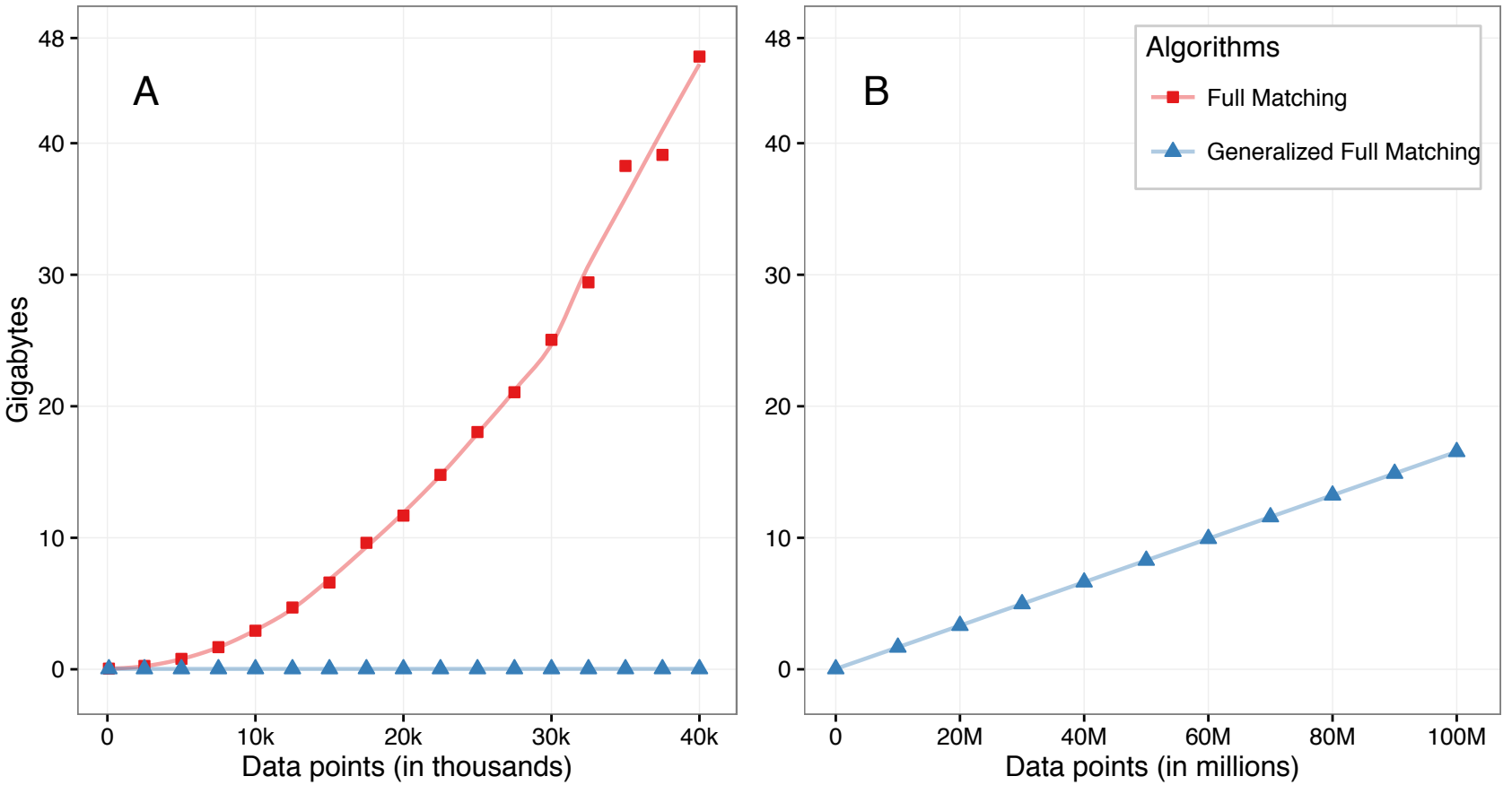
- An overview of the procedure:
  1. Construct a graph that **only** encodes the constraints and relevant similarities.
  2. Pick units that are evenly spaced in the graph ("seeds").
  3. Grow the matching from the seeds.

Simulation: Run time



▸ Setting ▸ Table

Simulation: Memory



▸ Setting ▸ Table

Generalized Full Matching

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: $\leq 4 \times \lambda$
- 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

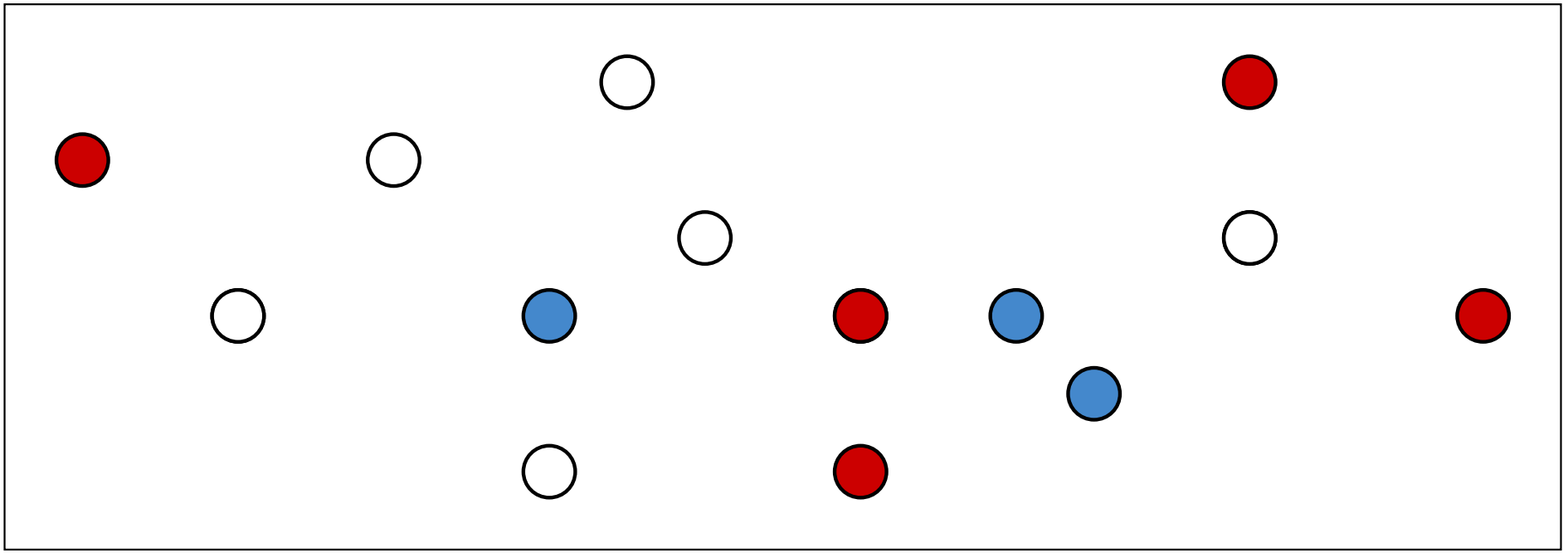
Simulation: Performance

Table: Mean and maximum within-group distances.

	1,000 units		10,000 units	
	FM	GFM	FM	GFM
Mean TC-distance	1.000	0.999	1.000	1.000
Max distance	1.000	1.002	1.000	1.001

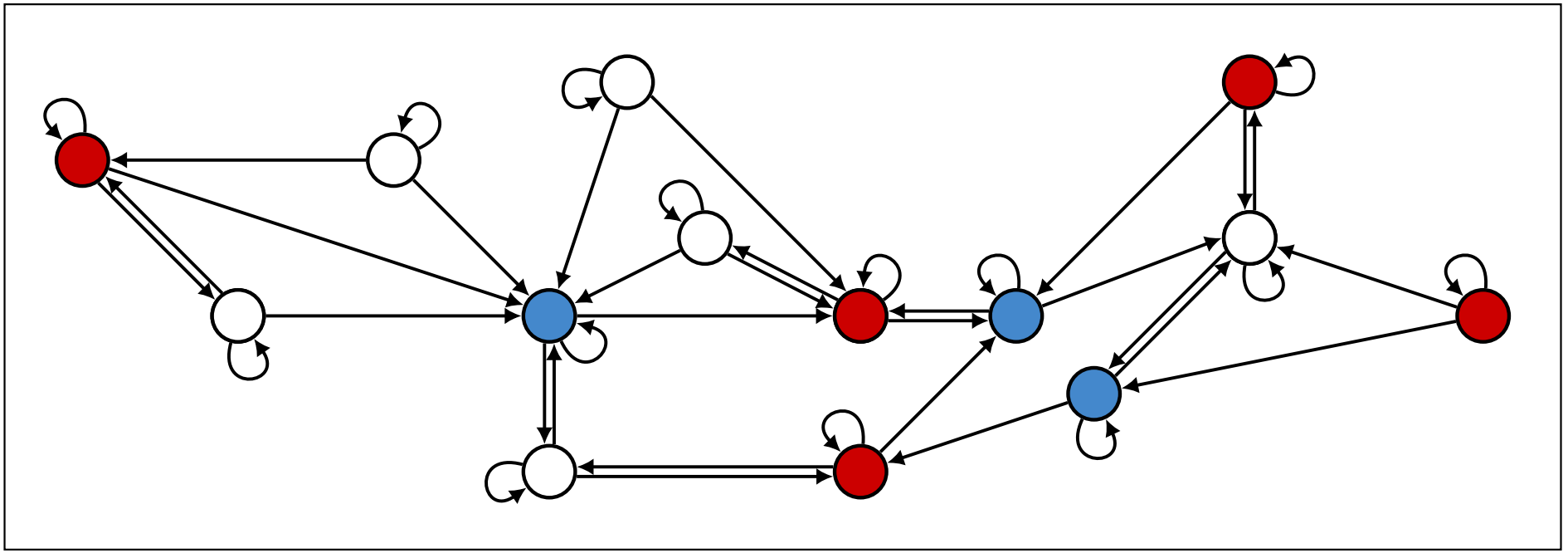
Notes: Based on 2,500 simulation rounds.

▸ Setting ▸ Other distances ▸ Cov. balance ▸ Bias & SE



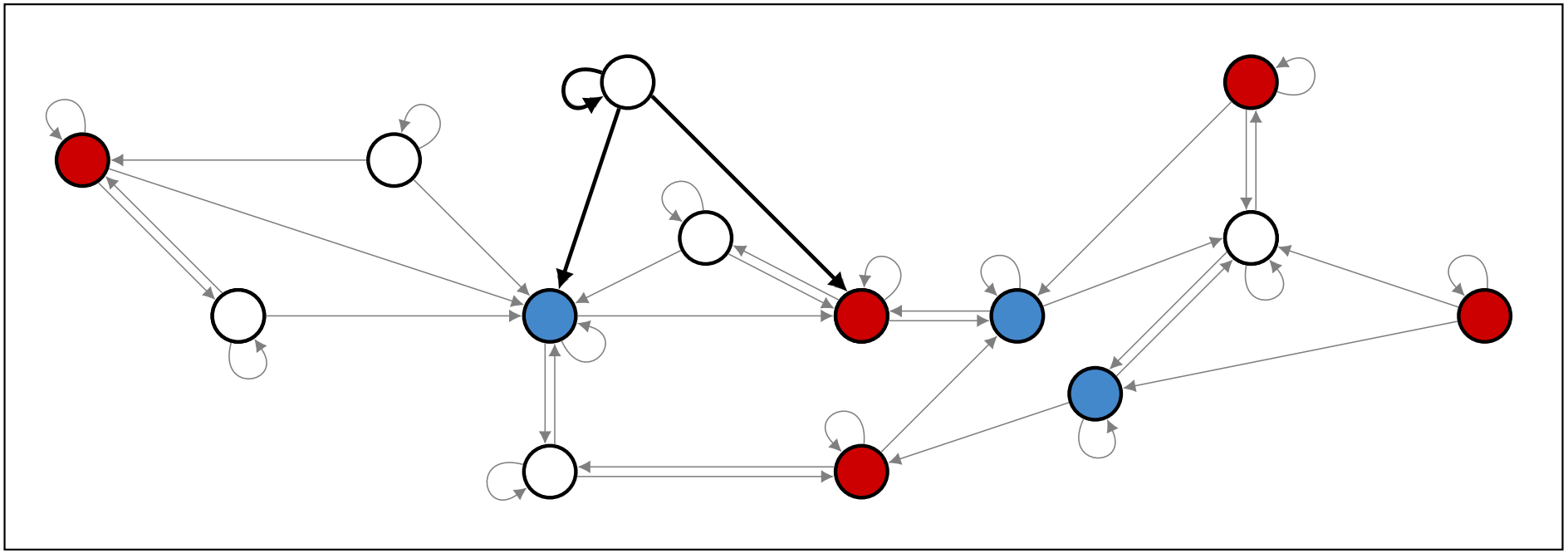
1. Construct a *constraint compatible nearest neighbor digraph*.
   - The smallest graph so that each neighborhood satisfies the size constraints.
2. Pick a maximal set of vertices ("seeds") with non-overlapping neighborhoods.
3. Grow the matched groups from the seeds:
   a. Assign unique labels to the seeds.
   b. Assign each seed's label to its neighbors.
   c. Unassigned units are assigned a label in its neighborhood.

▸ Optimality proof



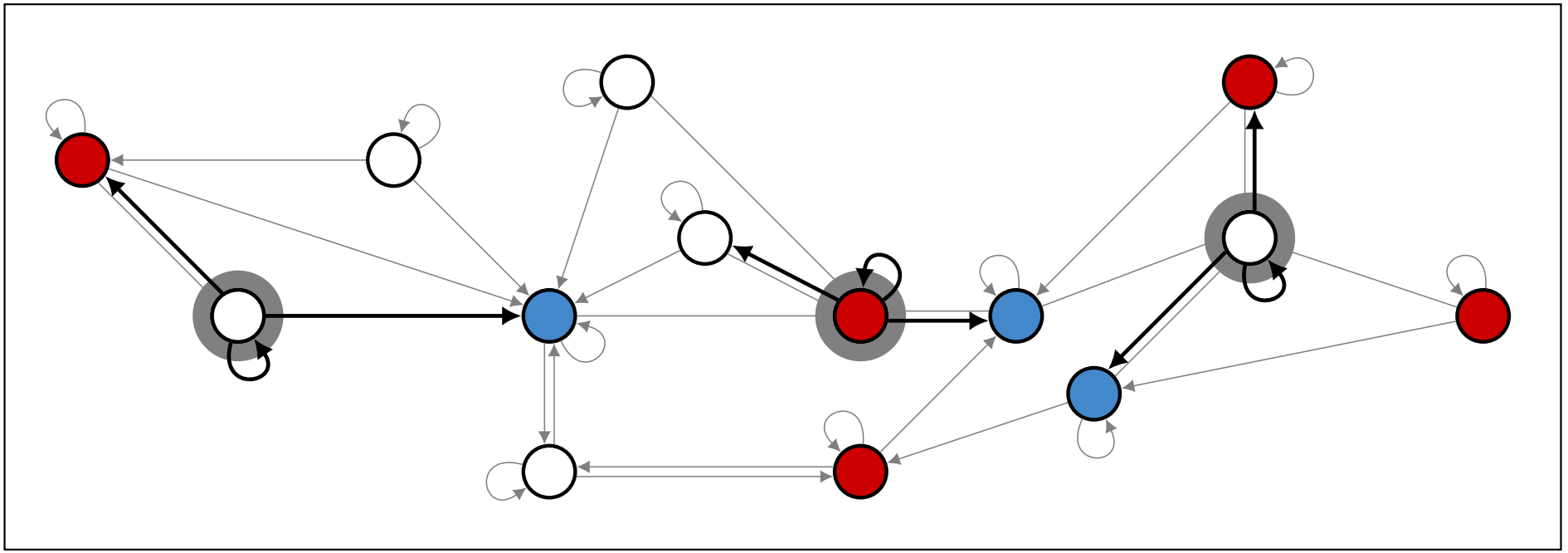
1. Construct a *constraint compatible nearest neighbor digraph*.
   - The smallest graph so that each neighborhood satisfies the size constraints.
2. Pick a maximal set of vertices ("seeds") with non-overlapping neighborhoods.
3. Grow the matched groups from the seeds:
   a. Assign unique labels to the seeds.
   b. Assign each seed's label to its neighbors.
   c. Unassigned units are assigned a label in its neighborhood.

▸ Optimality proof



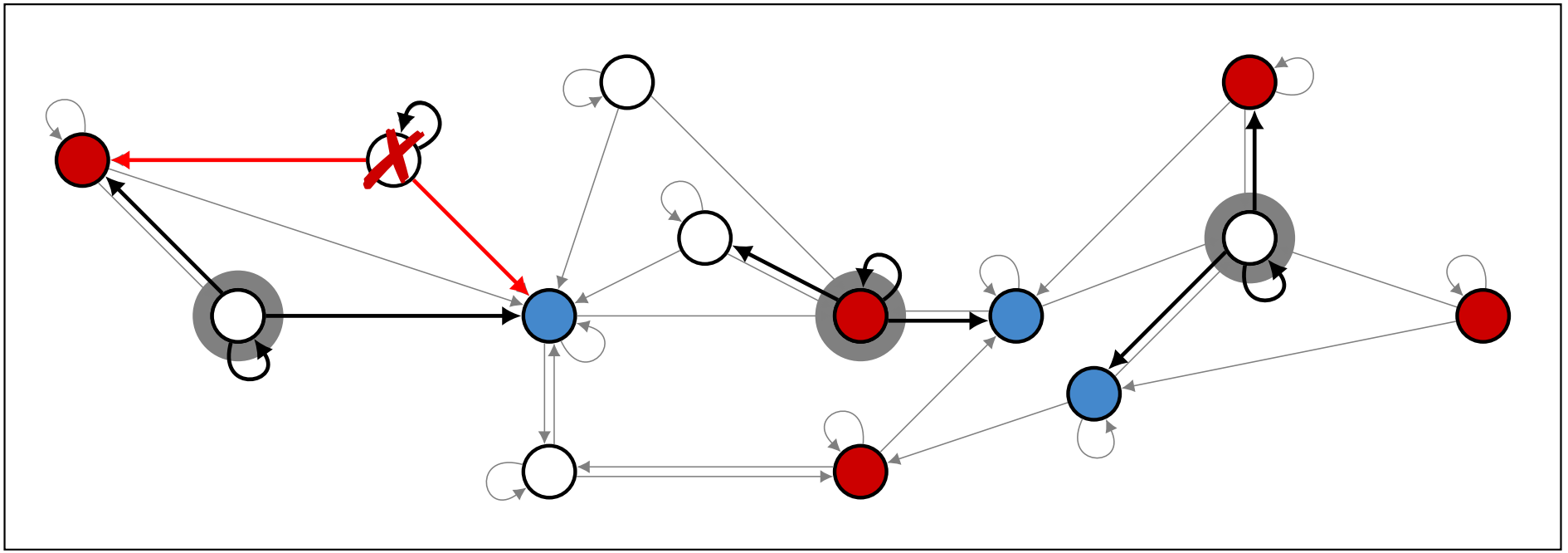
1. Construct a *constraint compatible nearest neighbor digraph*.
   - The smallest graph so that each neighborhood satisfies the size constraints.
2. Pick a maximal set of vertices ("seeds") with non-overlapping neighborhoods.
3. Grow the matched groups from the seeds:
   a. Assign unique labels to the seeds.
   b. Assign each seed's label to its neighbors.
   c. Unassigned units are assigned a label in its neighborhood.

▸ Optimality proof



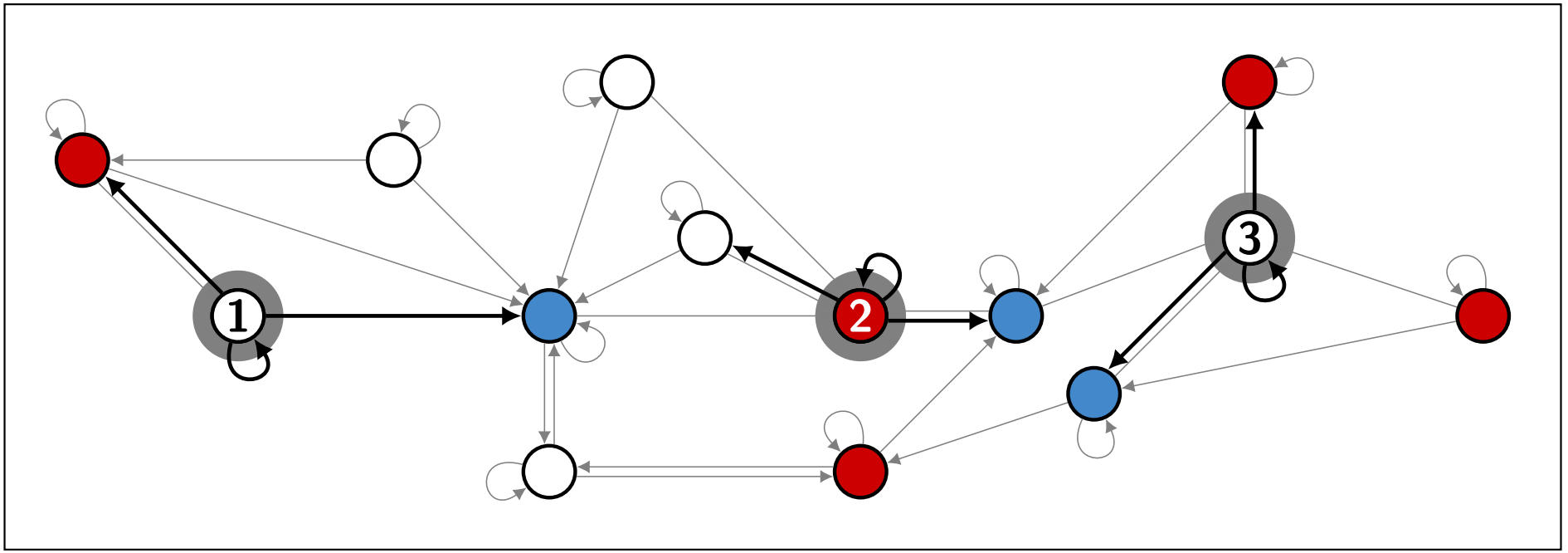
1. Construct a *constraint compatible nearest neighbor digraph*.
   - The smallest graph so that each neighborhood satisfies the size constraints.
2. Pick a maximal set of vertices ("seeds") with non-overlapping neighborhoods.
3. Grow the matched groups from the seeds:
   a. Assign unique labels to the seeds.
   b. Assign each seed's label to its neighbors.
   c. Unassigned units are assigned a label in its neighborhood.

▸ Optimality proof



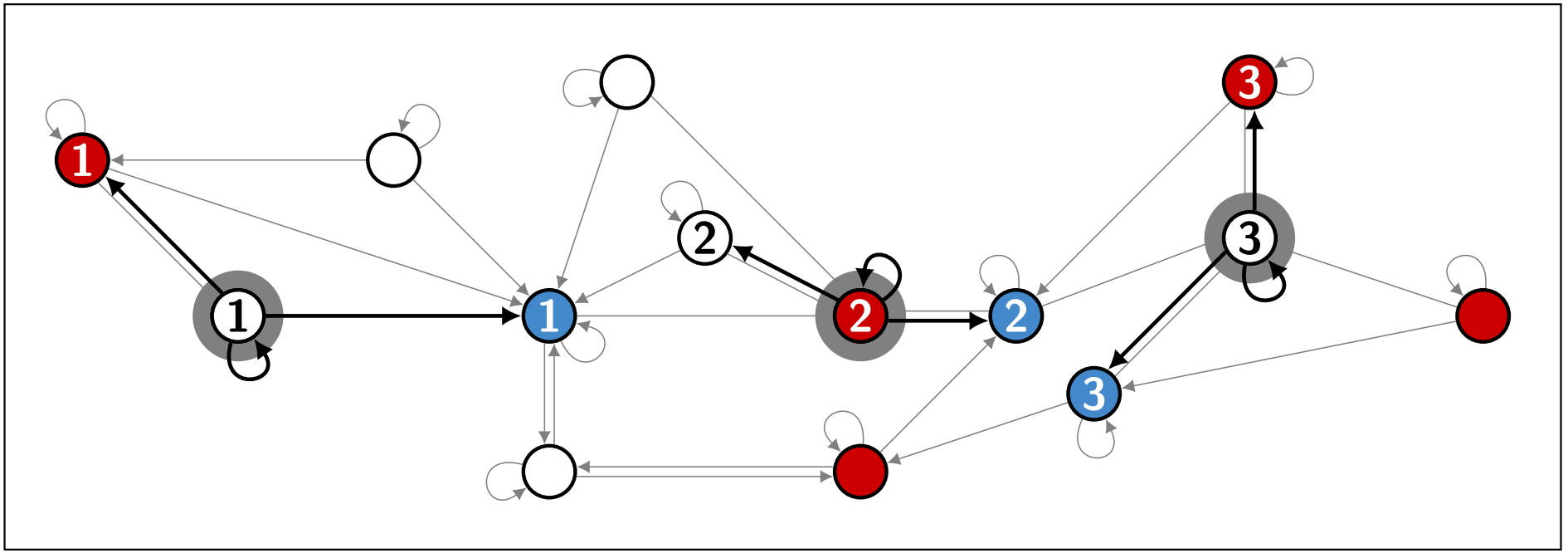
1. Construct a *constraint compatible nearest neighbor digraph*.
   - The smallest graph so that each neighborhood satisfies the size constraints.
2. Pick a maximal set of vertices ("seeds") with non-overlapping neighborhoods.
3. Grow the matched groups from the seeds:
   a. Assign unique labels to the seeds.
   b. Assign each seed's label to its neighbors.
   c. Unassigned units are assigned a label in its neighborhood.

▸ Optimality proof



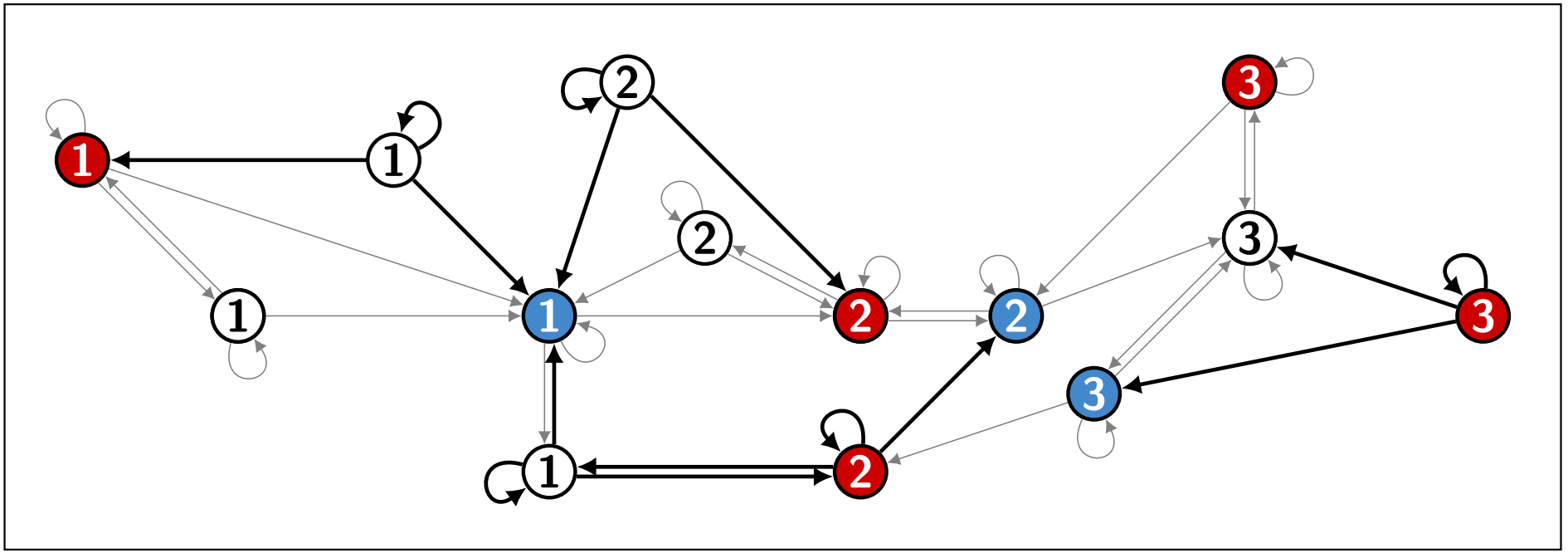
1. Construct a *constraint compatible nearest neighbor digraph*.
   - The smallest graph so that each neighborhood satisfies the size constraints.
2. Pick a maximal set of vertices ("seeds") with non-overlapping neighborhoods.
3. Grow the matched groups from the seeds:
   a. Assign unique labels to the seeds.
   b. Assign each seed's label to its neighbors.
   c. Unassigned units are assigned a label in its neighborhood.

▸ Optimality proof



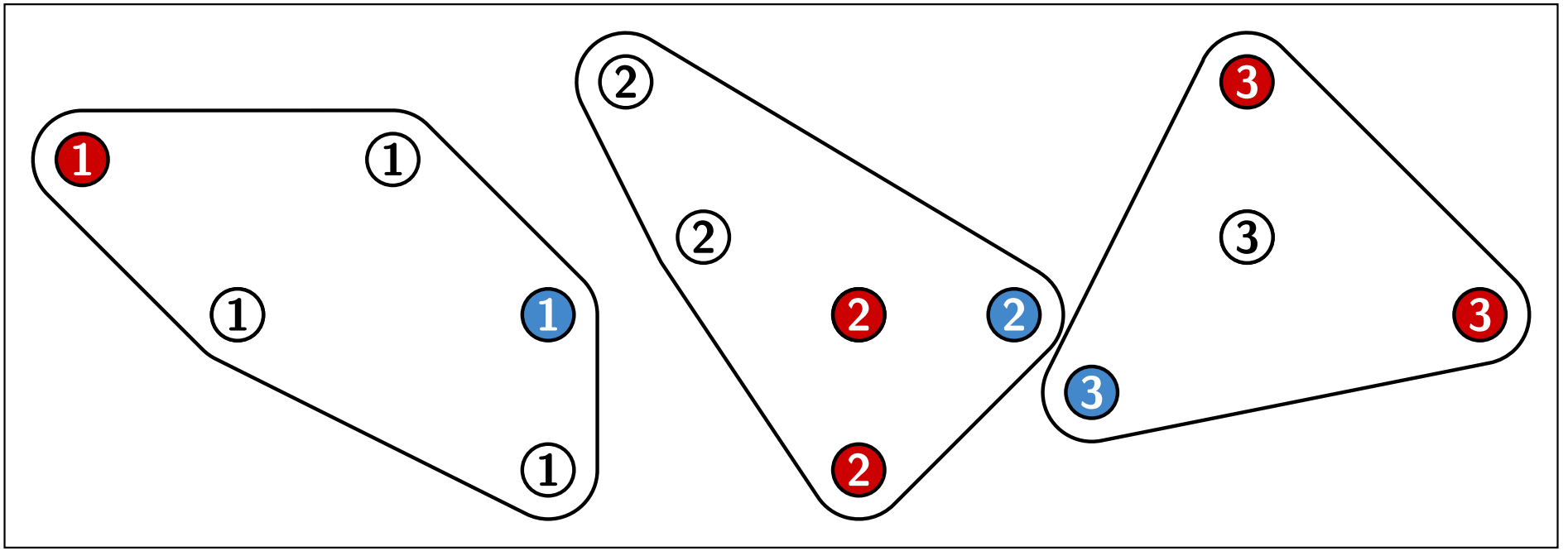
1. Construct a *constraint compatible nearest neighbor digraph*.
   - The smallest graph so that each neighborhood satisfies the size constraints.
2. Pick a maximal set of vertices ("seeds") with non-overlapping neighborhoods.
3. Grow the matched groups from the seeds:
   a. Assign unique labels to the seeds.
   b. Assign each seed's label to its neighbors.
   c. Unassigned units are assigned a label in its neighborhood.

▸ Optimality proof



1. Construct a *constraint compatible nearest neighbor digraph*.
   - The smallest graph so that each neighborhood satisfies the size constraints.
2. Pick a maximal set of vertices ("seeds") with non-overlapping neighborhoods.
3. Grow the matched groups from the seeds:
   a. Assign unique labels to the seeds.
   b. Assign each seed's label to its neighbors.
   c. Unassigned units are assigned a label in its neighborhood.

▸ Optimality proof



1. Construct a *constraint compatible nearest neighbor digraph*.
   - The smallest graph so that each neighborhood satisfies the size constraints.
2. Pick a maximal set of vertices ("seeds") with non-overlapping neighborhoods.
3. Grow the matched groups from the seeds:
   a. Assign unique labels to the seeds.
   b. Assign each seed's label to its neighbors.
   c. Unassigned units are assigned a label in its neighborhood.

▸ Optimality proof

Take away

1. Data alone does not solve the causal inference problem.

2. Existing matching methods do not work well in large samples and with complicated designs

3. The GFM algorithm makes well-performing matchings possible in massive samples.

4. The key trick is not to optimize a loss function directly, but to solve a bottleneck subgraph problem and to prove that it bounds the loss function

Thanks!

Extensions

- Simple, ordinary caliper.
- Complex caliper.
  - E.g., use caliper c_1 to ensure constraints are satisfied, but use $c_2 < c_1$ to match remaining units.
- Allow certain units to be discarded (similar to with replacement matching).
  - Cuts the optimality bound in half.
- Even more complicated design constraints:
  - Anything that can be encoded in a directed graph.
  - E.g., if $x_1 \geq 0$ match with two controls, if $x_1 < 0$ match with one.

Appendix: What is matching? ▸ Back

- 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.

Appendix: Matching as optimization

Back

- **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.

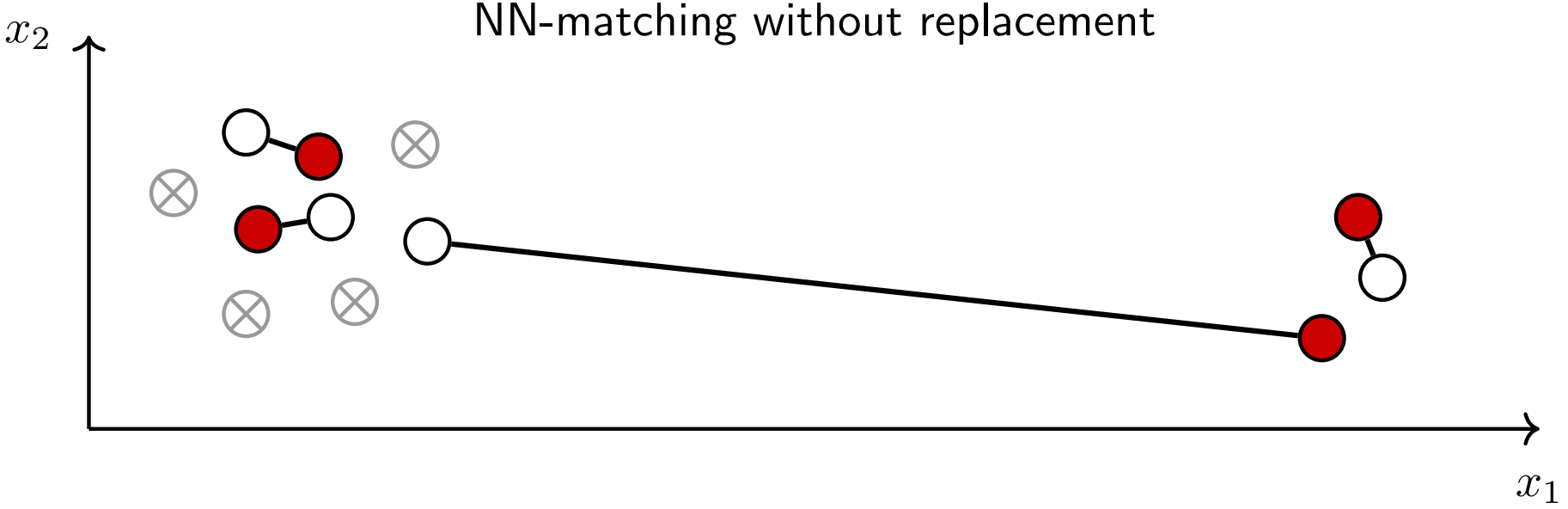
Objective function

Constraints

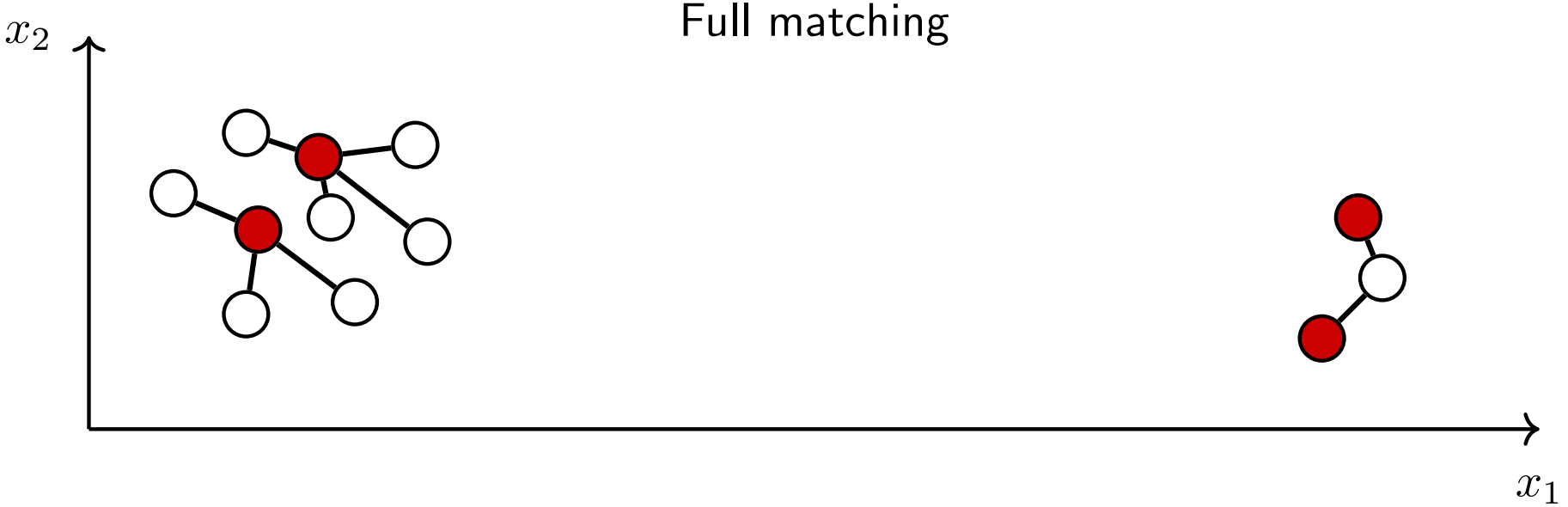
Algorithm

▸ Back ▸ Hansen (2004) ▸ Other methods

NN-matching without replacement

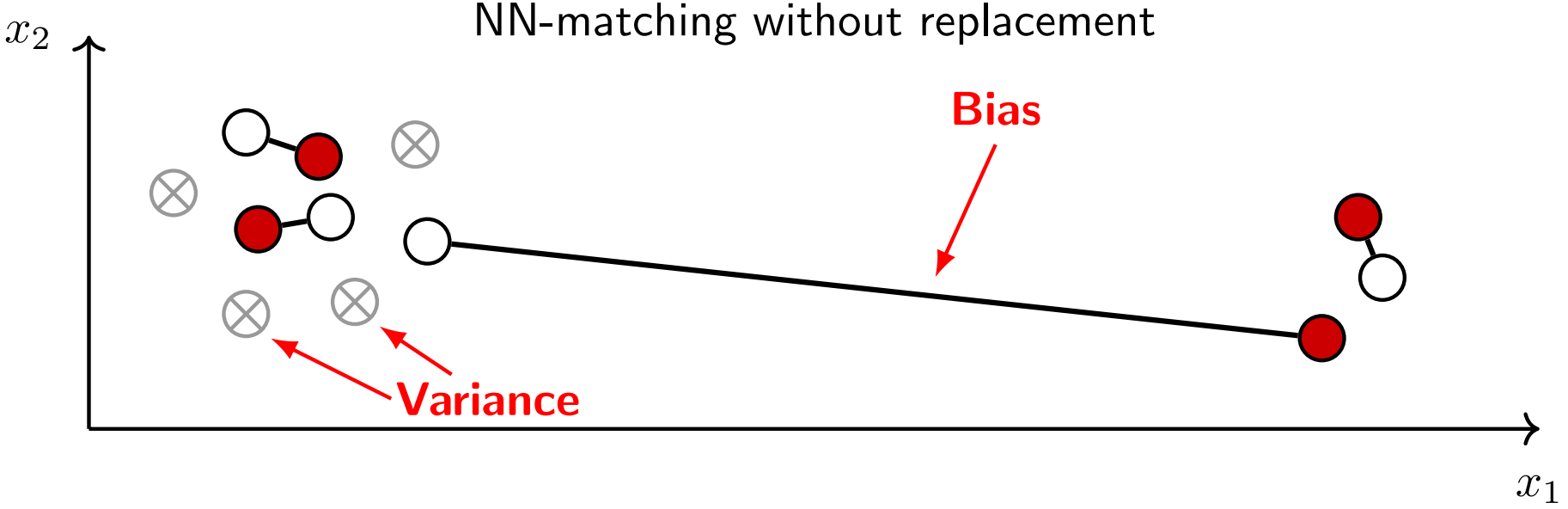


Full matching

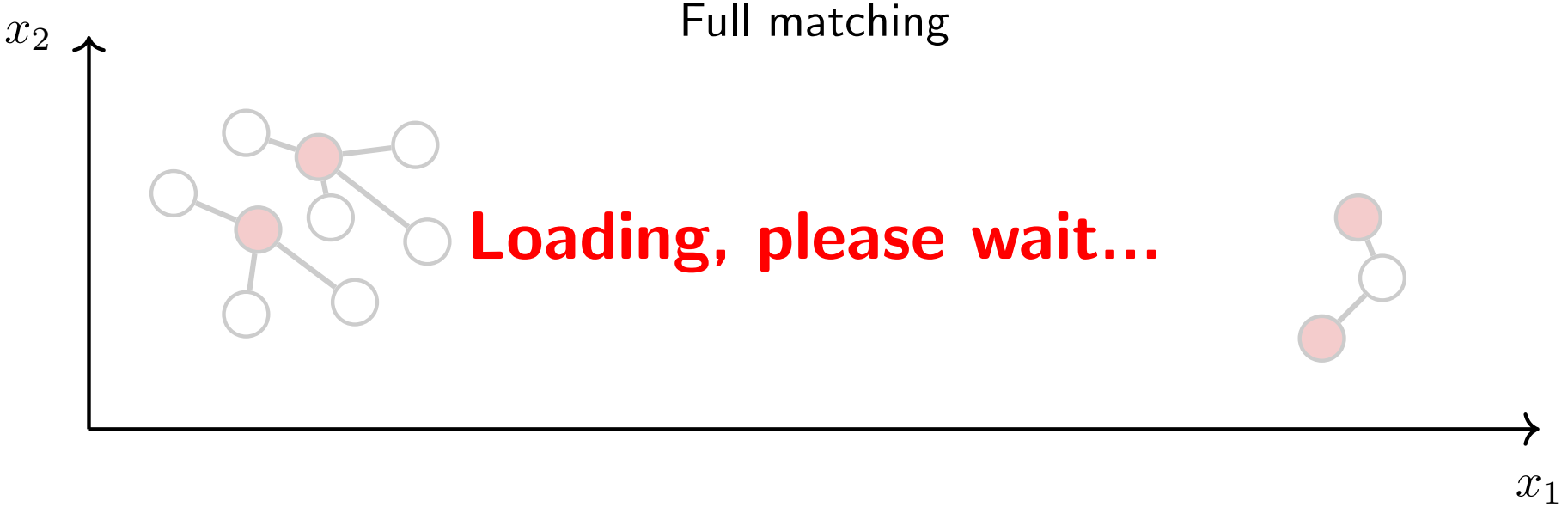


▸ Back ▸ Hansen (2004) ▸ Other methods

NN-matching without replacement



Full matching



▸ Back ▸ Hansen (2004) ▸ Other methods

Appendix: Hansen (2004) ▸ Back

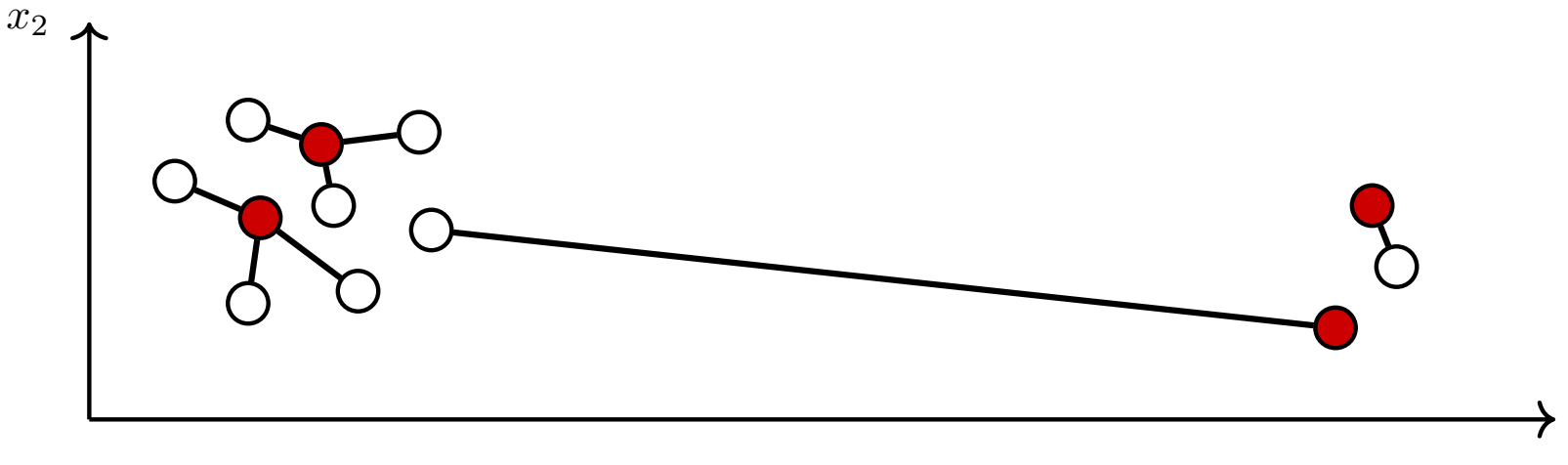
- Full matching does not optimally solve the variance/bias trade-off.
  - It sometimes leads to too much weight variation ⇒ high variance.
- Useful heuristic: bound the weights by limiting the treated/control-ratio.
  - Similar to increasing the size constraints in GFM.
- However:
  - Optimality depends on the DGP.
  - Global tuning, cannot be controlled locally.

Ordinary full matching

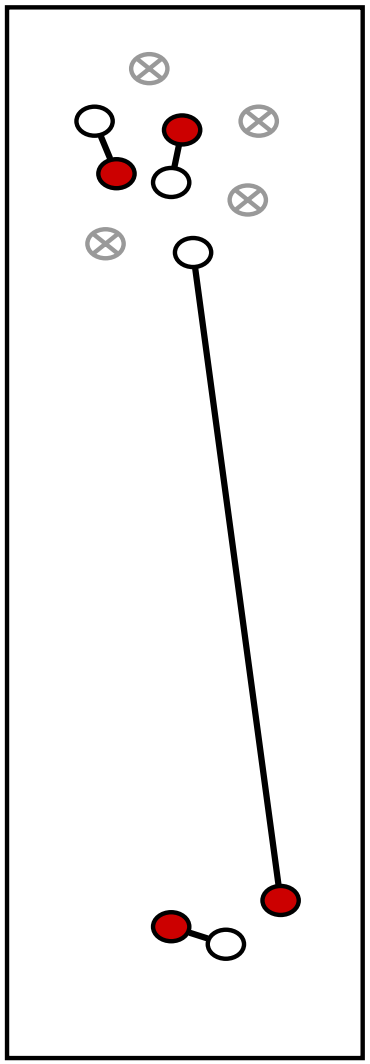
Appendix: Hansen (2004) ▸ Back

- Full matching does not optimally solve the variance/bias trade-off.
  - It sometimes leads to too much weight variation ⇒ high variance.
- Useful heuristic: bound the weights by limiting the treated/control-ratio.
  - Similar to increasing the size constraints in GFM.
- However:
  - Optimality depends on the DGP.
  - Global tuning, cannot be controlled locally.

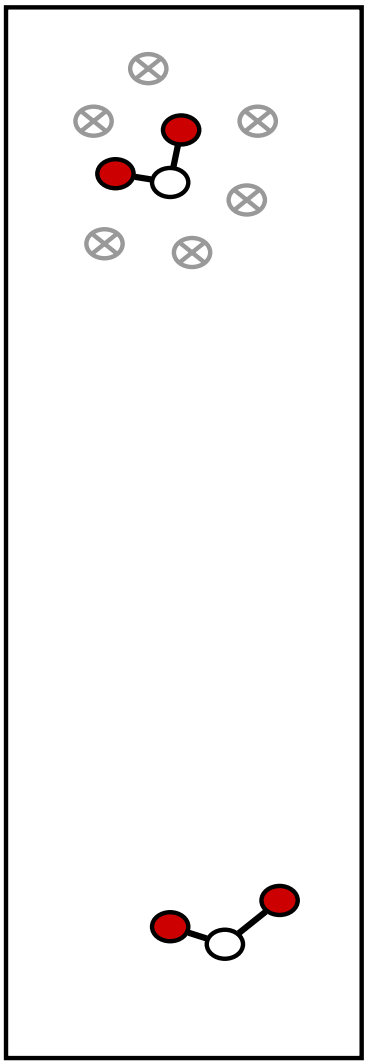
Full matching with 1:3 ratio restriction



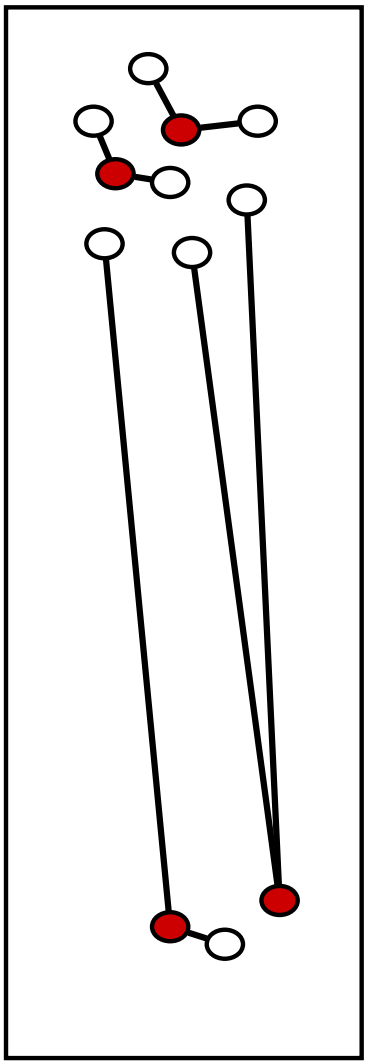
Without replacement



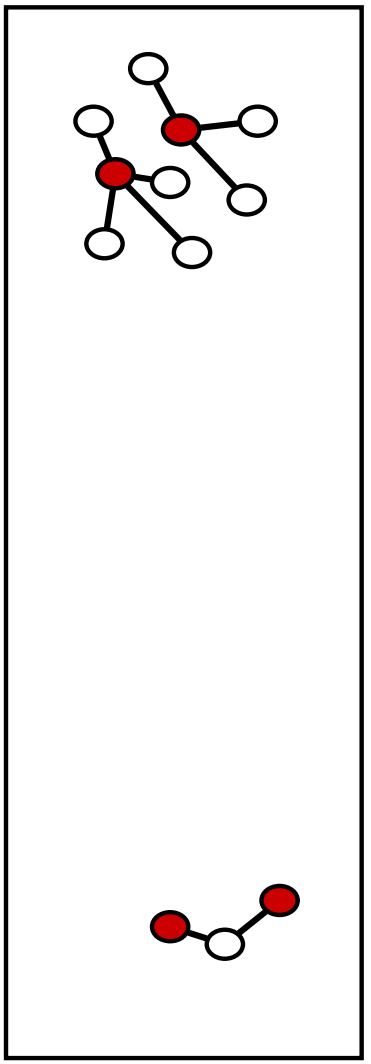
With replacement



1:k-matching



Full matching



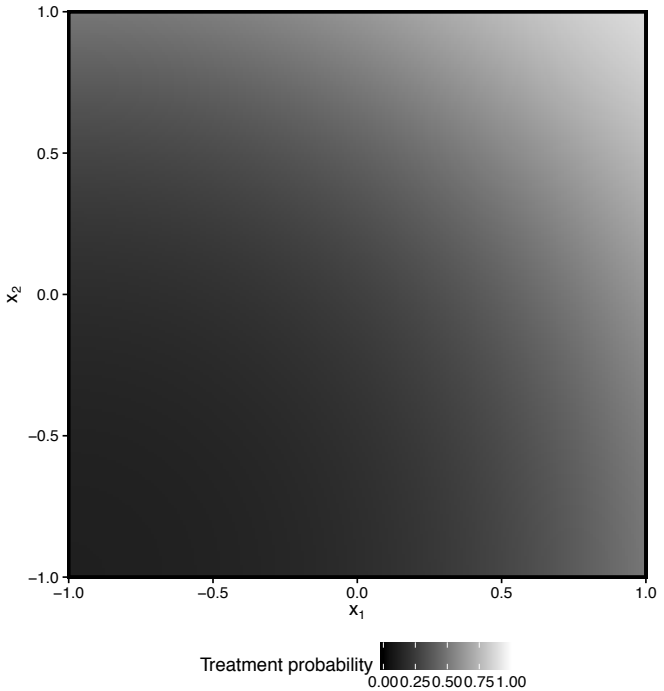
▸ Back

Appendix: Simulation setting

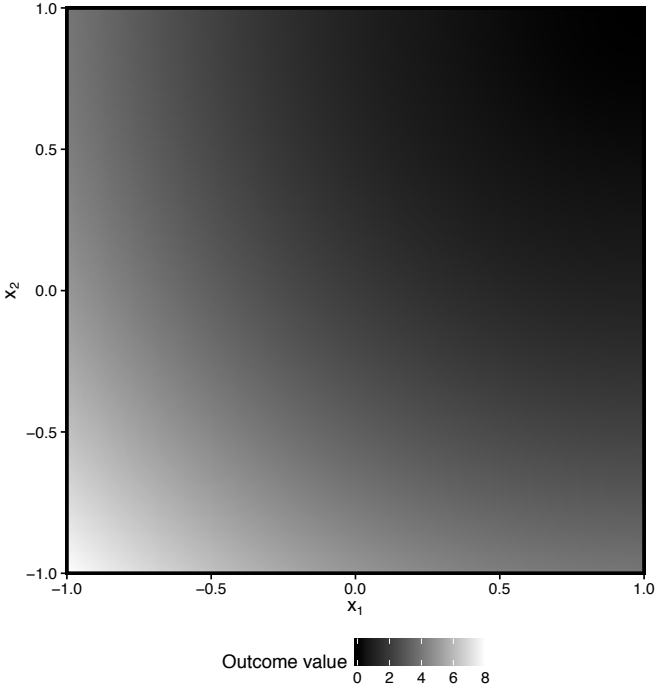
▸ Run time ▸ Memory ▸ Distances

- 2D uniform covariates: $x_1, x_2 \sim \mathcal{U}(-1, 1)$. Euclidean distances.
- Propensity score: $\Pr(t = 1|x_1, x_2) = \text{logistic}\left[\frac{(x_1+1)^2+(x_2+1)^2-4}{2}\right]$.
- Outcome: $y|x_1, x_2 \sim (x_1 - 1)^2 + (x_2 - 1)^2 + \{\text{standard normal}\}$.

: Propensity score



: Outcome CEF



Appendix: Runtime

Back

Table: Runtime in minutes by sample size.

	1K	10K	20K	40K	100K	1M	10M	100M
FM	0.026	0.727	3.073	13.544				
GFM	0.002	0.002	0.002	0.003	0.007	0.075	1.023	14.113

Notes: Based on 100 simulation rounds. K = thousands, M = millions.

Appendix: Memory

Back

Table: Memory use in gigabytes by sample size.

	1K	10K	20K	40K	100K	1M	10M	100M
FM	0.094	2.932	11.685	46.588				
GFM	0.028	0.029	0.029	0.032	0.039	0.168	1.671	16.543

Notes: Based on 100 simulation rounds. K = thousands, M = millions.

Appendix: Other aggregated distances

Back

Table: Mean and maximum within-group distances.

	1,000 units		10,000 units	
	FM	GFM	FM	GFM
Mean TC-distance	1.000	0.999	1.000	1.000
Mean distance	1.000	0.987	1.000	0.987
Max TC-distance	1.000	1.013	1.000	1.009
Max distance	1.000	1.002	1.000	1.001

Notes: Based on 2,500 simulation rounds.

Appendix: Covariate balance

Back

Table: Covariate balance for No Matching (NM), FM and GFM.

	1,000 units			10,000 units		
	NM	FM	GFM	NM	FM	GFM
x_1	53.08	1.00	0.72	498.90	1.00	0.71
x_2	52.56	1.00	0.72	495.90	1.00	0.70
x_1^2	8.97	1.00	0.81	78.72	1.00	0.77
x_2^2	8.51	1.00	0.81	79.74	1.00	0.76
$x_1 x_2$	8.02	1.00	0.89	60.08	1.00	0.91

Notes: Based on 2,500 simulation rounds. Normalized by FM.

Appendix: Bias and SE

Back

Table: Bias and standard error for No Matching (NM), FM and GFM.

	1,000 units			10,000 units		
	NM	FM	GFM	NM	FM	GFM
Normalized Bias	80.23	1.000	0.737	6296.50	1.000	1.268
Normalized SE	1.48	1.000	1.027	1.43	1.000	1.041
Bias / SE	8.83	0.163	0.117	28.68	0.006	0.008

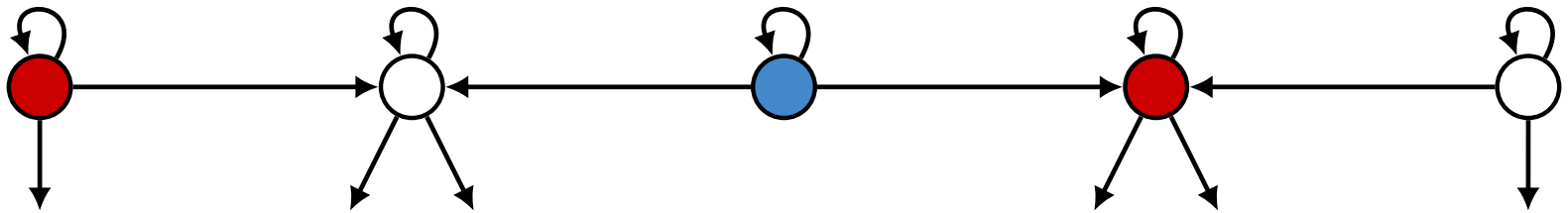
Notes: Based on 2,500 simulation rounds. First two rows are normalized by FM.

Appendix: Graphical optimality proof

Back

The Worst Case

Constraint compatible nearest neighbor digraph

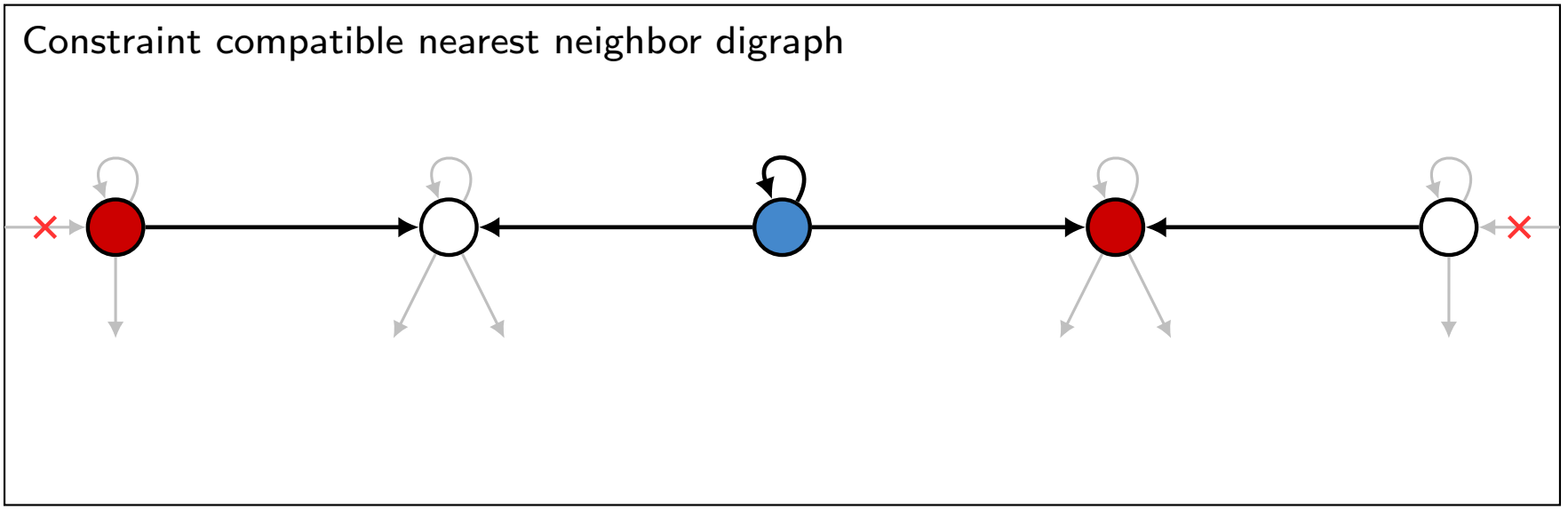


λ: optimal solution.

Appendix: Graphical optimality proof

Back

The Worst Case

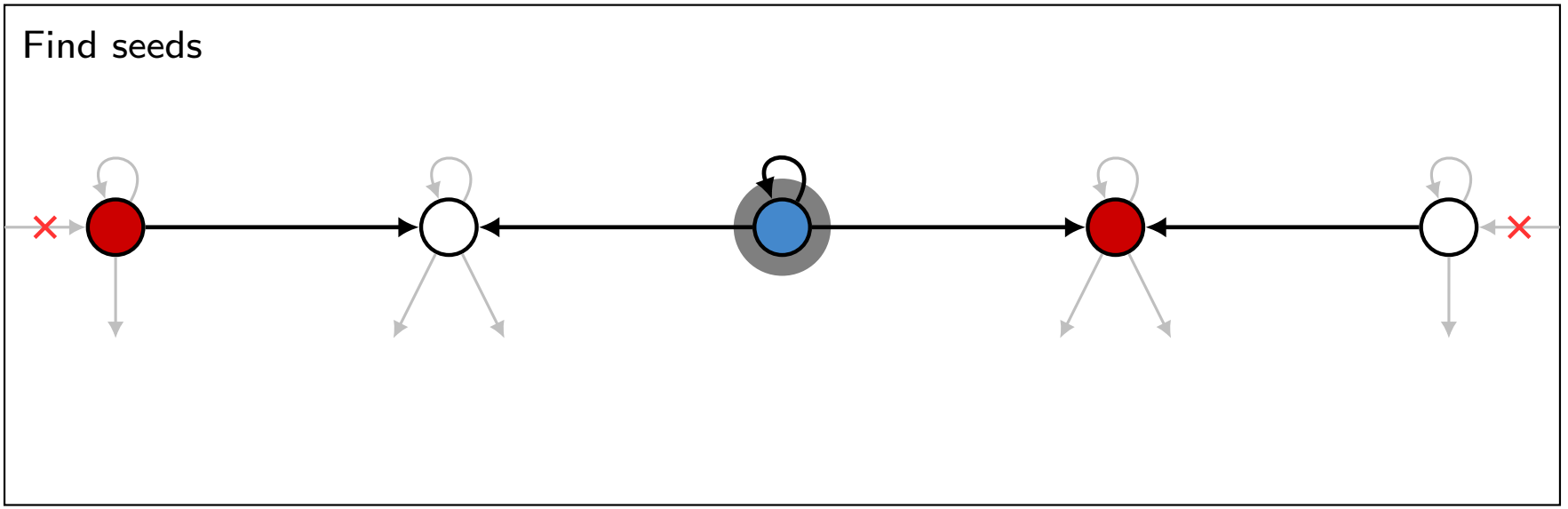


λ: optimal solution.

Appendix: Graphical optimality proof

Back

The Worst Case

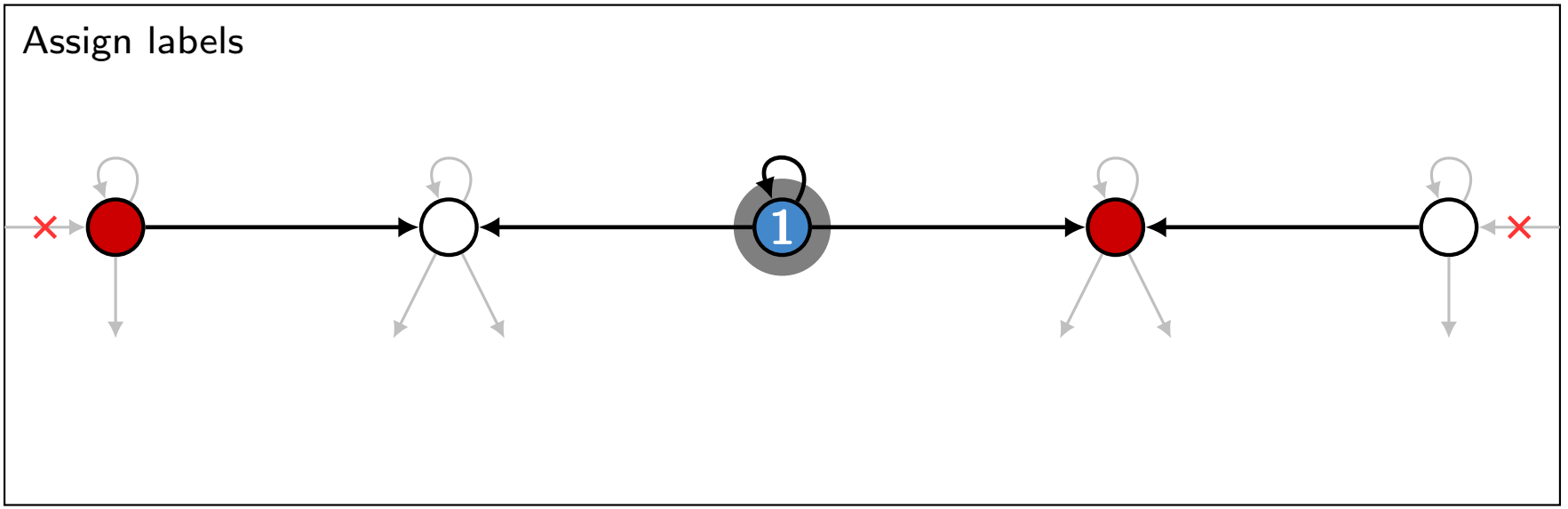


λ: optimal solution.

Appendix: Graphical optimality proof

Back

The Worst Case

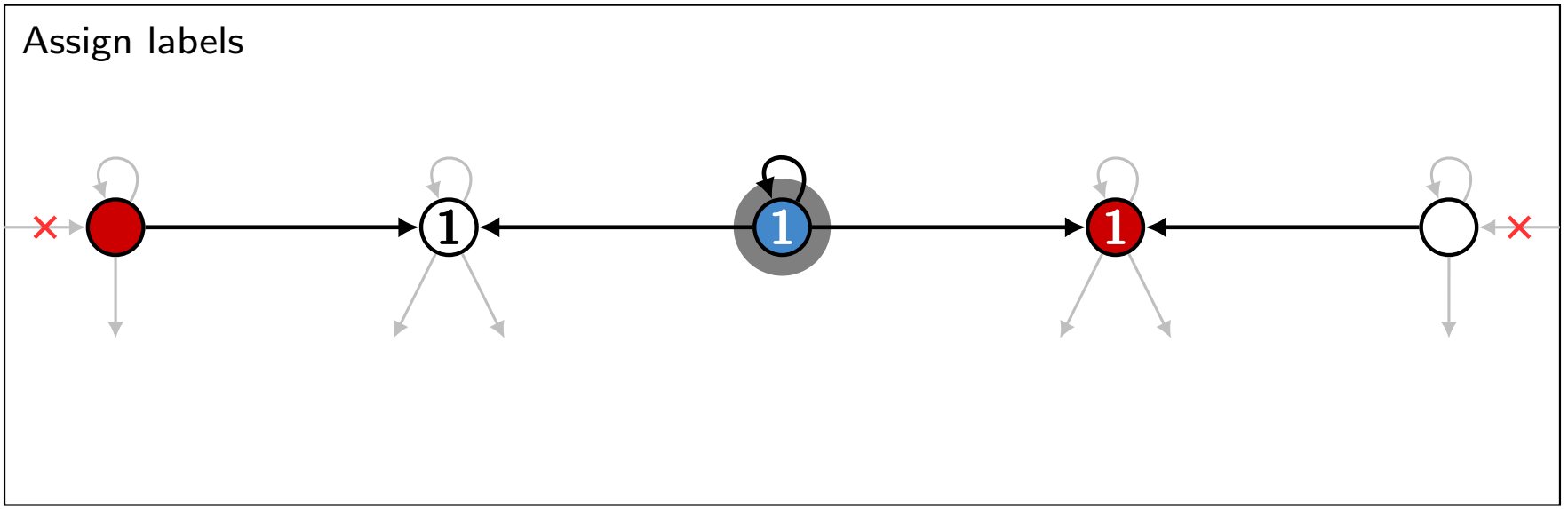


λ: optimal solution.

Appendix: Graphical optimality proof

Back

The Worst Case

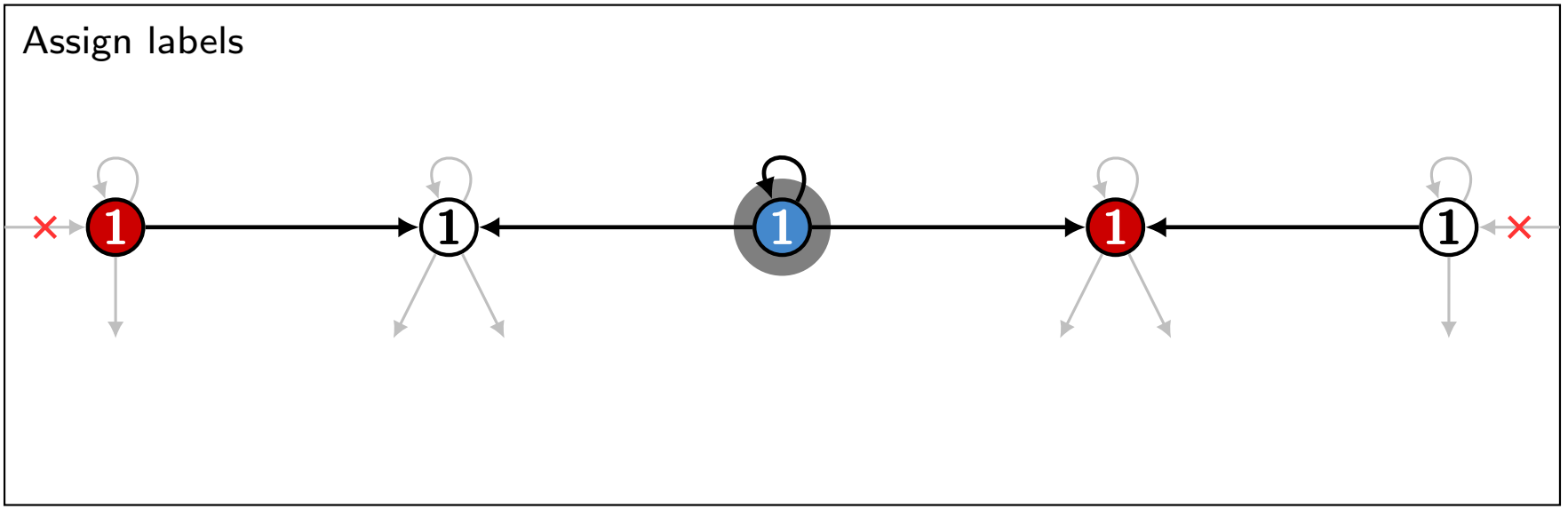


λ: optimal solution.

Appendix: Graphical optimality proof

Back

The Worst Case

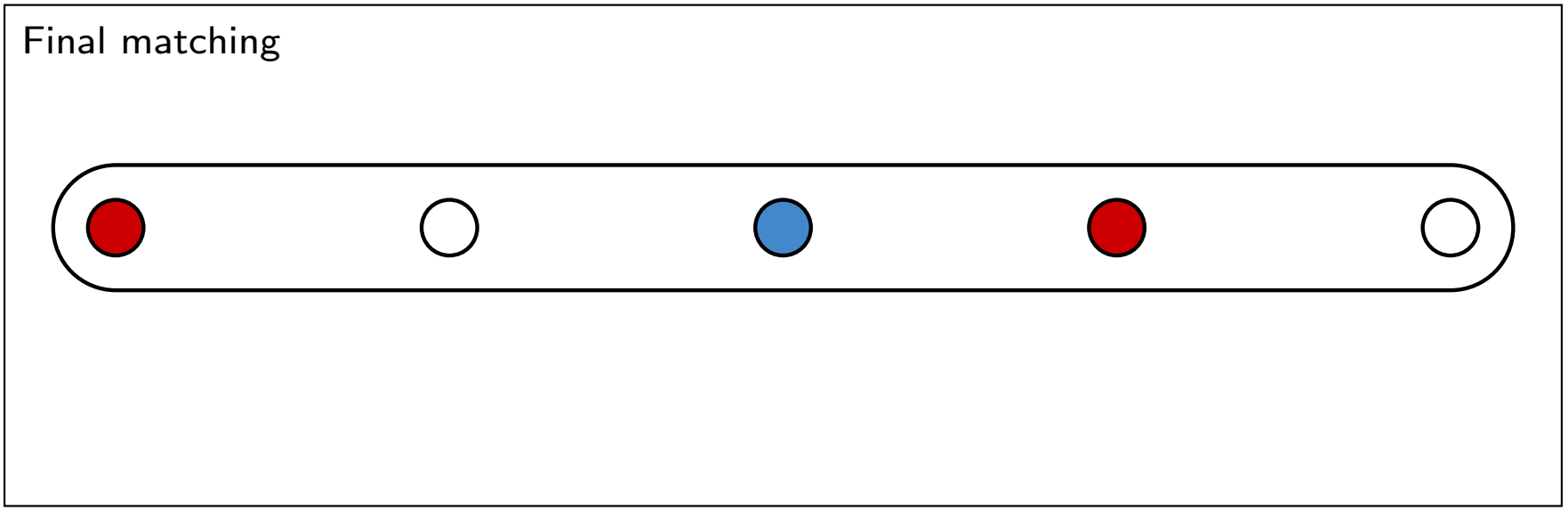


λ: optimal solution.

Appendix: Graphical optimality proof

Back

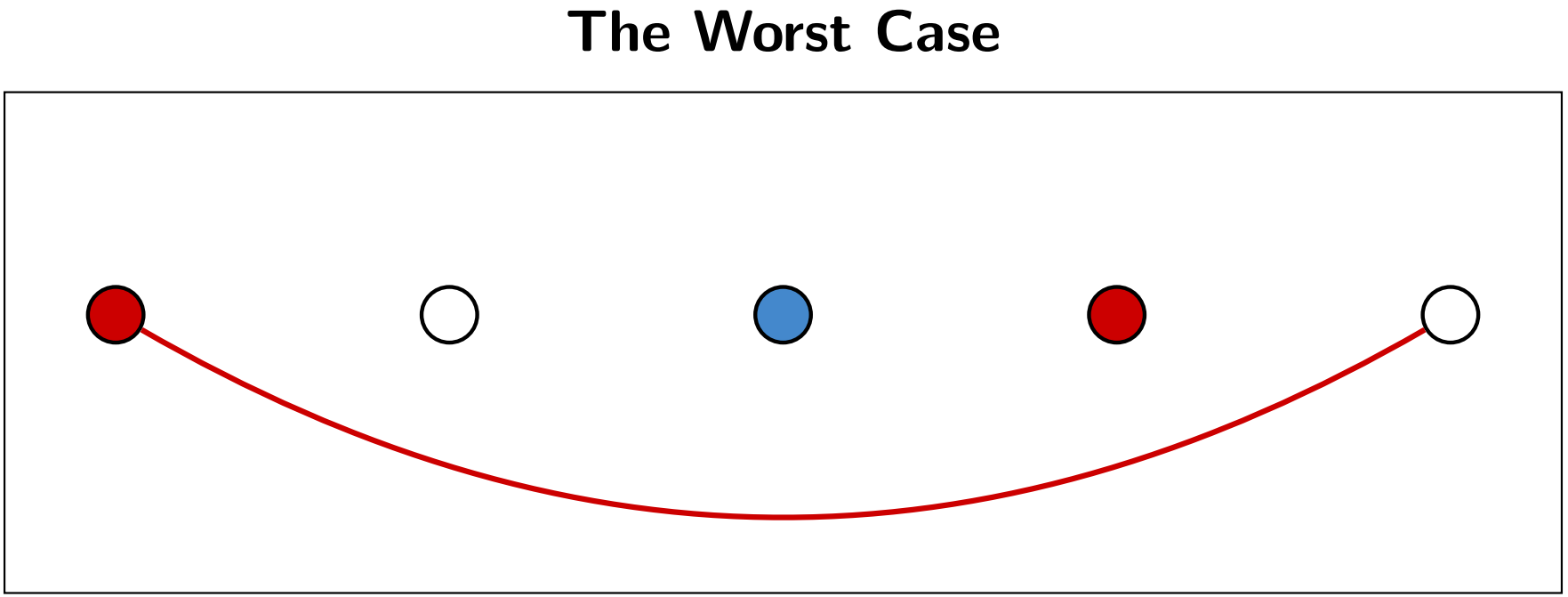
The Worst Case



λ: optimal solution.

Appendix: Graphical optimality proof

Back

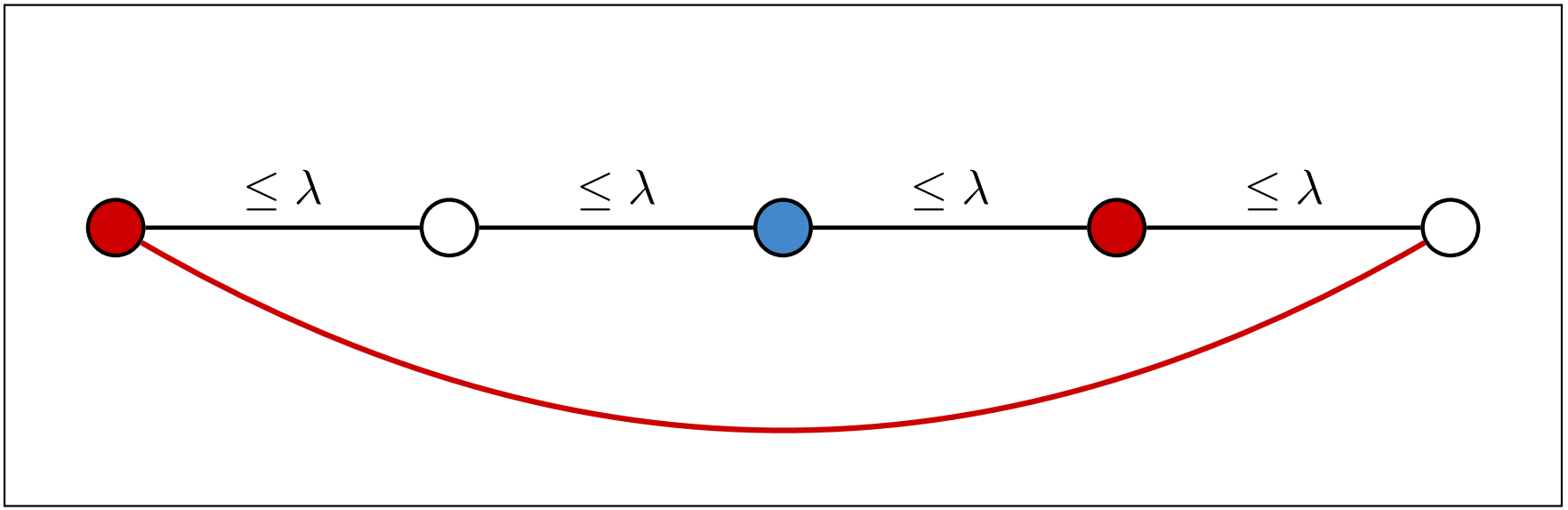


λ: optimal solution.

Appendix: Graphical optimality proof

Back

The Worst Case

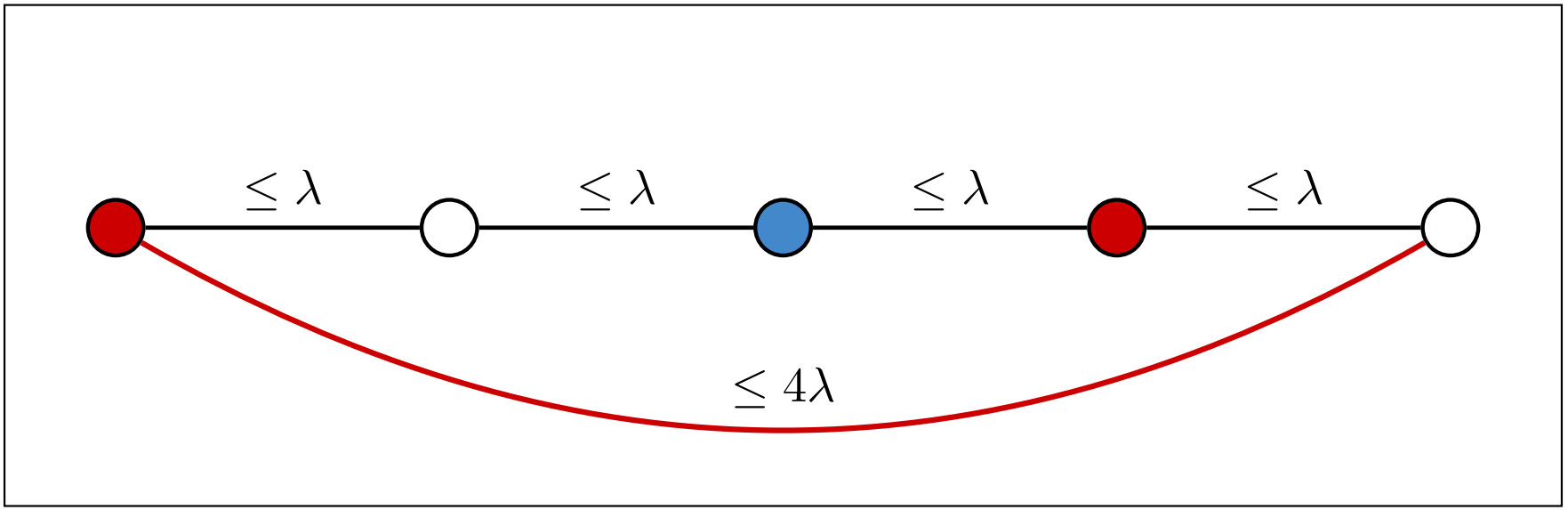


λ: optimal solution.

Appendix: Graphical optimality proof

Back

The Worst Case

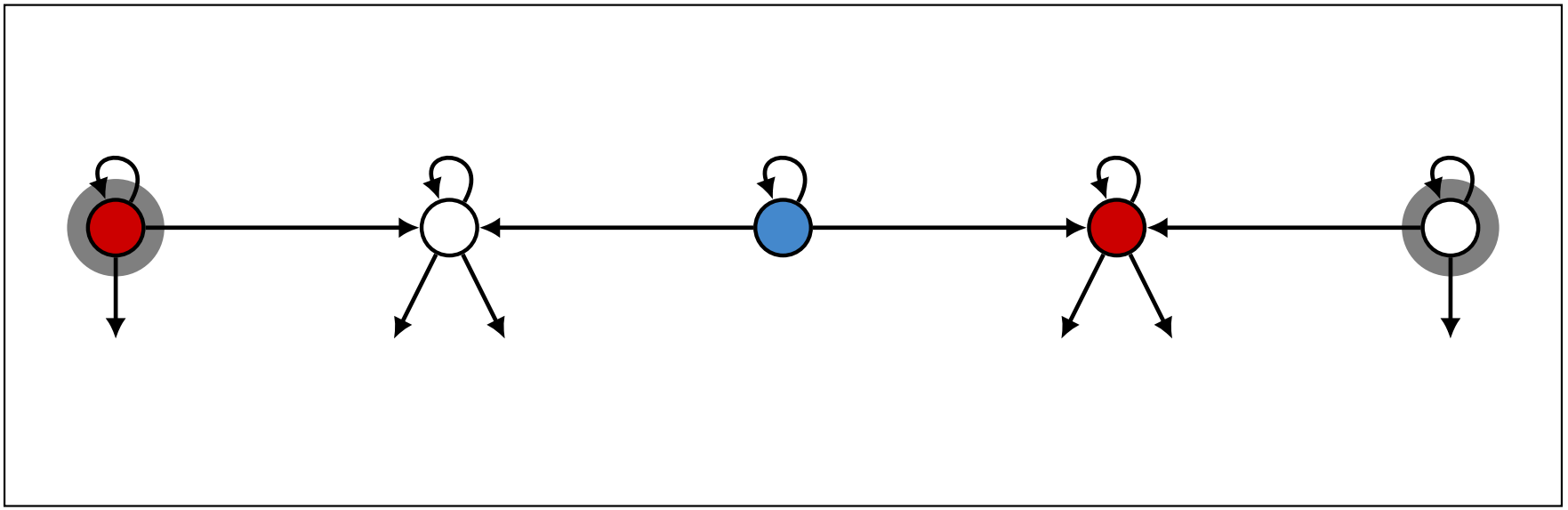


λ: optimal solution.

Appendix: Graphical optimality proof

Back

The ~~Worst~~ Case



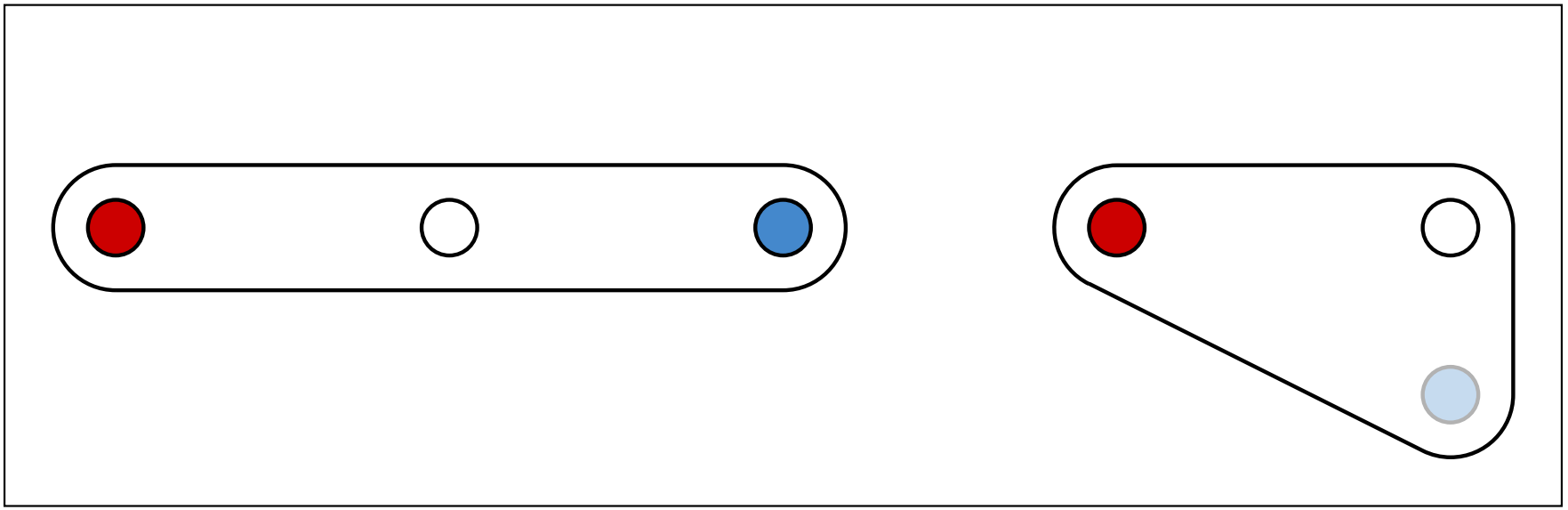
λ: optimal solution.

- We can avoid the worst case by picking seeds more carefully.

Appendix: Graphical optimality proof

Back

The ~~Worst~~ Case



λ: optimal solution.

- We can avoid the worst case by picking seeds more carefully.