*Statistical Science*
2020, Vol. 35, No. 3, 356–360
https://doi.org/10.1214/19-STS739
Main article: https://doi.org/10.1214/19-STS699


# Comment: Matching Methods for Observational Studies Derived from Large Administrative Databases

**Fredrik Sävje**

I first want to commend Yu, Silber and Rosenbaum (2019) for their paper. The matching procedure they have developed will help countless researchers improve their causal inferences in settings where randomized experiments are infeasible or impractical but where observational data is plentiful.

The aim of my remaining comments is to extend and complement the authors' discussion. I start by comparing their procedure with similar approaches developed in the computer science literature. This broader perspective provides some suggestions for possible improvements and extensions. I continue with a discussion about how optimality may be viewed with respect to statistical performance, match quality and runtime, and I describe an alternative procedure that may better align with some of these objectives. I conclude with some general remarks. To the greatest extent possible, I use the authors' notation.

## 1. A BROADER PERSPECTIVE

### 1.1 Previous Work on the Matching Problem

In its most condensed form, the problem the authors consider is to find a $\mu$ in $\mathbb{M}$ that minimizes some objective function $L$ where $\mathbb{M}$ collects all injective functions mapping treated units $\mathcal{T}$ to controls $\mathcal{C}$. A common choice for $L$ is the sum of some cost function $\delta : \mathcal{T} \times \mathcal{C} \to \mathbb{R}^+$ over the matches, in which case the problem becomes

$$\mathbb{M}^* = \arg\min_{\mu \in \mathbb{M}} \sum_{t \in \mathcal{T}} \delta\big(t, \mu(t)\big).$$

The task is the same as finding a minimum-cost maximum independent edge set in the complete bipartite graph with $\mathcal{T}$ and $\mathcal{C}$ as parts.[1] Such an edge set can be found as the solution to a minimum-cost network flow problem.

*Fredrik Sävje is Assistant Professor, Department of Political Science and Department of Statistics and Data Science, Yale University, Rosenkranz Hall, 115 Prospect Street, New Haven, Connecticut 06520, USA (e-mail: fredrik.savje@yale.edu).*

[1] An independent edge set is called a "matching" in the computer science literature, but the term has an extended meaning in the causal inference literature.

Through an impressively productive research program, the authors and their collaborators have described how a large number of variations of the objective function and constraints on $\mathbb{M}$ also can be encoded as minimum-cost network flow problems. Many of these variations, such as fine balancing, are reviewed in detail by the authors.

The network flow approach admits great flexibility, but the associated algorithms do not scale sufficiently well to accommodate large samples. The authors note that runtime tends to grow as $\mathcal{O}(NE + N^2 \log N)$ where $N$ is the number of vertices in the network and $E$ is the number of edges. Generally in matching problems, $E = \Omega(N^2)$ and $N = \Omega(T + C)$, where $\Omega$ denotes asymptotic lower bounds, so the time complexity is cubic in the sample size.

One way to reduce runtime is to prune edges in $E$ in a preprocessing step. For example, if one can achieve $E = \mathcal{O}(N \log N)$, runtime grows at only a quasi-quadratic rate. Such pruning must, however, be done with care. One potential problem is that the optimal flow derived from the reduced edge set may not be a maximum independent edge set in the full problem; that is, the matching produced from the pruned edge set may not be in $\mathbb{M}$. A second concern is that the solution might not be an optimal solution in the full problem; that is, the matching may not be in $\mathbb{M}^*$.

The authors set out to develop a procedure to prune edges while ensuring that the network flow solution is in $\mathbb{M}$. To that end, they solve another optimization problem:

$$\mathbb{M}^{\mathrm{B}} = \arg\min_{\mu \in \mathbb{M}} \max_{t \in \mathcal{T}} \delta\big(t, \mu(t)\big).$$

Problems that aim to minimize the maximum edge cost are called *bottleneck problems*. Bottleneck problems rarely have unique solutions. This does not concern the authors, however, because they seek a preprocessing step. With $\mathbb{M}^{\mathrm{B}}$ in hand, they substitute it for $\mathbb{M}$ in the original problem and find a $\mu$ in $\mathbb{M}^{\mathrm{B}}$ that minimizes $L$. Because $\mathbb{M}^{\mathrm{B}}$ is smaller than $\mathbb{M}$, the procedure will reduce runtime as long as the preprocessing can be completed quickly. Furthermore, because $\mathbb{M}^{\mathrm{B}}$ is a subset of $\mathbb{M}$, the matching found in this way will be admissible.

To better understand the authors' approach, it is helpful to reformulate the problem slightly. Let $\mathbb{M}(x) = \{\mu \in \mathbb{M} : L(\mu) \leq x\}$ be the set of admissible matchings with a lower value of the objective than $x$. Solving the matching problem is the same as finding the smallest $x$ such that $\mathbb{M}(x)$ is nonempty. Finding this minimum is generally no easier than the original problem because $\mathbb{M}(x)$ tends to be hard to construct or characterize. Bottleneck problems are exceptions.

With a bottleneck objective, each element in $\mathbb{M}(x)$ must correspond to a subgraph of the largest bipartite graph in which all edges are shorter than $x$. Such graphs are sometimes called *bottleneck graphs*. Thus, to check whether $\mathbb{M}(x)$ is empty, we can instead check whether there exists a $\mu$ in $\mathbb{M}$ as a subgraph of the corresponding bottleneck graph. This, in turn, is the same as determining whether a $\mathcal{T}$-perfect matching exists in the bottleneck graph, which is a well-studied decision problem. Once we find a suitable decision algorithm, the search for the smallest admissible $x$, which the authors denote $\varkappa$, is straightforward. Algorithms of this type are called *threshold algorithms*.

When viewed from this perspective, the authors consider the bottleneck matching problem when $\delta$ induces a convex graph. They describe a threshold algorithm using Glover's algorithm for the decision subproblem, and they use binary search over the number line to find $\varkappa$. This focus overlaps with questions of long-standing interest to the computer science community, and it appears the authors have rediscovered part of that literature. To the best of my knowledge, the bottleneck matching problem was first studied by Fulkerson, Glicksberg and Gross (1953), and the first threshold algorithm for the problem was introduced by Garfinkel (1971). The first discussion of a threshold algorithm based on Glover's algorithm appears to be Rinnooy Kan (1976), page 66.

### 1.2 Possible Improvements and Extensions

A few improvements and extensions present themselves when we better understand how the procedure relates to previous work. The first improvement is an alternative method to search for $\varkappa$. The authors suggest binary search. Binary search on the number line may, however, take a long time to terminate. It will fail to terminate altogether if decimal precision is unlimited. The solution the authors offer is to find an upper bound for $\varkappa$ based on a tolerance parameter. While this will suffice in many situations, there are cases in which the method performs poorly. In particular, the tolerance parameter ensures that the upper bound is close to $\varkappa$, but it does not ensure that the number of edges between the bound and $\varkappa$ is small.

The alternative method, commonly used for threshold algorithms, is to search over the edges. We know that $\varkappa$ must be equal to $\delta(t, c)$ for some $(t, c)$ in $\mathcal{T} \times \mathcal{C}$ when we use a bottleneck objective. If we find this edge, we find $\varkappa$. A straightforward procedure is to sort the edges by cost and run binary search over the indices of the sorted edges. The sorting is the most time-consuming step here, completed in $\mathcal{O}(N^2 \log N)$ time. If this is deemed too slow, convexity allows for improvements. For example, one may do binary search among the edges separately for each treated unit, thereby bypassing the need to calculate and sort all edges at once. The time complexity is still quasi-quadratic in the worst case, but the average runtime is greatly reduced.

Turning now to extensions, the obvious focus is to relax the need for convexity. The authors use the absolute difference in propensity scores as their cost function. This induces a convex graph and ensures that Glover's algorithm can be used. The procedure cannot be used when practitioners want to prune edges for other cost functions. Fortunately, several algorithms exist for the bottleneck matching problem. To the best of my knowledge, the one with the lowest time complexity for general edge costs is by Punnen and Nair (1994). Runtime grows as $\mathcal{O}(N^{2.5})$ here. While this is a considerably higher complexity than for convex graphs, it is still lower than the cubic rate of the network flow problem that follows, so it may be acceptable.

Algorithms for general costs are rarely required in practice, however, because the edge costs are often distances in metric spaces with some additional structure. Efrat, Itai and Katz (2001) introduce a set of algorithms for the bottleneck problem when the costs are distances in various normed vector spaces. The time complexity is between $\mathcal{O}(N^{1.5})$ and $\mathcal{O}(N^2)$, depending on the dimensionality of the space and the choice of norm. For example, runtime grows as $\mathcal{O}(N^{1.5} \log N)$ for Euclidean distances in two-dimensional space.

## 2. OPTIMALITY

### 2.1 Optimal in What Sense?

The authors open their paper by stating that they propose new optimal matching techniques, and they refer to $\varkappa$ as the optimal caliper. There are several senses in which a matching procedure can be optimal. The authors do not state which one they have in mind.

The ultimate objective of matching in causal inference is to reduce bias from confounded treatment assignment. There might be secondary objectives, such as precision in estimation. It is, however, rare that theoretical investigations of matching methods focus on these statistical objectives. Instead, such investigations tend to focus on match quality. A matching method is then said to be optimal if it minimizes the surrogate objective function used in the matching problem (such as the sum of edge costs in the previous section). Matchings that are optimal in this sense need not be optimal in a statistical sense. Statisticians and

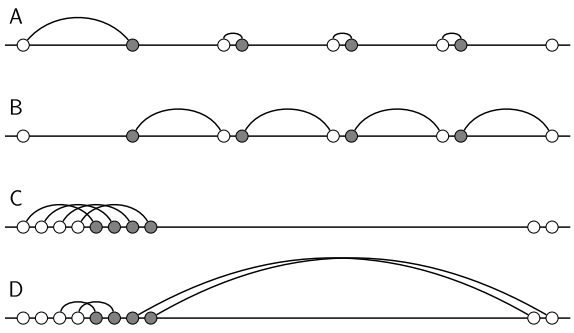


FIG. 1. *Best matchings before* (*A*, *C*) *and after* (*B*, *D*) *bottleneck preprocessing. Filled nodes represent treated units.*

methodologists are aware that the two optimality concepts do not coincide, but practitioners do not appear to always appreciate the difference. It may therefore be helpful to note that the procedure the authors present is not optimal in a statistical sense.

What might be puzzling also to statisticians is that the authors depart from the conventional definition of optimality. The matching produced by the network flow algorithm with $\varkappa$ as a caliper may not be optimal with respect to the surrogate objective function. That is, the matching need not be in $\mathbb{M}^*$. The authors allude to this, but the departure from the current literature is not stated explicitly, nor is it made clear that match quality can be considerably worse after preprocessing. Figure 1 provides two examples. Panels A and B illustrate the consequence of bottleneck preprocessing when the caliper is imposed on the propensity score, as described above. Panels C and D do the same when the caliper is on the ranks of the propensity score, as it is in the authors' application.

Further departures from optimality in match quality is caused by the authors' use of different cost functions in the preprocessing step and the main matching step. They use absolute differences in propensity scores for the bottleneck problem, which ensures convexity, and Mahalanobis distances in the underlying covariate space for the network flow problem. This disconnects the two steps, and the procedure as a whole would not be optimal even if each step maintained optimality with respect to its cost function. The approach is motivated by a heuristic introduced by Rosenbaum and Rubin (1985). This may be sound but, by virtue of being a heuristic, is not optimal.

The authors appear to instead have computational concerns in mind when they discuss optimality. The caliper provided by the authors' algorithm produces the sparsest network flow problem among all admissible calipers. This means that $\varkappa$ is the caliper with the largest reduction in runtime while ensuring that the resulting matching is in $\mathbb{M}$. This is the sense in which it is optimal. I do not necessarily object to this use of the term, but casual readers may not notice that it differs from how the term is used in most of the current matching literature.

### 2.2 How Optimal Is the Procedure?

To investigate how effectively the authors' procedure reduces runtime, we need to know how quickly $\varkappa$ shrinks. The only study I could find of the question was by Pferschy (1996). He shows that $\varkappa$ converges to zero at a fast rate when the edge costs are independent and uniformly distributed on the unit interval. This seems encouraging, but the edge costs are not uniformly distributed when based on propensity scores, and Pferschy's result does not apply. The concern is that control units are rare relative to treated units for larger propensity scores, so a large $\varkappa$ may be necessary to ensure that all treated units have matches. The authors' discussion about how outliers can make $\varkappa$ large is a related concern.

The authors offer two alternative approaches to ensure that sufficiently many edges are pruned even when $\varkappa$ converges slowly. The first approach is to include only edges between each treated unit and a fixed number of its nearest neighbors. The second approach has already been mentioned, namely to impose a caliper on the ranks of the propensity score rather than the score itself. The nearest neighbors approach appears to prune edges more efficiently in the authors' application, and it is used in their complexity proof, so it will be the focus here.

The approach dictates that an edge between a treated unit and a control is included in the network flow problem only when the control is among the $x$ nearest neighbors of the treated unit. The authors use a threshold algorithm to find the smallest $x$ that produces an admissible matching. The minimum, denoted by $\nu$, is used in Proposition 6 to investigate the overall time complexity. The authors first show that $E = \mathcal{O}(\nu N)$. They then assume that $\nu = \mathcal{O}(\log N)$, to conclude that number of edges grows at a quasi-linear rate. If true, this would indeed demonstrate that the procedure is optimal in a computational sense. However, assuming $\nu = \mathcal{O}(\log N)$ appears to be begging the question.

We need to better understand $\nu$ to know whether the assumption is reasonable. Characterizing $\nu$ directly is difficult, so a lower bound is the focus. Let $\mathcal{T}_\mathrm{R} = \{t \in \mathcal{T} : \rho(t) \geq 0.5\}$ collect all treated units with a propensity score greater than 0.5, and let $\mathcal{C}_\mathrm{R}$ be defined analogously for controls. If $|\mathcal{T}_\mathrm{R}| > |\mathcal{C}_\mathrm{R}|$, then at least $D = |\mathcal{T}_\mathrm{R}| - |\mathcal{C}_\mathrm{R}|$ units in $\mathcal{T}_\mathrm{R}$ must be matched with controls in $\mathcal{C} \setminus \mathcal{C}_\mathrm{R}$. It follows that $\nu$ is bounded from below by $D$.

The next step is to characterize $D$. Consider when units are sampled independently and identically distributed according to some density $f(z)$ over the propensity score $z \in [0, 1]$. By construction, the probability that a unit with propensity score $z$ is treated is $z$, or equivalently, $\Pr(i \in \mathcal{T} \mid \rho(i) = z) = z$. It follows that

$$\mathrm{E}[D] = N \int_{0.5}^{1} z f(z)\,\mathrm{d}z - N \int_{0.5}^{1} (1 - z) f(z)\,\mathrm{d}z.$$

This implies that $\mathrm{E}[D] = \Omega(N)$ whenever $\Pr(\rho(i) > 0.5)$ is positive, and $\nu$ will then grow proportionally to $N$. For example, if $f(z) = 32(1-z)/15$ for $z \in [0, 0.75]$ and zero otherwise, then $\mathrm{E}[D]$ is about $N/25$.

The conclusion is that the authors' procedure will generally not reduce the time complexity. It will reduce runtime because it will always prune some edges, but the runtime will grow at the same rate as without preprocessing. The issue is that edges are pruned using global constraints, either by $\varkappa$ or by $\nu$, so the same amount of pruning will be done in regions where controls are plenty as where they are scarce.

### 2.3 An Alternative Procedure

The authors' goal is computational optimality. Some practitioners may also wish to retain optimality in match quality. An alternative preprocessing procedure is then required. Glover's algorithm provides some helpful hints toward that end.

Glover used convexity to construct matches in a greedy fashion. He showed that the procedure yields a matching of maximum cardinality, but the matching is not necessarily of minimum cost. The reason is that a treated unit matched to a control may force a second treated unit to be matched to a more distant control because the first control is already matched. An optimal matching procedure must consider such chain effects, and Glover's greedy construction does not. These chain effects rarely extend throughout the sample, however, and one can greedily prune edges while retaining optimality.

Where Glover used convexity, the procedure described here similarly takes advantage of the underlying geometry of the matching problem when the costs are based on propensity scores. In particular, when the costs are absolute differences in a one-dimensional score, at least one $\mu$ exists in $\mathbb{M}^*$ without crossing matches. Two matches are said to be crossing if $\mu(t) > \mu(s)$ for $t, s \in \mathcal{T}$ such that $t < s$, where the unit indices have been sorted by the score. If a crossing exists, the sum of edge costs can be made weakly smaller by matching $t$ with $\mu(s)$ and $s$ with $\mu(t)$. I will not prove this formally, but Figure 2 may provide some intuition.

The observation that an optimal solution exists without crossings could prove useful more generally, and it may even allow us to find a minimum-cost solution directly, but it will here be used to prune edges.

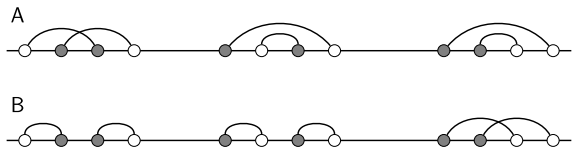


FIG. 2. *Matchings with* (*A*) *and without* (*B*) *crossings.*

Consider the following algorithm. Sort the units by their scores and assign indices according to their ranks. Initialize a queue with capacity $T$, and initialize $(a_i, b_i) = (i, i)$ for each $i \in \mathcal{C}$. Traverse the units from left to right, and do the following for each unit $i$:

- If $i \in \mathcal{T}$, enqueue $i$.
- If $i \in \mathcal{C}$, dequeue the front index, $j$, and set $a_i = j$. Do nothing if the queue is empty.

Clear the queue and repeat the process, but substitute $b_i = j$ for $a_i = j$ and traverse the units from right to left this time. Once finished, find the minimum-cost network flow using $E = \{(t, c) \in \mathcal{T} \times \mathcal{C} : a_c \le t \le b_c\}$ as the edge set.

This alternative approach has a few attractive properties. First, it maintains optimality with respect to the sum of edge costs. That is, a minimum-cost matching with the reduced edge set is a minimum-cost matching also with the full edge set, so it is in $\mathbb{M}^*$. The reduced edge set may, however, not retain optimality with respect to more intricate network flow problems.

Second, disregarding the time spent on initialization, the procedure traverses the units twice with a constant time requirement for each unit. This is approximately the same runtime as two calls to Glover's algorithm. The overall complexity, including the sorting during initialization, is quasi-linear.

Third, the pruning is local. That is, the pruning will be more aggressive in regions where controls are plenty and more conservative in regions where they are scarce. This will remove more edges than the corresponding global restrictions. In the worst case, the number of retained edges is still $\mathcal{O}(N^2)$, but this tends to happen when overlap (in the causal inference sense) is poor, and we might be occupied by statistical concerns in such situations.

## 3. CONCLUDING REMARKS

Readers should remember that covariate adjustments using matching largely is an exercise in heuristics. Theoretical discussions of matching methods, including my comments here, have yet to prove themselves particularly informative. Instead, experience gained from either applications or simulation studies provide the most helpful insights. The usefulness of the preprocessing procedure the authors describe becomes clear when seen in this more pragmatic light. Samples containing several million observations are rare, but samples with a few hundred thousand observations are common. The authors' procedure and their R package make complex network flow matching possible in these moderately large samples. This makes it an important and welcome contribution.

Before practitioners start pruning edges, however, there are a few things they may wish to consider. First, the purpose of the caliper in the authors' procedure differs from its typical use. Calipers are conventionally used to prune

treated units that lack matches of sufficiently high quality. Here, the caliper is used to prune edges. In fact, the caliper is chosen under the restriction that it cannot exclude any treated units. If a caliper is used for its conventional purpose, there is no reason to separately prune edges in this way because a conventional caliper will be smaller than $\varkappa$. Edge pruning based on nearest neighbors may, however, still be useful.

Second, practitioners do not always need complex matching procedures for their investigations. I have encountered researchers desperately trying to find a matching method that will work when the number of observations is in the millions, only to discover that they want to match on a handful of binary covariates. The best approach in such cases is exact matching or stratification. Even stratification on coarsened continuous variables might be reasonable because large samples allow the coarsening to be quite detailed. The authors mention that fine-balancing constraints might be difficult to impose in such cases. The strata can, however, be added to one large network flow problem after stratification to attend to such concerns. This is essentially what the authors do in their application when they match exactly on the 463 surgical procedures. These more straightforward approaches are not perfect, but they do attend to most of the covariate imbalances while retaining the simplicity that often prompts practitioners to choose matching in the first place.

Finally, readers may find interest in some recent work on matching in large samples not cited by the authors. Bennett, Vielma and Zubizarreta (2018) show that the integer programming problem underlying the matching problem can be relaxed to a more tractable linear program. They also construct a reference group, or template, to which the units are matched. The reference group can be made considerably smaller than the overall sample, thereby reducing runtime. When using the two techniques in conjunction, Bennett et al. demonstrate that samples with more than 700,000 units can be matched using fine balancing objectives within minutes.

My own work together with Michael Higgins and Jasjeet Sekhon (2017) may also be of interest. We explore the idea that much of the information needed to solve fairly complex matching problems can be encoded in sparse, unweighted graphs. In particular, such graphs can be constructed to encode both information about the similarity between units and various matching constraints. Finding matchings in a sparse graph is considerably simpler than the original problem. We leverage this insight to construct an approximation algorithm for a generalized version of full matching which terminates in $\mathcal{O}(N \log N)$ time whenever the underlying metric space allows for nearest neighbor search in $\mathcal{O}(\log N)$ time. A simulation study demonstrates that an implementation of the algorithm in R can match 100 million observations within 15 minutes.

## ACKNOWLEDGMENTS

I thank Winston Lin for helpful comments.

## REFERENCES

BENNETT, M., VIELMA, J. P. and ZUBIZARRETA, J. R. (2018). Building representative matched samples with multi-valued treatments in large observational studies: Analysis of the impact of an earthquake on educational attainment. Available at arXiv:1810.06707.

EFRAT, A., ITAI, A. and KATZ, M. J. (2001). Geometry helps in bottleneck matching and related problems. *Algorithmica* **31** 1–28. MR1840188 https://doi.org/10.1007/s00453-001-0016-8

FULKERSON, R., GLICKSBERG, I. and GROSS, O. (1953). A production line assignment problem Technical Report No. RM-1102 The Rand Corporation Santa Monica.

GARFINKEL, R. S. (1971). An improved algorithm for the bottleneck assignment problem. *Oper. Res*. **19** 1747–1751.

PFERSCHY, U. (1996). The random linear bottleneck assignment problem. *RAIRO Oper. Res*. **30** 127–142. MR1424230 https://doi.org/10.1051/ro/1996300201271

PUNNEN, A. P. and NAIR, K. P. K. (1994). Improved complexity bound for the maximum cardinality bottleneck bipartite matching problem. *Discrete Appl. Math*. **55** 91–93. MR1298517 https://doi.org/10.1016/0166-218X(94)90039-6

RINNOOY KAN, A. H. G. (1976). *Machine Scheduling Problems*: *Classification*, *Complexity and Computations*. Martinus Nijhoff, The Hague.

ROSENBAUM, P. R. and RUBIN, D. B. (1985). Constructing a control group using multivariate matched sampling methods that incorporate the propensity score. *Amer. Statist*. **39** 33–38.

SÄVJE, F., HIGGINS, M. J. and SEKHON, J. S. (2017). Generalized full matching. Available at arXiv:1703.03882.

YU, R., SILBER, J. H. and ROSENBAUM, P. R. (2020). Matching methods for observational studies derived from large administrative databases. *Statist. Sci*. **35** 338–355.