Improving massive experiments with threshold blocking

Michael J. Higgins[a], Fredrik Sävje[b], and Jasjeet S. Sekhon[c,d,1]

[a]Department of Statistics, Kansas State University, Manhattan, KS 66506; [b]Department of Economics, Uppsala University, SE-751 85 Uppsala, Sweden; [c]Department of Political Science, University of California, Berkeley, CA 94720; and [d]Department of Statistics, University of California, Berkeley, CA 94720

Edited by Richard M. Shiffrin, Indiana University, Bloomington, IN, and approved December 9, 2015 (received for review June 15, 2015)

Inferences from randomized experiments can be improved by blocking: assigning treatment in fixed proportions within groups of similar units. However, the use of the method is limited by the difficulty in deriving these groups. Current blocking methods are restricted to special cases or run in exponential time; are not sensitive to clustering of data points; and are often heuristic, providing an unsatisfactory solution in many common instances. We present an algorithm that implements a widely applicable class of blocking—threshold blocking—that solves these problems. Given a minimum required group size and a distance metric, we study the blocking problem of minimizing the maximum distance between any two units within the same group. We prove this is a nondeterministic polynomial-time hard problem and derive an approximation algorithm that yields a blocking where the maximum distance is guaranteed to be, at most, four times the optimal value. This algorithm runs in *O*(*n* log *n*) time with *O*(*n*) space complexity. This makes it, to our knowledge, the first blocking method with an ensured level of performance that works in massive experiments. Whereas many commonly used algorithms form pairs of units, our algorithm constructs the groups flexibly for any chosen minimum size. This facilitates complex experiments with several treatment arms and clustered data. A simulation study demonstrates the efficiency and efficacy of the algorithm; tens of millions of units can be blocked using a desktop computer in a few minutes.

experimental design | blocking | big data | causal inference

Properly executed experiments with random assignment guarantee that estimated treatment effects are equal to the true causal effects of interest in expectation. However, only one assignment is realized for a particular experiment, and there could be chance differences between treatment and control groups that muddle any comparison. Indicative of such differences are imbalances in observed baseline characteristics between the treatment groups. For example, in a medical study on the effect that a drug has on life expectancy, it may occur by chance that the control group is older and sicker than the treatment group. Whenever imbalances in prognostically important covariates are observed, there is reason to suspect that the resulting estimates are inaccurate. Studies that do not attended to this issue cannot be considered to follow the gold standard of randomized experiments (1): Viewed before assignment, investigators allowed for unnecessarily high variance; viewed after assignment, they allowed the estimator to be biased conditional on the observed distribution of covariates.

Since R. A. Fisher's canonical treatment (2), "blocking" has been the default experimental design to deal with this problem. With this design, the investigator forms groups of units, or "blocks," that are as similar as possible. Treatments are then randomly assigned in fixed proportions within blocks and independently across them. This prevents imbalances in observed covariates, which can increase precision if these covariates are predictive of outcomes.

Unadjusted estimates for even massive experiments are often too variable to enable reliable inferences because the effects of interest may be small and distributional issues may result in surprisingly large variances. A prominent case is A/B testing of the effectiveness of online advertising (3). The effects of the advertisements are generally very small (although economically relevant due to the low costs), and consumers' behaviors tend to follow distributions with fat tails (4).

Moreover, with the rise of massive data, researchers, policy makers, and industry leaders have become increasingly interested in making fine-grained inferences and targeting treatments to subgroups (5). The recent focus on personalized and precision medicine is a noteworthy example (6). Even with large experiments, subgroups of interest often lack data because of the vagaries of random assignment and the curse of dimensionality. Blocking enables researchers to define the subgroups of interest *ex ante*. This ensures that there will be sufficient data to make fine-grained inferences.

Finally, because blocking adjusts for covariates in the design of the study, it limits both the need for and the effect of adjusting the experiment *ex post*. Such adjustments often lead to incorrect test levels if investigators specify models based on the observed treatment assignments (7), or if they pick models based on test results—a habit that appears to be prevalent (8).

In short, blocking is an essential tool for experiential design. It enables one to follow Fisher's advice that Nature should be asked one question at a time, which is the central motivation for random assignment in the first place (9).

Despite its providence, usefulness, and wide applicability, there are many situations where an effective blocking design is desirable but where none is possible or feasible. In particular, current blocking algorithms have primarily focused on the special case where blocks with exactly two units are desired, the so-called "matched-pair design" (10). There exist optimal, polynomial time algorithms for this design, such as nonbipartite matching (11), but they are limited to experiments with only two treatment conditions. Although there exist heuristic algorithms that can facilitate larger block sizes, their theoretical properties are unknown, and their performance has not been fully evaluated (12). In many cases, even with relatively modest samples and considerable computational power, several years would be required to obtain results using algorithms with a proven level of optimality. For the largest of experiments, existing algorithms are too computationally demanding even for the matched-pair design.

In this paper, we introduce the first algorithm, to our knowledge, that produces guaranteed near-optimal blockings for any desired block size. Specifically, we consider the blocking problem

This paper results from the Arthur M. Sackler Colloquium of the National Academy of Sciences, "Drawing Causal Inference from Big Data," held March 26–27, 2015, at the National Academies of Sciences in Washington, DC. The complete program and video recordings of most presentations are available on the NAS website at www.nasonline.org/Big-data.

Author contributions: M.J.H., F.S., and J.S.S. designed research, performed research, contributed new reagents/analytic tools, analyzed data, and wrote the paper.

The authors declare no conflict of interest.

This article is a PNAS Direct Submission.

[1]To whom correspondence should be addressed. Email: sekhon@berkeley.edu.

This article contains supporting information online at www.pnas.org/lookup/suppl/doi:10.1073/pnas.1510504113/-/DCSupplemental.

where one wants to minimize the greatest within-block dissimilarity, as measured by an arbitrary distance metric, subject to a minimum required block size. We prove that this problem is nondeterministic polynomial-time hard (NP-hard), and we provide an approximation algorithm that, in the worst case, produces a solution that is four times greater than the optimum. The algorithm uses computational resources very efficiently: It is guaranteed to terminate in linearithmic time with linear space complexity. This makes it applicable in many cases where existing algorithms are impractical, including experiments with large samples or multiarmed treatment schemes.

In additional to large data, our approximation algorithm is likely to perform well in traditional, smaller experiments, not the least when designs other than matched pairs are desired. Our formulation of the blocking problem, threshold blocking, differs from the past literature in that it allows for some flexibility in the block structure. This leads to blockings that respect natural clusters of units that may improve performance.

Blocking as a Graph Partition Problem

A blocking of an experiment's sample is a partition of its units into disjoint sets, referred to as blocks. The blocking problem is to find a blocking where units assigned to the same block are as similar as possible—either to minimize differences on prognostically important covariates or to facilitate the study of subgroups of interest. In the former case, when treatments are assigned in fixed proportions within blocks, blocking reduces imbalances between treatment groups and improves the precision of estimated effects.

Blocking problems can be viewed as graph partitioning problems (11, 13). Each experiment yields a weighted graph where vertices represent units in the sample. Edges connect each pair of units, and edge costs are measured dissimilarity between corresponding units (e.g., the Euclidean or Mahalanobis distance between their covariate vectors). Minimizing within-block edge costs when this graph is partitioned subject to a cardinality condition is equivalent to deriving an optimal blocking. In the matched-pair design, the objective is to minimize the sum of all within-block edge costs subject to that each block contains exactly two vertices.

To improve blockings and facilitate the approximation algorithm, we consider a formulation of the blocking problem that differs from the past literature in three aspects. First, we facilitate designs other than matched pairs by allowing for any desired block size. Second, we consider blockings where each block is required to contain at least the desired number of units. Such threshold blockings have several advantages compared with the "fixed-sized" blockings derived by previous methods, where blocks are forced to be exactly of the desired size. Every fixed-sized blocking is also a threshold blocking; hence, for any sample and objective function, the optimal solution for the latter case is guaranteed to be at least as good as in the former (14). In particular, fixed-sized blocks might not respect natural clusterings of units, and one is sometimes forced to assign similar units to different blocks just to satisfy the cardinality condition.

Third, we consider a "bottleneck" objective function. That is, we wish to find a blocking that minimizes the maximum within-block edge cost—making the two least similar units assigned to the same block as similar as possible. The bottleneck objective has some advantages over the commonly used sum (or average) objective. Going back to at least Cochran, statisticians have observed that a few large imbalances are often more problematic than many small ones, especially when blocking is combined with *ex post* adjustments (15). Furthermore, parallel to monotonic imbalance bounding in observational studies (16), controlling the maximum imbalance within a block guarantees that the average imbalance cannot exceed this maximum after treatments are assigned. If an infinity norm is used to measure dissimilarity (i.e., the Chebyshev distance), this also applies to each covariate in isolation. Minimizing sums or averages does not provide such guarantees. Finally, bottleneck optimization problems often have approximate solutions that can be found efficiently (17). Although the algorithm cannot readily be extended to other objective functions, it has a local optimality property that provides good performance with respect to the average within-block edge cost.

The Bottleneck Threshold Blocking Problem

Let k denote a threshold for the minimum block size. Consider the complete graph $G=(V,E)$ describing an experimental sample, where V denotes the set of n vertices (the experimental units) and E denotes the set of edges connecting all pairs of vertices. (Refer to *Graph Theoretical Definitions* for graph theoretical terminology and notation used in this paper.) For each $ij\in E$, there is an associated cost, c_{ij}, indicating the dissimilarity between i and j; lower costs mean that units are more similar. We require that these costs satisfy the triangle inequality

$$\forall ij, j\ell, i\ell \in E, c_{ij}+c_{j\ell}\geq c_{i\ell}. \qquad [1]$$

This ensures that the direct route between two vertices is no longer than a detour through a third vertex. All distance metrics fulfill this criterion by definition.

Definition 1: A threshold blocking with threshold k is a partition $\mathbf{b}=\{V_1,\cdots,V_m\}$ of V where each block satisfies the size threshold

$$\forall V_x\in\mathbf{b}, |V_x|\geq k. \qquad [2]$$

Definition 2: The subgraph generated by a blocking $\mathbf{b}=\{V_1,\ldots,V_m\}$, denoted $G(\mathbf{b})=[V,E(\mathbf{b})]$, is the union of subgraphs of G induced by the components of $\mathbf{b}$; that is, an edge $ij\in E(\mathbf{b})$ only if i and j are in the same block,

$$E(\mathbf{b})\equiv\{ij\in E:\exists V_x\in\mathbf{b}, i,j\in V_x\}. \qquad [3]$$

Let $\mathbf{B}_k$ denote the set of all possible threshold blockings of G with a threshold of k. The bottleneck threshold blocking problem is to find a blocking in $\mathbf{B}_k$ such that the maximum within-block dissimilarity is minimized. This amounts to finding an optimal blocking $\mathbf{b}^*\in\mathbf{B}_k$ such that the largest edge cost in $G(\mathbf{b}^*)$, is as small as possible; let λ denote this minimum,

$$\max_{ij\in E(\mathbf{b}^*)} c_{ij}=\min_{\mathbf{b}\in\mathbf{B}_k}\max_{ij\in E(\mathbf{b})} c_{ij}\equiv\lambda. \qquad [4]$$

Definition 3: An α-approximation algorithm for the bottleneck threshold blocking problem derives a blocking $\mathbf{b}\in\mathbf{B}_k$ with a maximum within-block cost no larger than $\alpha\lambda$,

$$\max_{ij\in E(\mathbf{b})} c_{ij}\leq\alpha\lambda. \qquad [5]$$

In *Proof of NP-Hardness*, we show that, unless $\mathbf{P}=\mathbf{NP}$, no polynomial-time $(2-\epsilon)$-approximation algorithm exists for any $\epsilon>0$. Therefore, the problem is NP-hard, and finding an optimal solution is computationally intractable except for special cases or very small samples.

An Approximately Optimal Blocking Algorithm

We present a 4-approximation algorithm for the threshold blocking problem. Outside of an initial construction of a nearest neighbors graph, this algorithm has $O(kn)$ time and space complexity. Hence, it can be used in experiments with millions of units. Although the algorithm guarantees a threshold blocking with maximum within-block cost no larger than 4λ, simulations indicate that derived blockings are much closer to the optimum in practice.

The Algorithm. Given the graph representation of the experimental sample, $G=(V,E)$, and a prespecified threshold k, the approximate blocking algorithm proceeds as follows:

1. Construct a $(k-1)$-nearest neighbor subgraph of G. Denote this graph $G_{nn}=(V,E_{nn})$.
2. Find a maximal independent set of vertices, $\mathbf{S}$, in the second power of the $(k-1)$-nearest neighbor subgraph, G^2_{nn}. Vertices in $\mathbf{S}$ are referred to as the block seeds.
3. For each seed $i\in\mathbf{S}$, create a block comprising its closed neighborhood in G_{nn}, $V_i=\mathrm{N}_{G_{nn}}[i]$.
4. For each yet unassigned vertex, assign it to any block that contains one of its adjacent vertices in G_{nn}.

When the algorithm terminates, the collection of blocks, $\mathbf{b}_{alg}=\{V_i\}_{i\in\mathbf{S}}$, is a valid threshold blocking of the experimental units that satisfies the optimality bound.

Informally, the algorithm constructs the blocking by selecting suitable vertices, the seeds, from which the blocks are grown. Seeds are spaced sufficiently far apart so as not to interfere with each other's growth, but they are dense enough so that all nonseed vertices have a seed nearby. Specifically, the second step of the algorithm prevents any vertex from being adjacent to two distinct seeds in the $(k-1)$-nearest neighbor subgraph, but also never more than a walk of two edges away from a seed. This ensures that the seeds' closed neighborhoods do not overlap, and that vertices assigned to the same block are at a close geodesic distance. Fig. 1 illustrates how the algorithm constructs blocks in an example sample.

Validity and Complexity. We first prove that the algorithm is guaranteed to produce a valid threshold blocking, and then examine its time and space complexity.

Lemma 1. *For any nonseed vertex, $i\notin\mathbf{S}$:*

1. *There exist no two seeds both adjacent to i in G_{nn}.*
2. *There exists a walk in G_{nn} of two or fewer edges from i to the seed of the block that i is assigned to.*

Proof: The lemma follows from $\mathbf{S}$ being a maximal independent set in G^2_{nn}. Refer to *Proof of Lemma 1* for a complete proof.

Theorem 1. (Validity) *The blocking algorithm produces a threshold blocking: $\mathbf{b}_{alg}\in\mathbf{B}_k$.*

Proof: By Lemma 1, each vertex assigned in the third step is adjacent to exactly one seed; thus it will be in exactly one block. In the fourth step, vertices are assigned to exactly one block each. This ensures that the blocks are disjoint and span V; thus $\mathbf{b}_{alg}$ is a partition of V.

All seeds have at least $k-1$ adjacent vertices in G_{nn}. In the third step, these vertices and the seeds themselves will form the blocks, ensuring that each block contains at least k vertices. This satisfies Definition 1.

Theorem 2. (Complexity) *The blocking algorithm terminates in polynomial time using $O(kn)$ space.*

Proof: Naively, the $(k-1)$-nearest neighbor subgraph can be constructed by sorting each vertex's edge costs and finding its $k-1$ nearest neighbors. Thus, G_{nn} is constructed in, at most, $O(n^2\log n)$ time (18). To enable constant time access to the neighbors of any vertex, store the nearest neighbor subgraph in n lists containing each vertex's edges. There can be, at most, $(k-1)n$ edges in G_{nn}. This implies an $O(kn)$ space complexity for the edge lists.

Using the edge lists, a maximal independent set in the second power of G_{nn} can be found in $O(kn)$ time without changing the space complexity. See *Subroutine for the Second Step of the Algorithm* for additional details. The third step is completed within $O(n)$ time as Lemma 1 ensures that, at most, n units will be assigned to blocks in this step and the edge lists enable constant time access to the seeds' neighbors. In the fourth step, it will never be necessary to search through all edge lists more than once, implying a complexity of $O(kn)$.

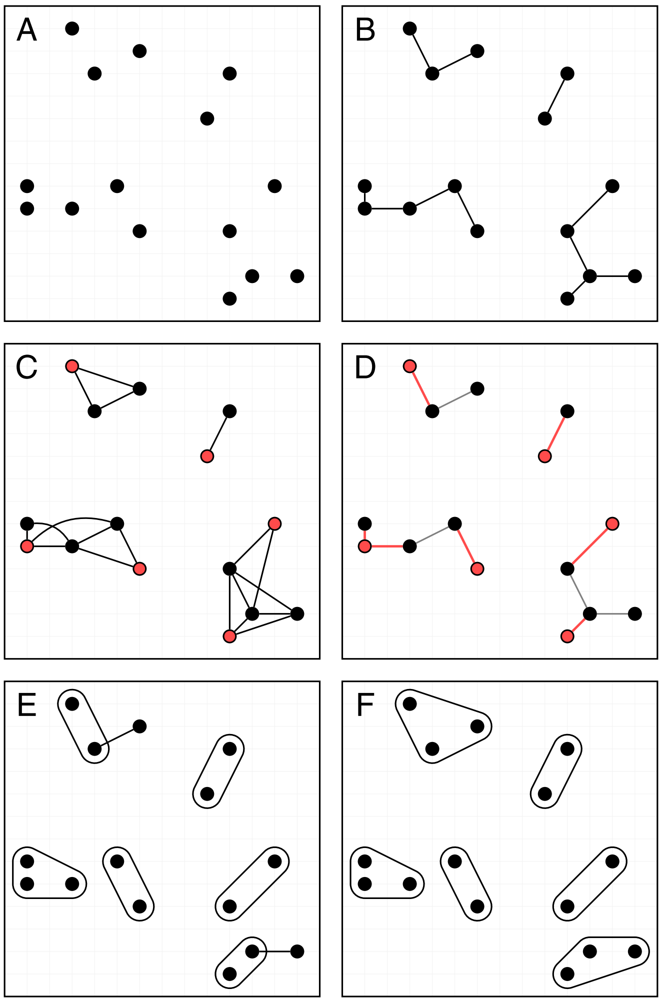

Fig. 1. An illustration of the approximation algorithm for a sample with 2D covariate data when a minimum block size of two is desired ($k=2$). (*A*) The algorithm is provided with a set of data points and forms the graph by drawing an edge between all possible pairs of units. The edges are here omitted to ease presentation. (*B*) A $(k-1)$-nearest neighbor subgraph is constructed. (*C*) The second power of the nearest neighbor subgraph is derived, as shown by the edges, and a maximal independent set is found, as shown by the red vertices (the seeds). (*D*) All vertices adjacent to a seed in the nearest neighbor subgraph are included in the blocks formed by the seeds, as shown by the edges marked in red. (*E*) The two yet unassigned vertices are assigned to the blocks that contain one of their adjacent vertices in the nearest neighbor subgraph. (*F*) The final blocking.

Remark 1: After the initial construction of the $(k-1)$-nearest neighbor subgraph, the algorithm terminates in $O(kn)$ time. As the nearest neighbor search problem is well studied, the naive subroutine in the proof can be improved on in most applications. In particular, most experiments will have reasonably low-dimensional metric spaces. Using specialized algorithms, the subgraph can, in that case, be constructed in $O(kn\log n)$ expected time (19) or worst-case time (20). If the covariates are not few to begin with, it is often advisable to use some dimensionality reduction technique before blocking so as to extract the most relevant information.

Run time can also be improved by using an approximate nearest neighbor search algorithm. However, approximate optimality is not guaranteed in that case.

Remark 2: It is rarely motivated to increase the block size as the sample grows; thus k can be considered fixed in the typical experiment. When k is fixed and one can use a specialized procedure to derive the nearest neighbor subgraph, the algorithm has $O(n \log n)$ time and $O(n)$ space complexity.

Approximate Optimality. To prove the optimality bound, we will first show that the edge costs in the $(k-1)$-nearest neighbor subgraph are bounded. As the algorithm ensures that vertices in the same block are at a close geodesic distance in that subgraph, approximate optimality follows from the triangle inequality.

Lemma 2. *No edge cost in G_{nn} can be greater than the maximum cost in the optimal blocking,*

$$\forall ij \in E_{nn}, c_{ij} \le \lambda. \qquad \textbf{[6]}$$

Proof: Consider the graph

$$G_\lambda = \left(V, E_\lambda = \{ij \in E : c_{ij} \le \lambda\}\right). \qquad \textbf{[7]}$$

For all edges in an optimal blocking, $ij \in E(\mathbf{b}^*)$, we have $c_{ij} \le \lambda$ from optimality. It follows that $E(\mathbf{b}^*) \subseteq E_\lambda$.

Let $c_+ = \max\{c_{ij} : ij \in E_{nn}\}$ and consider

$$G_+ = \left(V, E_+ = \{ij \in E : c_{ij} < c_+\}\right). \qquad \textbf{[8]}$$

The minimum degree of this graph, $\delta(G_+)$, must be less than $k-1$. If not, a $(k-1)$-nearest neighbor graph exists as a subgraph of G_+. As this new graph does not contain c_+, it is contradictory that c_+ is the maximum edge cost in G_{nn}.

Suppose that $c_+ > \lambda$. It then follows that $E_\lambda \subseteq E_+$; thus

$$\delta[G(\mathbf{b}^*)] \le \delta(G_\lambda) \le \delta(G_+) < k-1. \qquad \textbf{[9]}$$

That is, there exists a vertex in $G(\mathbf{b}^*)$ with fewer than $k-1$ edges. It follows that there must exist a block in $G(\mathbf{b}^*)$ with fewer than k vertices and, as Definition 1 then is violated, it cannot be a valid blocking. The contradiction proves that $c_+ \le \lambda$ which bounds all edges in E_{nn}.

Theorem 3. (Approximate optimality) *The blocking algorithm is a 4-approximation algorithm,*

$$\max_{ij \in E(\mathbf{b}_{alg})} c_{ij} \le 4\lambda. \qquad \textbf{[10]}$$

Proof: Let $\mathbf{b}_{alg}$ denote the blocking produced by the algorithm. Consider any within-block edge $ij \in E(\mathbf{b}_{alg})$. We must show that c_{ij} is bounded by 4λ.

If $ij \in E_{nn}$, we have $c_{ij} \le \lambda$ by Lemma 2. If $ij \notin E_{nn}$ and $i \notin \mathbf{S}, j \in \mathbf{S}$, then, by Lemma 1, there exists some ℓ so that $i\ell, \ell j \in E_{nn}$. Lemma 2 applies to both these edges. By Eq. **1**, the triangle inequality, it follows,

$$c_{ij} \le c_{i\ell} + c_{\ell j} \le \lambda + \lambda = 2\lambda. \qquad \textbf{[11]}$$

If $ij \notin E_{nn}$ and $i, j \notin \mathbf{S}$, let $\ell \in \mathbf{S}$ be the seed in the block that vertices i and j are assigned to. From above, we have $c_{i\ell}, c_{\ell j} \le 2\lambda$, and, by the triangle inequality,

$$c_{ij} \le c_{i\ell} + c_{\ell j} \le 2\lambda + 2\lambda = 4\lambda. \qquad \textbf{[12]}$$

As there is exactly one seed in each block, $i, j \in \mathbf{S}$ is not possible, and we have considered all edges in $E(\mathbf{b}_{alg})$.

Remark 3: In some settings, a slight reduction in the sample size is acceptable or required, e.g., for financial constraints or when blocks are constructed before units are sampled using secondary data sources. In these cases, the algorithm can easily be altered into a 2-approximation algorithm. By terminating at the end of the third step and disregarding the unassigned vertices, one ensures that all remaining vertices are, at most, a distance of λ from the seed (where λ refers to the maximum distance in the optimal blocking of the selected subsample). Applying the triangle inequality proves that all edge costs in the blocking of the subsample are bounded by 2λ. It is also possible to apply a caliper to the blocking so as to restrict the maximum possible edge cost by excluding some hard-to-block vertices.

A concern when using a bottleneck objective is that densely populated regions of the sample space will be ignored, as the blocks in these regions will not affect the maximum edge cost. This is especially worrisome when there are a few hard-to-block vertices that result in a large λ. This can lead to poor performance, as covariate balance often can be improved by ensuring good block assignments for all vertices. However, as the presented algorithm does not directly use the bottleneck objective to form the blocks, it avoids this issue. Instead, its optimality follows from the use of the nearest neighbor subgraph as the basis of blocking, and from this graph's connection with the optimal edge cost as shown in Lemma 2.

The following theorem shows that our algorithm leads to approximate optimality not only in the complete sample but also in all subsamples. Thus, if there is a densely populated region, the algorithm ensures that the blocking is near optimal also within that region.

Theorem 4. (Local approximate optimality) *Let $\mathbf{b}_{sub} \subseteq \mathbf{b}_{alg}$ be any subset of blocks from a blocking constructed by the algorithm. Define $V_{sub} = \cup_{V_x \in \mathbf{b}_{sub}} V_x$ 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},*

$$\max_{ij \in E(\mathbf{b}_{sub})} c_{ij} \le 4\lambda_{sub}. \qquad \textbf{[13]}$$

Proof: Theorem 4 is proven in *Proof of Theorem 4*.

Heuristic Improvements. The algorithm allows for several improvements of heuristic character. Although the guaranteed optimality bound or complexity level remains unchanged, we expect these changes to improve general performance. In particular, the algorithm has a tendency to construct blocks that are too large. Although flexibility in the block size is beneficial—the main idea behind threshold blocking—the current version tends to overuse that liberty.

The first improvement exploits an asymmetry in the nearest neighbor subgraph that currently is disregarded. The cardinality condition is met as each seed and its $k-1$ nearest neighbors are assigned to the same block. However, in addition to those necessary neighbors, the current version assigns vertices that have the seed as their nearest neighbor to the block. With some minor alterations, detailed in *Algorithm Using Nearest Neighbor Digraphs*, the algorithm can use a $(k-1)$-nearest neighbor digraph to form the blocks. This digraph is such that an arc (i.e., directed edge) is drawn from i to j if j is among the $(k-1)$ closest neighbors of i. Using the digraph, one can differentiate whether an edge indicates that a vertex is a neighbor of the seed or vice versa. Fig. S1 illustrates the difference between the undirected and directed versions of the algorithm.

There is rarely a unique maximal independent set in the second power of the nearest neighbor graph (i.e., the seeds). The current version selects one arbitrarily. The second improvement is to choose the seeds more deliberately. As each seed assigns at least $k-1$ vertices to its block, a straightforward way to reduce the block sizes is to maximize the number of seeds—a larger set is expected to produce better blockings. The ideal may be the maximum independent set, but deriving such a set is an NP-hard problem. Most heuristic algorithms are, however, expected to perform well.

Third, despite the above improvements, the algorithm will occasionally produce blocks that are much larger than k. Whenever a block contains $2k$ or more vertices, it can safely be split into two or more blocks, each containing at least k vertices. As the algorithm ensures that all edge costs satisfy the optimality bound and no edges are added by splitting, this can only lower the maximum within-block cost. In *Greedy Threshold Algorithm*, we describe a greedy threshold blocking algorithm for splitting blocks that runs fast and is expected to perform well. This algorithm can also be used to block the complete sample, but will not perform on par with the approximation algorithm.

A fourth improvement changes how vertices are assigned in the fourth step. With larger desired block sizes, some blocks may contain peripheral vertices that are far from their seeds. In these cases, it is often beneficial to assign the remaining vertices in the fourth step to the block containing their closest seed instead of the block containing a closest neighbor. This avoids the situation where a vertex is assigned to distant block due to having a peripheral vertex close by. The optimality bound is maintained as Theorem 3 ensures that at least one seed exists at a distance of at most 2λ. If a vertex is not assigned to that seed, it must have been assigned to a seed that is at a closer distance.

Finally, once a blocking is derived, searching for moves or swaps of vertices between blocks can lead to improvements as in other partitioning problems (21). It is, however, seldom feasible to let such searches continue until no additional improvements are possible (i.e., until a local optimal is found), as the flexible block structure allows for a vast number of moves and swaps.

Simulation Study

We provide the results from a small simulation study. Apart from the original algorithm, we include versions that use the improvements discussed in *Heuristic Improvements*. Specifically, we include a version using the nearest neighbor digraph (the first improvement) and a version using the first three improvements. A version that uses all four improvements is generally only beneficial with larger block sizes and is included when such settings are investigated in Tables S1–S3.

For comparison, we also include the greedy threshold blocking algorithm discussed in *Greedy Threshold Algorithm* and the currently best-performing greedy fixed-sized blocking algorithm (12). When $k = 2$, we can also include a commonly used implementation of the nonbipartite matching algorithm (11).

We investigate a simple setting where each data point is sampled independently from a uniform distribution over a two-dimensional plane,

$$x_1, x_2 \approx \mathcal{U}(0,10), \qquad \textbf{[14]}$$

and similarity is measured as the Euclidean distance on this plane. Although many experiments will have data with higher dimensions, it is often not motivated to include all those dimensions when deriving blocks. Typically, one wants to reduce the dimensionality in a preprocessing step to extract the information that is most predictive of the potential outcomes (22). The investigated algorithms are, however, not restricted to low-dimensional data.

All simulations are run on a single CPU core reflecting the performance of a modern desktop computer. See *Implementation and Hardware* for details about the implementation of the algorithm and the hardware used to run the simulations.

Run Time and Memory. To investigate the resource requirements, we let each algorithm block samples containing between 100 and 100 million data points generated from the model above. Each setting was replicated 250 times. As time and memory use are quite stable over the runs, these replications suffice to investigate even small differences. In these simulations, the directed version performs almost identically to the original version, and it is omitted from the graphs to ease presentation.

Fig. 2 presents the time and memory needed for the algorithms to successfully terminate. All versions of the approximation algorithm run fast and terminate within a minute for samples sizes up to a few millions. With 100 million data points, the original version terminates within 11 min, whereas the version will all three refinements does so in less than 16 min—all very manageable in real applications. Memory use is increasing linearly at a slow rate for both versions. A modern desktop computer would have enough memory to block samples with tens of millions of units.

The three comparison algorithms paint another picture altogether. For the rather modest sample size of 20,000 data points, these algorithms take more than 20 min—up to 2 h—to terminate. Even more problematic is their extensive memory use. For samples larger than 50,000 data points, all three algorithms try to allocate more than 48 gigabytes of memory; under these settings, these algorithms do not terminate successfully, and no results can be shown.

Detailed results for these simulations and simulations with input data with higher dimensionality are presented in Tables S4–S6.

Minimizing Distances. To investigate how well the algorithms minimize distances, we increase the number of replications to 5,000; this statistic is less stable, and the differences between algorithms are smaller. However, with this number of replications, no difference can be attributed to simulation error.

The first three data columns of Table 1 show the maximum within-block distance, averaged over the simulation rounds, when the desired minimum block size is two. Refer to Table S1 for results when the desired minimum block size is four. We report values normalized by the performance of the approximation algorithm to ease interpretation. The two improved versions of the approximation algorithm lead to quite substantial decreases in the maximum distance. All versions of the approximation algorithm outperform the two greedy algorithms—for the fixed-sized version, drastically so. However, only the version with all three improvements performs better than nonbipartite matching.

In the fourth through sixth data columns of Table 1, the average within-block distance is presented. This measure is the objective of the greedy fixed-sized algorithm and nonbipartite matching, so it is not surprising that these algorithms show better performance. Nonbipartite matching is, here, the best performing algorithm, and only the approximation algorithm with all three improvements outperforms the fixed-sized greedy algorithm.

The seventh through ninth data columns in Table 1 present the average size of the blocks produced by the different algorithms. The improvements discussed in *Heuristic Improvements* are shown to be effective in taming the original algorithm's tendency to construct blocks that are too large. However, smaller blocks do not automatically lead to better performance. This is evident from the greedy threshold algorithm, which produces smaller blocks than the approximation algorithms but has worse performance. The two fixed-sized blocking algorithms produce blocks of constant size by construction.

Reducing Uncertainty. To investigate how the blockings affect an estimator's performance, one must specify a data-generating process for the outcome. The results are highly sensitive to the details of this process. Even blockings that are optimal with respect to within-block distances need not lead to the lowest variance. An extensive investigation is beyond the scope of this paper, and we provide only indicative results.

We consider a simple setting with two treatment conditions when the desired minimum block size is two. Refer to Table S2 for results when the minimum block size is four. The outcome (y)

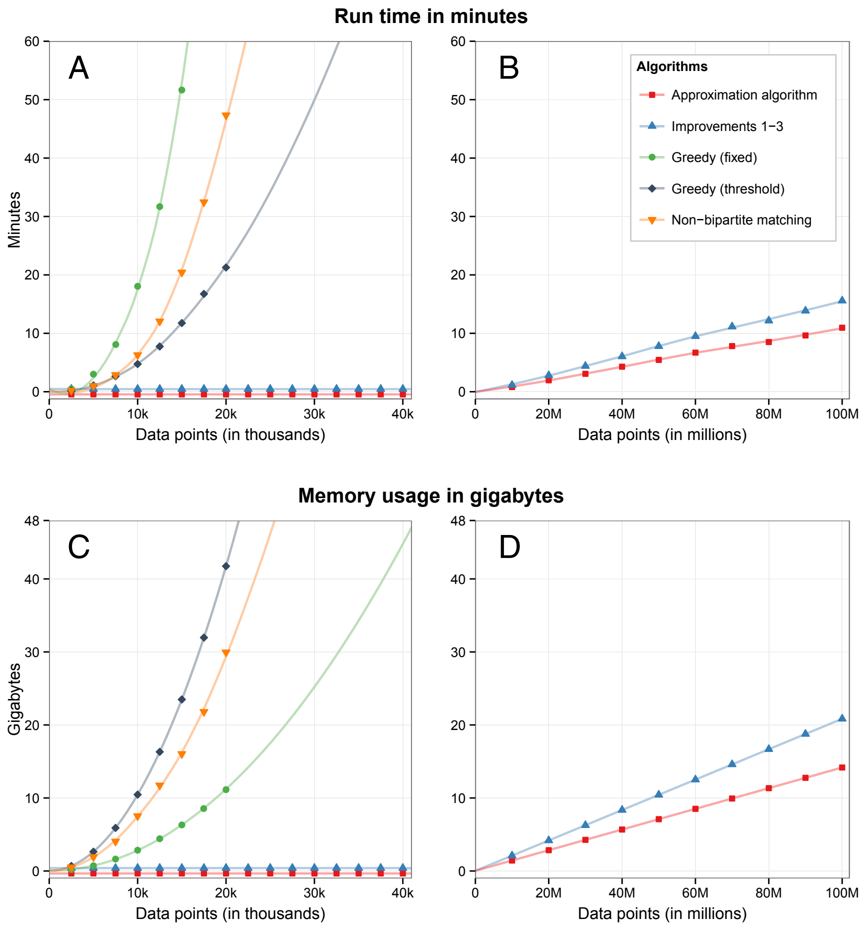

Fig. 2. Run time (*A* and *B*) and memory use (*C* and *D*) of five blocking algorithms with 2D input data over a range of sample sizes. Marker symbols are actual simulation results, and the connecting lines are interpolations. Results are presented with different scales due to the large differences in performance. Results are presented for all algorithms for sample sizes up to 40,000 data points (*A* and *C*), whereas results for sample sizes up to 100 million data points are only shown for the two approximation algorithms (*B* and *D*). No simulations were successful for the greedy algorithms and nonbipartite matching for sample sizes larger than 20,000, due to excessive run time or memory use. The undirected version of the approximation algorithm (shown in red) has almost identical run time and memory use as the directed version, as described in *Heuristic Improvements*, and its results are not shown in the figure. See Table S4 for detailed results.

is the product of the covariates with additive, normally distributed noise,

$$y = x_1 x_2 + \varepsilon, \quad \varepsilon \approx \mathcal{N}(0,1). \qquad [15]$$

Note that the treatment does not enter into the model and thus has no effect—the potential outcomes are equal. The covariates are highly predictive in this model—more so than one can expect in a real application. This enables the methods to make the most out of the covariate information and helps us differentiate between the algorithms' performances. For all blocking methods, we will use a difference-in-means estimator that is weighted by the size of each block to estimate treatment effects. Refer to *Estimation Methods* or ref. 23 for additional details on estimation with threshold blocking. We ran 5,000 replications in this setting, and results are presented relative to the approximation algorithm. Tables S3 and S7 present the results without normalization.

Table 2 presents results for the root of the average squared difference between estimates and the true treatment effect (RMSE) for each method's estimator. All blocking methods with proven optimality level perform well. For the smaller sample sizes, the approximation algorithm with all three improvements

Table 1. Performance of blocking algorithms by sample size: Maximum (Max.) and average (Avg.) within-block distances relative to approximation algorithm and average block size

	Max. within-block distance			Avg. within-block distance			Avg. block size		
Algorithm	10^2	10^3	10^4	10^2	10^3	10^4	10^2	10^3	10^4
Approximation algorithm	1.000	1.000	1.000	1.000	1.000	1.000	2.67	2.66	2.66
Directed version	0.917	0.895	0.883	0.933	0.933	0.933	2.55	2.54	2.54
Improvements 1–3	0.791	0.755	0.729	0.825	0.826	0.826	2.31	2.30	2.30
Fixed greedy	3.126	8.149	21.938	0.931	0.924	0.908	2.00	2.00	2.00
Threshold greedy	1.076	1.148	1.191	1.079	1.116	1.127	2.33	2.33	2.33
Nonbipartite matching	0.838	0.795	0.765	0.742	0.732	0.728	2.00	2.00	2.00

Table 2. Root mean square error relative to approximation algorithm by sample size

Method	10^2	10^3	10^4
Approximation algorithm	1.000	1.000	1.000
Directed version	0.973	0.987	0.999
Improvements 1–3	0.931	0.960	0.992
Fixed greedy	1.609	1.598	1.152
Threshold greedy	1.207	1.146	1.041
Nonbipartite matching	0.952	0.949	0.983
Unadjusted	6.092	15.158	20.710
Least square regression (OLS)	2.352	5.776	7.900

performs best, and nonbipartite matching is slightly better in the larger samples. All blocking methods seem, however, to converge as the sample size grows.

In addition to the six blocking methods, Table 2 includes the results of two methods that do not use blocking. In both cases, the RMSE is markedly higher than in any of the methods using blocking. When controlling for imbalances using ordinary least square regression (OLS), the RMSE is at least twice as large as that of any blocking method with proven optimality. When the estimate is completely unadjusted for imbalances, the RMSE is up to 20 times higher than for the approximation algorithm. However, this difference certainly overstates the benefits of blocking that one can expect in real applications, as the covariates are unusually predictive in this simulation.

Discussion

Our approximation algorithm enables large and complex experiments to make use of blocking. No feasible algorithm with proven optimality properties has been available for massive experiments. Although many of the commonly used blocking algorithms run in polynomial time, none run in quasilinear time as the approximation algorithm does. Polynomial time is not a sufficient condition for tractability with very large data (24). One can see this in our simulations.

The threshold blocking algorithm is expected to perform well in most cases, given its approximate optimality. However, nonbipartite matching, when it is feasible, is likely the best choice in experiments with matched-pair designs because it is exactly optimal. Our simulation results seem to point toward this conclusion as well. The matched-pair design is, however, limited to the case of only two treatment conditions, and the design complicates the estimation of SEs because block sizes of larger than two are sometimes needed, for example, to estimate conditional variances. Our approximation algorithm is therefore an important arrow in the quiver of experimental design.

Whenever blocking reduces imbalances in prognostically important covariates, blocking will improve the expected precision of estimates. In some settings, even when the covariates contain no information about the outcomes, blocking cannot increase the variance of the treatment effect estimator compared with when no blocking is done (25). However, theoretical results depend on the randomization model, estimand, and estimator used. There are some rare instances where blocking may decrease precision.

As an alternative to blocking, some advocate rerandomization when a given randomization results in poor balance in observed covariates (26, 27). Rerandomization restricts the randomization scheme, as assignments with poor balance are ruled out. If the rule for which randomizations are acceptable is precise and set a priori, randomization inference is well defined. One worries, however, that researchers will not use well-specified rules that they will later recall to restrict randomization distributions. Especially if the initial randomization results in good balance, it is doubtful that investigators will adjust their test levels as they had planned.

Far more common than blocking or rerandomization are *ex post* methods of adjusting experimental data such as poststratification or using a model-based estimator that incorporates covariate information. Such methods can work well. For example, poststratification is nearly as efficient as blocking: The difference in their variances is on the order of $1/n^2$, with a constant depending on treatment proportion (28). However, poststratification can increase variance if the number of strata is large and the strata are poorly chosen. Regression adjustment can provide significant gains in precision (29, 30), and model-based hypothesis tests can be asymptotically valid even when the adjustment model is misspecified (31). However, regression adjustment, like poststratification, may increase the finite sample variance, and will do so, on average, for any sample size, if the covariates are not informative (32).

A key argument in favor of blocking as opposed to *ex post* adjustment is that one is increasing the transparency of the analysis by building covariate adjustment into the experimental design. The results cited regarding poststratification and model adjustment assume that the investigator did not pick the strata or model as a function of the realized treatment assignment. One further assumes that the investigator does not run a number of adjustment models and then only report the one with the desired results. Human nature being what it is, this assumption is probably optimistic. A major benefit of randomized experiments, aside from the randomization, is that the design stage is separated from the analysis stage by construction (33). Blocking allows one to use design-based estimators that adjust for covariate information (34). The less that there is to do at the analysis stage, the less likely it is that the investigator will fish for particular results, unconsciously or not.

When researchers select adjustment models based on observed p-values, it is called p-hacking, and the habit appears to be prevalent (8). Because of concerns about p-hacking, there has been a move toward creating preanalysis plans for experimental studies in both medicine and the social sciences. Such plans force researchers to lay out in advance how they will analyze the experiment and what subgroups and outcomes are of primary interest. Unfortunately, evidence from medicine, where the practice is best established, shows that preanalysis plans are often ignored and allow significant leeway in selection of covariates, subgroups, outcomes, and adjustment methods, and readers and reviewers are rarely informed of departures (35). Blocking allows one to encode into the design of a study the covariates the investigator, a priori, thinks are important. After randomization, one may still adjust for these variables, as small imbalances may remain. Blocking then acts as an effective signal that the investigator intended to do such adjustments before seeing initial results. Moreover, some covariates may not be measured at the time of randomization, and they could be adjusted *ex post*, although concerns about p-hacking may arise.

Finally, blocking is motivated by partial knowledge about how the covariates relate to the outcomes. The performance of any blocking algorithm depends on how well the chosen similarity measure captures this relationship. As this choice is subject-specific, general recommendations are hard to come by. However, as noted in our remarks, if one has many covariates, some dimension reduction to those that most likely relate to the outcomes is often advantageous. If more complete knowledge exists, such as a good estimate of the potential outcomes under control, one would gain more precision by directly blocking on that estimate.

There are no free lunches in statistics, but blocking comes close. It has few downsides and risks relative to complete randomization, other than computational challenges, and any experimenter should be motivated to block their sample (23). In this paper, we have enabled the technique for experiments where it previously was infeasible.

ACKNOWLEDGMENTS. We thank Peter Aronow, Walter R. Mebane, Jr., Marc Ratkovic, and Yotam Shem-Tov for helpful comments. This research is partially supported by Office of Naval Research Grant N00014-15-1-2367.

1. Rubin DB (2008) Comment: The design and analysis of gold standard randomized experiments. *J Am Stat Assoc* 103(484):1350–1353.
2. Fisher RA (1926) The arrangement of field experiments. *J Minist Agric G B* 33:503–513.
3. Lewis R, Rao J (2015) The unfavorable economics of measuring the returns to advertising. *Q J Econ* 130(4):1941–1973.
4. Fithian W, Wager S (2015) Semiparametric exponential families for heavy-tailed data. *Biometrika* 102(2):486–493.
5. Athey S, Imbens G (2016) Machine learning methods for estimating heterogeneous causal effects. *Proc Natl Acad Sci USA* 113:7353–7360.
6. Ashley EA (2015) The precision medicine initiative: A new national effort. *JAMA* 313(21):2119–2120.
7. Permutt T (1990) Testing for imbalance of covariates in controlled experiments. *Stat Med* 9(12):1455–1462.
8. Simmons JP, Nelson LD, Simonsohn U (2011) False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychol Sci* 22(11):1359–1366.
9. Speed TP (1992) *Breakthroughs in Statistics*, Springer Series in Statistics, eds Kotz S, Johnson NL (Springer, New York), pp 71–81.
10. Imai K, King G, Nall C (2009) The essential role of pair matching in cluster-randomized experiments, with application to the Mexican universal health insurance evaluation. *Stat Sci* 24(1):29–53.
11. Greevy R, Lu B, Silber JH, Rosenbaum P (2004) Optimal multivariate matching before randomization. *Biostatistics* 5(2):263–275.
12. Moore RT (2012) Multivariate continuous blocking to improve political science experiments. *Polit Anal* 20(4):460–479.
13. Rosenbaum PR (1989) Optimal matching for observational studies. *J Am Stat Assoc* 84(408):1024–1032.
14. Sävje F (2015) The performance and efficiency of threshold blocking. arXiv: 1506.02824.
15. Cochran WG (1965) The planning of observational studies of human populations. *J R Stat Soc Ser A* 128(2):234–266.
16. Iacus SM, King G, Porro G (2011) Multivariate matching methods that are monotonic imbalance bounding. *J Am Stat Assoc* 106(493):345–361.
17. Hochbaum DS, Shmoys DB (1986) A unified approach to approximation algorithms for bottleneck problems. *J Assoc Comput Mach* 33(3):533–550.
18. Knuth DE (1998) *Sorting and Searching*, The Art of Computer Programming (Addison Wesley Longman, Redwood City, CA), 2nd Ed, Vol 3.
19. Friedman JH, Bentley JL, Finkel RA (1977) An algorithm for finding best matches in logarithmic expected time. *ACM Trans Math Softw* 3(3):209–226.
20. Vaidya PM (1989) An *O*(*n* log *n*) algorithm for the all-nearest-neighbors problem. *Discrete Comput Geom* 4(1):101–115.
21. Kernighan B, Lin S (1970) An efficient heuristic procedure for partitioning graphs. *Bell Syst Tech J* 49(2):291–307.
22. Imbens GW, Rubin DB (2015) *Causal Inference for Statistics, Social, and Biomedical Sciences* (Cambridge Univ Press, New York).
23. Higgins M, Sävje F, Sekhon JS (2015) Blocking estimators and inference under the Neyman-Rubin model. arXiv:1510.01103.
24. National Research Council (2013) *Frontiers in Massive Data Analysis* (Natl Acad Press, Washington, DC).
25. Imai K (2008) Variance identification and efficiency analysis in randomized experiments under the matched-pair design. *Stat Med* 27(24):4857–4873.
26. Hayes RJ, Moulton LH (2009) *Cluster Randomised Trials* (CRC Press, London).
27. Morgan KL, Rubin DB (2012) Rerandomization to improve covariate balance in experiments. *Ann Stat* 40(2):1263–1282.
28. Miratrix LW, Sekhon JS, Yu B (2013) Adjusting treatment effect estimates by post-stratification in randomized experiments. *J R Stat Soc Series B Stat Methodol* 75(2): 369–396.
29. Bloniarz A, Liu H, Zhang C-H, Sekhon JS, Yu B (2016) Lasso adjustments of treatment effect estimates in randomized experiments. *Proc Natl Acad Sci USA* 113:7383–7390.
30. Rosenblum M, van der Laan MJ (2010) Simple, efficient estimators of treatment effects in randomized trials using generalized linear models to leverage baseline variables. *Int J Biostat* 6(1):13.
31. Rosenblum M, van der Laan MJ (2009) Using regression models to analyze randomized trials: Asymptotically valid hypothesis tests despite incorrectly specified models. *Biometrics* 65(3):937–945.
32. Lin W (2013) Agnostic notes on regression adjustments to experimental data: Reexamining Freedman's critique. *Ann Appl Stat* 7(1):295–318.
33. Rubin DB (2008) For objective causal inference, design trumps analysis. *Ann Appl Stat* 2(3):808–840.
34. Aronow PM, Middleton JA (2013) A class of unbiased estimators of the average treatment effect in randomized experiments. *J Causal Inference* 1(1):135–154.
35. Humphreys M, de la Sierra RS, van der Windt P (2013) Fishing, commitment, and communication: A proposal for comprehensive nonbinding research registration. *Polit Anal* 21(1):1–20.
36. Kirkpatrick DG, Hell P (1978) On the completeness of a generalized matching problem, *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing* (Assoc Comput Machinery, New York), pp. 240–245.
37. Chen Y, Davis TA, Hager WW, Rajamanickam S (2008) Algorithm 887: CHOLMOD, supernodal sparse cholesky factorization and update/downdate. *ACM Trans Math Software* 35(3):22.