An approximation algorithm for blocking of an experimental design

Bikram Karmakar

Summary. Blocked randomized designs are used to improve the precision of treatment effect estimates compared to a completely randomized design. A block is a set of units that are relatively homogeneous and consequently would tend to produce relatively similar outcomes if the treatment had no effect. The problem of finding the optimal blocking of the units into equal sized blocks of any given size larger than two is known to be a difficult problem—there is no polynomial time method guaranteed to find the optimal blocking. All available methods to solve the problem are heuristic methods. We propose methods that run in polynomial time and guarantee a blocking that is provably close to the optimal blocking. In all our simulation studies, the proposed methods perform better, create better homogeneous blocks, compared with the existing methods. Our blocking method aims to minimize the maximum of all pairwise differences of units in the same block. We show that bounding this maximum difference ensures that the error in the average treatment effect estimate is similarly bounded for all treatment assignments. In contrast, if the blocking bounds the average or sum of these differences, the error in the average treatment effect estimate can still be large in several treatment assignments.

Keywords: Approximation algorithms; average treatment effect; blocking; experimental designs.

1. Introduction: context; approximation algorithm; review; contributions

1.1. *Blocking of an experimental design, paired study design, and larger block sizes.* In simple words, Cox (1958) describes a block in an experimental design as "a set of units, all the units in the set being as alike as possible." A blocked randomized design, by which treatments are randomized to units separately within each block, has been a fundamental and common sense tool to reduce the statistical error in treatment effect estimation (Fisher, 1935). In a successful blocking, the unit-to-unit variability is much smaller within blocks than across the blocks. Thus blocking removes some sources of variability from treatment comparisons, producing more precise treatment effect estimators than a design in which treatments are assigned to units completely at random. An example of blocking is a paired study design that has blocks of size two, exactly one in a pair is randomly chosen to be treated. Fogarty (2018) calls such designs finely stratified designs. Blocked designs are also useful for comparing multiple treatments, which can be from multiple levels of a treatment factor, or from combinations of multiple treatments factors, each with two or more levels. For example, to evaluate the effectiveness of mass media interventions to prevent smoking in youth, Flay et al. (1988) compared four different treatments: television only, television plus classroom program, classroom only, and a control, and Ellickson et al. (2003) compared a control and two treatments created from a combination of school based curriculum and the National Youth

Anti-Drug Media Campaign. When comparing multiple treatments using a complete blocked design, each block needs to have multiple units; e.g., four treatments can be randomized in a block of size 4 at a 1:1:1:1 ratio. For a review of block randomization and its practical benefits see Kernan et al. (1999). We will consider blockings with equal sized blocks.

Grouping similar units is a relatively straightforward problem when they are to be similar in a few discrete covariates. Moore (2012) makes the argument that blocking on continuous as well as discrete covariates is better than complete randomization or blocking only on a small number of discrete covariates. With many covariates, design of a paired study design has an "optimal" solution (Beck et al., 2016; Greevy et al., 2004; Lu et al., 2011). But, for equal sized blocks of size larger than two it is known that no algorithm can always quickly, in a sense to be made explicit, calculate the optimal solution for every given collection of units. This paper provides an algorithm, guaranteed to run quickly for calculating a near-optimal blocking for any collection of units, for blocked designs which are more complex than a paired study design.

Rerandomization (Morgan and Rubin, 2012) is another tool to constrain randomization. For specific problems, rerandomization creates identical designs to blocking. Parallelly, any blocked randomization may be achieved, albeit computationally inefficiently, by the rejective sampling in rerandomization. However, while the standard randomization-based ANOVA easily adapts to blocked designs, it is not generally appropriate for rerandomization designs. Also, highly constrained rerandomization designs put similar units in the same treatment arm in every assignment, thus reduces the effective sample size (Moulton, 2004).

The optimal pairing algorithm, mentioned above, minimizes the average within block differences in covariates of the units, but cares little about the largest pairwise difference. We refer to this algorithm as the Minimize Average Paired Difference (MAPD) algorithm. Rosenbaum (2017) notes: "... minimizing the mean pair difference is consistent with having dozens of poorly matched pairs." Moreover, this situation is more frequent in average cases than a practitioner will be comfortable to accept (see, §4.3 for simulations). Many researchers have advocated for controlling the largest of the in-block differences rather than the average in-block differences (Higgins et al., 2016, and Moore and Schnakenberg, 2016). Cochran (1965) noted that, a few large differences is more problematic than many small ones because a regression adjustment during analysis easily mitigates the small differences (Rubin, 1979; Snedecor and Cochran, 1967, Chapter 14).

In a designed experiment, a practitioner would want to investigate that the imbalance in the covariates after treatment assignment is not large. For a blocked randomization design that bounds the largest within block difference, the covariate imbalance is bounded for any treatment assignment. But this feature is not enjoyed by blocked designs that bound the average pairwise differences. Specifically, §3 shows that bounding the maximum difference ensures that the error in the average treatment effect estimate is similarly bounded for all treatment assignments, but not when bounding the average pairwise differences.

1.2. *Goal: the blocking problem, polynomial time algorithms, and approximation algorithms.* Our blocking problem, formalized in §2.2, is to find, for a given set of study units and the differences in the covariates among them, blocking of the units into blocks of a given size so that the maximum of the differences of any two units in the same block is minimized. Section

5.2 considers a related but different blocking problem that aims to minimize the maximum of the average in-block differences. We aim to find algorithms that run in time polynomial in the number of units to solve these blocking problems.

Algorithms that run in polynomial time in the size of the problem are desired, but often unattainable for many optimization problems; these computationally hard problems are categorized as NP problems according to their computational complexity (Papadimitriou and Steiglitz, 1998). Proposition 4 establishes that the blocking problem for blocks of any size more than two does not permit a polynomial time solution, unless P=NP.

Mixed integer programming problems (MIPs) is another class of theoretically hard problems. Modern mathematical optimizers have been used to solve MIP formulations of certain observational study designs (Zubizarreta, 2012; Zubizarreta and Keele, 2017). Although, they do not provide computation time guarantees. While our blocking problems can also be formulated as a MIP, this formulation is theoretically and practically intractable.

The compromise we make for the lack of any polynomial time solution to the blocking problem for larger block sizes is an approximation algorithm (Williamson and Shmoys, 2011).

***Definition* 1.** *An approximation algorithm for an optimization problem is a polynomial-time algorithm that for all instances of the problem produces a solution whose value is within a factor of of the optimal solution for some constant .*

An approximation algorithm ensures efficiency and assures that the produced solution is not too far from the optimal solution and, therefore, preferable to a heuristic algorithm.

1.3. *Review of existing blocking methods.* Consider study units to be blocked using numbers, summarizing the pairwise differences of the units on their covariates. In practice, paired study designs are created using the MAPD algorithm that solves a weighted non-bipartite matching problem for a complete graph of vertices with the covariate differences between the two units as the edge weights (Beck et al., 2016; Lu et al., 2011).

A few methods are available in the literature for blocking with larger block sizes. Moore (2012) proposed the "optimal-greedy" method. Given units and the desired block size , this algorithm first creates pairs greedily. Then, if , it greedily matches one unit to each pair. For larger , the method proceeds similarly, until a desired blocking is created. Although it is in the name, the "optimal-greedy" method is not known to have any optimality property. Karmakar (2018) proposed a randomized algorithm for blocking which first randomly chooses units as template units for the blocks and then assigns the remaining units optimally to these n blocks at a ratio : to create a blocking. Template matching methods have been used elsewhere in the literature, e.g., Hu et al. (2018) and Silber et al. (2014). Neither of these methods provide any performance guarantee.

Higgins et al. (2016) proposed the 'threshold blocking' method, which is an approximation algorithm. This method aims to create a blocking that minimizes the largest difference between any two units in the same block. But, the design can have any block size of . There are two challenges with such designs. First, when there are treatments, it is unclear in general how to assign the treatments to the units in the resultant blocks. Simulations show that the blocks created by this algorithm can often have a much larger number of units than ; see the supplement. Second, tests for the treatment effects are more efficient when the sample sizes for the treatment combinations are the same.

1.4. *Contributions: estimation error bounds, the blocking problems, approximation algorithms, extensions.* The definition of the technical terms used in this section are given in §2.1. The blocking methods presented in the paper aim to create a blocking that minimizes the largest difference of any two units in the same block.

In §3.1 we study the estimation error for the average treatment effect in blocked randomized designs. Theorem 1 establishes a uniform upper bound for the error for all treatment assignments in terms of the bound on the maximum pairwise differences in the units in the same block. If, on the other hand, the average of the pairwise differences is bounded, the theorem also gives a lower bound on the error for a large class of treatment assignments that gets worse with the size of the blocks and the number of blocks.

The blocking problem for a paired study design is formulated as a *non-bipartite matching* problem in §4.1. Algorithm 1, in §4.2, solves this non-bipartite matching problem in polynomial time. The algorithm starts with an empty *match* and expands the size of the *match* by augmenting it with the *shortest augmenting path* for the problem (Derigs, 1988) and finally creates a *maximum cardinality match*, which, as Theorem 2 shows, solves the non-bipartite matching problem. The simulation study in §4.3 compares Algorithm 1 with the MAPD algorithm.

In §5.1, we propose Algorithm 2, to create a blocking with block size , for a positive integer . Theorem 3 shows that Algorithm 2 is an approximation algorithm with approximation factor . We further study the local property of this algorithm. Theorem 5 shows that Algorithm 2 may fail to create a locally optimal blocking. We propose Algorithm 3, that uses solutions to certain *bipartite matching problems* for local improvements of Algorithm 2, in §5.2. In Section 5.3, we then show that, with some small changes, Algorithm 2 and 3 can be used for blocking of any block size.

We compare our proposal to existing blocking methods using synthetic and simulated data sets. Using a synthetic dataset we compare our proposal with MAPD in paired randomized designs. The results in §6 show that using the proposed method gives much larger power for testing the null hypothesis of no treatment effect, and smaller root mean squared error in average treatment effect estimation. The simulation results in §7 compare the treatment effect estimation error in the proposed methods to several existing methods in paired randomized designs and blocked randomized designs with larger blocks.

The proposed methods have been implemented in `R` and are available from https://github.com/bikram12345k/BlockingAlgo/. The repository also includes code to reproduce the simulation and data analysis results of this paper. An online supplement includes, among other things, all the proofs and details of the implementation of the methods.

2. The problem of creating blocks

2.1. *Notation and some definitions.* For a finite set , denotes the cardinality of the set. A graph is an ordered pair of sets , where is a finite set of vertices and is the edge set. Two graphs and are isomorphic if there is a bijection so that if and only if . For an equivalence class of isomorphic graphs of vertices, the vertices may be labeled by . A cost function on the edge set of a graph is a real valued

function on . Let be the cost of the edge . Throughout this paper is assumed to be nonnegative, without any loss of generality. In our discussion, we will often be interested in complete graphs that has for all .

For a graph and a partition of its vertices $\mathcal{V}$, the projected graph $\mathcal{V}$ $\mathcal{V}$ $\mathcal{V}$ is defined as $\mathcal{V}$ if for some and , .

Below we give definitions of some graph theoretic concepts. For a textbook discussion of these concepts see Bondy and Murty (1976). A *match* of a graph is a subset of the edge set such that no two edges, called matched edges, in share a common vertex. A match is a *maximum cardinality match* of if for any other match of , it satisfies . A *perfect match* is a match where each vertex in is in a matched edge. In a complete graph, a perfect match is a maximum cardinality match and vice versa.

A *path* is a sequence of vertices, which do not repeat, so that any two consecutive vertices in the path are connected by an edge. A path in is called an *alternating path* with respect to a match if the alternating edges in the path are in and not in . An *augmenting path* in with respect to a match is an alternating path which starts at an unmatched vertex and ends at an unmatched vertex. An *alternating cycle* is an alternating path where the start and the end vertex are the same. Suppose is a matching of and is an augmenting path with respect to which is expressed as the set of all edges connecting two consecutive vertices in the path. The *augmented match* of with respect to is defined as . Notice that, ; thus the name augmenting path for . For an alternating cycle , define , another match with .

A graph is a complete bipartite graph if , \ , and if and only if and . Clearly, if a perfect match of exists, then , and is also a maximum cardinality match with . The *bipartite matching problem* is the problem of finding a perfect match in a bipartite graph that aims to optimize certain objective. Bipartite matching problems are regularly solved in designs of observational studies, to match a treated group and a control group on their propensity scores and observed pre-treatment covariates, see Hansen and Klopfer (2006); Pimentel et al. (2015a, 2018); Rosenbaum (1989); Yang et al. (2012); and Zubizarreta (2012). Strictly speaking, a *non-bipartite matching* is a matching on a graph that contains an odd cardinality cycle. For the purpose of this paper, to keep our discussion relevant to blocking problems, by a non-bipartite matching we mean a matching on a complete graph.

2.2. *The blocking problem.* For a complete graph of vertices, , a blocking of size is a set of non-overlapping blocks of size each, $\mathcal{B}$ with , \ for and . Given , the blocking problem is to find a blocking $\mathcal{B}$ that minimizes

$$\mathcal{B} \tag{1}$$

Higgins et al. (2016) consider a different version of the problem, called the threshold blocking. They require that the blocks are of size at least , , instead of the hard cardinality constraint . Both of these problems are NP-hard when . A proof of hardness of the threshold blocking problem is given by Higgins et al., and a proof of the

hardness of (1) is given in Proposition 4. As was discussed in §1.3, in experimental designs, a blocking with the hard cardinality constraint is more practical than the threshold. It is well recognized that a hard cardinality constraint makes an optimization problem comparatively more difficult to solve (Papadimitriou and Steiglitz, 1998; Williamson and Shmoys, 2011).

3. **Treatment effect estimation for blocked designs**

3.1. *Efficiency of treatment effect estimation using blocked designs.* Higgins et al. (2016) note the advantage of using a blocking objective that minimizes the maximum within-block differences that it guarantees the average imbalance cannot exceed this maximum after the treatment is assigned; minimizing the sums or averages does not provide such a guarantee. Here we state a result establishing further that the above fact has implication in the efficiency in the treatment effect estimation. Specifically, Theorem 1 below shows that when the maximum difference is bounded, the absolute error in the average treatment effect estimate is similarly bounded uniformly for all treatment assignments. While, when the average difference is bounded, the absolute error in the average treatment effect estimate can still be large for some treatment assignments. It is important to note that the choice of blocking objective is important only in so far as it impacts the efficiency of the analysis, while both will provide an unbiased estimator of the (intended to treat) treatment effect under a randomized design.

Consider a single block of size . We are interested in estimating the treatment effect under the above two strategies for blocking. For multiple blocks, the upper bound in Theorem 1 for a blocking that bounds the maximum in-block difference remains the same and the lower bound for a blocking that bounds the average difference becomes worse.

In this section, we make it explicit that the difference, , is based on the covariates. Let denote the dimensional vector of covariates for unit and let denote the distance between units and .

Under the potential outcomes framework let be the potential outcome for unit under treatment . We focus only on binary treatments, so that or . Let . We are interested in estimating the average treatment effect on the design units

$$\text{ATE} \quad -$$

Assume a completely randomized assignment of treatments and (call them control and treated, respectively) among these units so that of these units are assigned treatment. Let be the treatment indicator for unit and $\mathcal{Z}$ denote the set of all possible treatment assignments where are treated and the rest are control; thus $\mathcal{Z}$. We assume that the observed outcome of unit is . We can unbiasedly estimate ATE using the Horvitz-Thompson estimator

$$\text{ATE} \quad — \qquad ———$$

which is the difference of the mean outcomes of the treated and control units.

Following Iacus et al. (2011), who show the benefit of minimizing the maximum imbalance in designing a matched-pairs observational study, suppose the outcome and . For the function of -dimensional covariates, define the class of Hölder continuous functions parametrized by and as

$\mathcal{G}$

When this is a class of Lipschitz continuous functions. Assume without loss of any generality that .

***Theorem* 1.** *Suppose satisfies the triangle inequality, i.e., for any and .*

(a) If then

$$\text{ATE} \quad \text{ATE}$$
$$\mathcal{Z} \quad \mathcal{G} \qquad \mathcal{G}$$

(b) There exists values satisfying so that the following is true.
Let . Suppose $\mathcal{Z}$ be the set of all possible assignments to units according to assignments of units to treated and control. Then there is a subset $\mathcal{Z}$ of $\mathcal{Z}$ of cardinality $\mathcal{Z}$ such that the following is true for any

$$\text{ATE} \quad \text{ATE}$$
$$\mathcal{Z} \quad \mathcal{G} \qquad \mathcal{G}$$

While the error bound in part (a) of the theorem is easy to understand, to understand the error lower bound in part (b) consider and . Then there is a treatment assignment that gives an error of at least . This error gets larger with the ratio . It is not possible to know if we will face this error until the treatment has been assigned. The role of the index in the result is to highlight the fact that there is not one, but many relatively poor treatment assignments where a poor assignment of only of the units is enough to make the whole assignment poor no matter how the other units are assigned. We include to allow for the possibility of continuous values. Theorem 1(a), on the other hand, shows that bounding the maximum with a small value of achieves a primary goal of a blocked design which is to guarantee that the estimate is good for all treatment assignments to the blocked units, i.e., controls haphazard variation (Cox and Reid, 2000).

When there are blocks of size the unbiased difference-in-means estimator is the difference of the mean outcomes of all the treated and all the control units across the blocks. In this case the bound in (a) remains the same where is an upper bound for the maximum difference of any pair of units in a block, but the lower bound in (b) will now have in place of where is an upper bound for the average of all pairwise differences of two

within-block units. The worsening performance is because poor assignment is now possible across blocks and the errors accumulate.

***Remark* 1.** The lower bound in Theorem 1(b) for the error in ATE estimation is derived by a construction of a set of distances . But we remark that such set of values is not pathological. Suppose is drawn randomly from a standard normal distribution and . Then Lemma S1 in the supplement gives a minimum conditional probability of observing a set of values that implies the lower bound given that the average pairwise difference is at most . Using that lemma we calculate, when , and the probability is 3.2%, and when , and the probability is at least 5%. Additionally, an even worse lower bound in Theorem 1(b) can be proved if varied with .

***Remark* 2.** The proof of the theorem also gives a clue as to when blocking with objective (1) will provide an improvement on blocking that minimizes the average of all pairwise differences. A blocking with many small pairwise differences and some relatively large pairwise differences can have a small average pairwise difference but could also be a poor blocking. This is because when different treatments are assigned to units that are further from each other, the imbalance in the study will be large. Similar effect is seen with multiple blocks where some blocks have units with larger differences. This is consistent with observations of other researchers such as Cochran (1965). Under (1) such blocking is less likely as it does not attempt to get the smallest average difference. On the other hand, if under a certain distribution of there is a feasible blocking where the values of the in-block pairs are almost all very small, then we are unlikely to see benefits of blocking by minimizing (1). Still, we also do not expect a substantial loss in efficiency because largest in-block pairwise differences in such blockings will also not be too large.

3.2. *Some remarks on practical use of blocking.* The discussions of §4 and §5 are broadly on the methodological aspects of blocking units. Here we give some remarks on practical use of blocked randomization.

Blocking is a fundamental strategy to constrain randomization in an experiment (Fisher, 1935; Bailey, 1987). Blocking may be incorporated, as indicated in §1, with different types of randomized designs, such as clustered randomized design (Kelcey et al., 2017), factorial design (Box and Hunter, 1961; Cheng and Mukerjee, 2001) and split-plot design (Federer, 2007; Robinson, 1970). Two methods for analyses of blocked designs are randomization or design based analysis that bases the analysis on the randomization of the treatment assignment mechanism and model based analysis that uses a statistical model for the outcomes. Cox and Reid (2000) give a textbook discussion of these analyses methods. Many common blocked designs may be analyzed with the `R` package `ri2`, while model based analyses of most blocked studies are now common to standard statistical software.

Generally, a blocking will improve precision in treatment effect estimation when the covariates are correlated with the outcome. In our simulations we use Mahalanobis distance based cost functions. With more knowledge on how the outcome depends on the covariates, specific to a study, we can define a better cost function. Problem (1) proposes to minimize the maximum in-block paired distances. There can be multiple blockings that give the optimal objective value of problem (1). This would not hinder the use of one of these several

blockings in the design. But it would be useful to investigate a blocking prior to experimentation. In an uncommon situation the blocking might be unacceptable because most in-block distances are relatively much smaller than a few large in-block distances. The design, in this situation, could be improved by removing a few units from the study that are far from the others. A blocked design formed with a certain blocking objective can help only in increasing precision in treatment effect estimation. Our numerical results in §6 and §7 highlight the efficiency gain in statistical inference due to blocking over completely randomized designs and blocking using the proposed objective (1).

4. Two arm experiments: blocks of size two

4.1. *Non-bipartite pair matching problem to minimize the largest pairwise difference.* Consider two treatment arms, or a treatment arm and a control arm, to be randomized in a paired study design. In this special case with , the blocking problem (1) has a polynomial time solution, and this section presents an efficient algorithm. Problem (1) is equivalent to the problem of finding a perfect match , on a complete graph of vertices and cost function , that solves

(2)

The blocks are the resulting matched pairs. This formulation of the blocking problem opens up opportunities to use results from graph theory to solve the problem.

The MAPD algorithm of Greevy et al. (2004) and Lu et al. (2011) solves a different non-bipartite matching problem that minimizes or , rather than the maximum objective in (2). This nonbipartite matching problem with the sum of weights objective can be solved using the shortest path method of Derigs (1988). For practitioners, Derigs' algorithm has been made available in the R package `nbpMatching` (Beck et al., 2016).

4.2. *An algorithm to solve the non-bipartite matching problem.* Before presenting the algorithm to solve (2), it is useful to provide the pieces that lead to such algorithms for solving non-bipartite matching problems. A fundamental theorem of matching in a graph due to Berge states that, a matching is a maximum cardinality matching if and only if it admits no alternating path. The simple proof of Berge's theorem can be found in any textbook on graph theory. Although this theorem characterizes all perfect matchings in a graph, it does little to help build an algorithm to solve for a matching that satisfies a property, which in our case is to find the matching with the minimum of maximum within pair difference. The following theorem alludes us to a more practical algorithm. For any matching of the graph with edge cost define

Theorem 2. *Call an alternating cycle with respect to a matching a negative alternating cycle if . For an unmatched vertex with respect to , let $\mathcal{P}$ be the set of all augmenting paths with respect to . We call $\mathcal{P}$ a shortest augmenting path if*

for all $\mathcal{P}$

With these definitions, we have:

(a) A maximum cardinality matching, equivalently a perfect matching, solves (2) *if and only if it allows no negative alternating cycle.*
(b) If is a matching of that permits no negative alternating cycle, then for any shortest augmenting path , the augmented match permits no negative alternating cycle.

This theorem is the backbone of Algorithm 1 below. The starting point of the algorithm is an empty match and the goal is a maximum cardinality matching that allows no negative alternating cycle, which by part (a) of the theorem solves (2). Theorem 2 is adopted from Derigs (1988) for the maximum objective. The only if implication of part (a) of the theorem follows by a proof by contradiction argument: in case the implication is incorrect, one notes that the matching that solves (2) can be found by adding a few alternating cycles to . The complete proof of the theorem is given in the supplement.

The smallest matching does not allow any negative alternating cycle. Immediately, for any if we let such that for all , then by Theorem 2(b), is a matching, now with one pair, that permits no negative alternating cycle.

If there is an algorithm that finds a shortest augmenting path for any given matching, then an algorithm that at each step augments the current matching by the shortest augmenting path, will find the optimal matching that solves (2) in exactly steps; since each step increases the size of the matching by one. Algorithm 1 is such an algorithm.

Algorithm 1. Initialize: , .
while: :
do (i)
if: is unmatched in .
; break.
(i) **continue.**
Find $\mathcal{P}$ which is a shortest augmenting path.

end while.
Return M.

A shortest augmenting path for a matching that starts at an unmatched node can be solved in polynomial time by building a tree rooted at in a depth first manner and using a certain kind of labeling strategy. A shortest path finding algorithm is given in the supplement.

***Remark* 3. Search for value algorithms.** One strategy of solving minimization problems with the maximum objective as in (2) is an iterative approach (Derigs, 1984; Gabow and Tarjan, 1988). This approach searches for the optimal value of the objective rather than the optimal matching which Algorithm 1 does. Let . Then this algorithm would first guess a lower bound of , say . With this value, one checks whether there is a perfect matching when the edge set of the graph is restricted to . If such a matching exists, . If unsuccessful, it would have to make a better guess of . The performance of such a 'search for value' algorithm depends on the starting guess. A

bad guess can lead to a long search for the solution. Compare this to Algorithm 1, which is a one stop algorithm for finding the best match using Theorem 2.

***Remark* 4. Computational complexity.** The linear search to find an unmatched unit in each step of Algorithm 1 requires comparisons. An efficient way to encode is to use a binary valued sparse matrix whose non-zero value indicates that is a match in . Whereas, an efficient data structure for is a sequence of alternating edges. If one uses these data structures, the augmenting step also requires operations. Finally, the algorithm to find the shortest augmenting path requires an computation time. Hence, as Algorithm 1 loops for times, it has an overall computation time requirement of , which is same as that of finding the maximum of a sequence of length , or of inverting a square matrix of size by using Gauss-Jordan elimination method. This is also the computational time complexity of the MAPD algorithm.

4.3. *Comparison with the standard pair blocking algorithm.* In this section Algorithm 1 is compared with the standard Minimize Average Paired Difference (MAPD) algorithm. Using two scenarios we compare these two algorithms in their abilities to create homogeneous pairs. In each scenario there are 100 units, and either algorithm creates pairs. Results are given in Table 1 and in Figure 1.

Scenario 1 simulates the differences in the following way. For every pair of units, and , let the difference , where and are simulated independently from a mixture of two uniform distributions. One component of this mixture distribution is uniform on and has weight . This component indicates that the units are similar in a certain covariate. The other component of the mixture is a uniform distribution on and has weight . This component indicates that the units are different in the value of the covariate. A value of less than indicates that both and are from a uniform distribution on . On the other hand, if the distance is more than then both and are from a uniform distribution on . A value between and is also possible, if the two numbers and are from the two different components of the uniform mixture. In each simulated instance, differences are simulated — on an average, in each instance, of these differences are larger than and are smaller than . Either Algorithm 1 or MAPD selects of these differences.

The second simulation scenario calculates the differences by simulating two random variables and for each unit , and defines , the Euclidean distance of the covariates for units and . The random variables are independently sampled from mixture normal distributions. The first variable is simulated independently from mixture of three normal distributions with means and , each with variance 1, with weights , and . The second variable, , has the same distribution as . Since both and have the same variance and they are independent, is also proportional to the Mahalanobis distance between and .

The boxplots in Figure 1 show the differences of the pairs selected by the two algorithms. Compared to Algorithm 1, the MAPD algorithm is aggressive in finding many pairs which are very close to each other, as it finds, in each simulated instance, the 50 pairs that minimizes the average difference. This algorithm does poorly in ensuring that all the pairs are close.

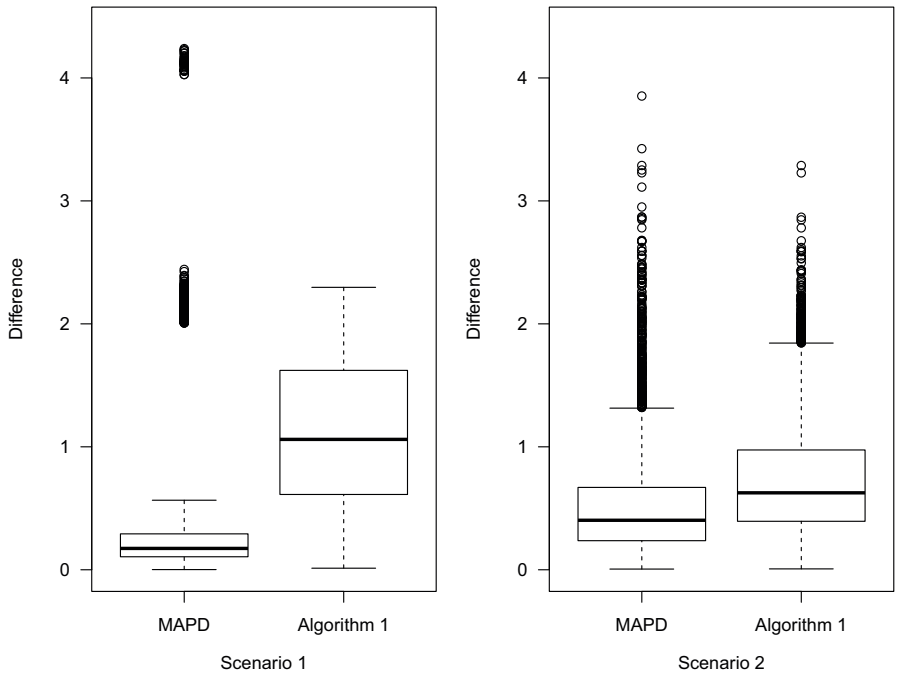


FIGURE 1. Boxplots of the pair differences from the two simulation scenarios, comparing MAPD with Algorithm 1. In each boxplot, 50×100 paired differences from 100 simulated instances are plotted. MAPD overemphasizes very small differences and creates many pairs with very large differences. Algorithm 1 balances the most largest paired differences and the average difference.

TABLE 1. Comparison of MAPD with Algorithm 1 on the worst pairs. This table shows the summary statistics of the two largest differences in each simulated instances, i.e., of a total of 2×100 distances based on 100 iterations. The final column is the average of the ratios of the largest differences from Algorithm 1 over the largest differences from MAPD in the same simulated instance.

Simulation		Quantiles of the 2 largest differences					Avg. ratio of
Scenario	Method	50%	75%	95%	99%	100%	the max. (%)
Scenario 1	MAPD	2.23	2.32	4.15	4.22	4.24	100
	Algorithm 1	2.15	2.18	2.23	2.26	2.30	82
Scenario 2	MAPD	1.89	2.22	2.84	3.29	3.85	100
	Algorithm 1	1.76	2.05	2.59	2.87	3.29	89

In 40% of the instances the largest pairwise difference for the pairs selected by the MAPD algorithm was more than 4. Compare this to Algorithm 1, where in only 10% of the simulated instances the maximum pairwise difference is more than 2; the largest of all of them is 2.30. When designing an experiment, a practitioner might be lucky to find themself in a situation where the maximum of the differences for the pairs created by MAPD is not extremely large. But, when in Scenario 1, in a large number of the instances the design would include a few unacceptable pairs. A similar observation is made from the boxplots for Scenario 2,

albeit the worse pairs for MAPD are less pronounced. Overall, a small value of the average difference of the pairs can give a false impression of all the blocks being homogeneous in the design. Table 1 further zooms in on the large pairwise differences of Figure 1. From each simulated instance the two largest paired differences each from the pairings created by the two algorithms are collected, and the five quantiles of these values are reported. Algorithm 1 consistently finds pairings with the largest difference that is never worse than MAPD, and it is often considerably better — the average improvement is 18% for Scenario 1 and 11% for Scenario 2 in the simulations.

We coded Algorithm 1 completely in `R`, and MAPD has been coded in `Fortran`. In Scenario 2, the average runtime for MAPD was 4/100-th of a second, while the average runtime for Algorithm 1 was 0.83 seconds with a standard computer (a controlled environment was not used for this benchmarking). This difference in the runtimes is most likely because our coding is less than efficient.

5. **Blocking with block sizes larger than two**

5.1. *An approximation algorithm.* Of the two simulation scenarios considered in §4.3, the cost function for the second scenario is a proper distance; it is non-negative, with , , and , for any three units and . The cost function for Scenario 1 does not satisfy the triangle inequality. Algorithm 1 does not require that the cost function be a proper distance. In this section we consider the blocking problem for blocks of size larger than . This problem, problem (1) with , is a hard problem. More than that, without any restriction on the values, it is a hard problem to find an efficient algorithm that guarantees a blocking close to the optimal blocking (see, Definition 1, §1.2).

We first discuss the blocking problem where the blocks are of size for some ; later, in §5.3, we show that the algorithm proposed to solve this special case of the problem can be modified slightly to solve the problem for any . Algorithm 2 below takes the differences between units as input and outputs a blocking with blocks. The first call by Algorithm 2 is to Algorithm 1 to create a partition of the units into pairs. Let's call them $\mathcal{V}$. Next, it projects the complete graph of the units into the partition $\mathcal{V}$ to define $\mathcal{V}$, which is a complete graph of $\mathcal{V}$ vertices. Define the cost function for $\mathcal{V}$ as

$$\mathcal{V} \tag{3}$$

For the consistency of the notation, let $\mathcal{V}$ and $\mathcal{V}$ if $\mathcal{V}$. For a perfect matching of a graph on $\mathcal{V}$, let $\mathcal{V}$ denote a partition of $\mathcal{V}$ with respect to .

Algorithm 2. Initialize: $\mathcal{V}$.
while: :
Output of Algorithm 1 with arguments $\mathcal{V}$ and $\mathcal{V}$ $\mathcal{V}$
Update $\mathcal{V}$ $\mathcal{V}$ and correspondingly $\mathcal{V}$.
Update $\mathcal{V}$ using (3).

end while.

Return $\mathcal{BV}$.

This algorithm calls Algorithm 1 times. The first call creates blocks of size 2, i.e., pairs, such that the maximum paired distance is minimized. Each subsequent call halves the number of blocks by pairing the previous set of blocks so that the maximum of the paired within block distances is minimized in this local problem.

We now present a theoretical analysis of Algorithm 2. We study the performance of the algorithm in creating a blocking close to the optimal blocking that solves problem (1) and to the locally optimal blocking. When , , Algorithm 2 collapses to Algorithm 1, thus it creates the optimal blocking. Let be the value of the objective for $\mathcal{B}$, the blocking created by Algorithm 2.

***Theorem* 3.** *Suppose the cost function satisfies the triangle inequality: for any and . Let be the smallest value of* (1) *attained by the optimal blocking. Then, , with equality when . In other words, Algorithm 2 is a -approximation algorithm for the blocking problem* (1).

Theorem 3 requires a minimal structural assumption on , that it satisfies the triangle inequality. For an intuition of the proof consider the case , where the approximation factor is . In the proof of the theorem the approximation factor 3 appears from a possible mispairing and by the first call to Algorithm 1, which are then put into the same block by Algorithm 2. When this mispairing happens, the value of is at least , and a worst value of comes from the following implication of the triangle inequality

(4)

Algorithm 2 does not correct for this mispairing which leads to the approximation factor 3. Under this structural assumption, part (b) of Proposition 4 shows that, no efficient algorithm can do better than what Algorithm 2 guarantees. Part (a) of Proposition 4 does not assume the triangle inequality of the and concludes that no approximation algorithm exits.

***Proposition* 4.** *Let and as in Theorem 3. Unless P=NP,*

(a) there does not exist any polynomial time algorithm so that if is the objective value of the solution produced by the algorithm then for every instance of the problem, for some .

(b) if the cost function satisfies (4)*, there does not exist any polynomial time algorithm so that, if is the objective value of the solution produced by the algorithm then for every instance of the problem, , for some .*

Proving the hardness results of Proposition 4 requires establishing a *reduction* of problem (1) to another problem that is known to be hard. The reduced problem is such that, an algorithm that can solve the original problem can be used to solve the reduced problem. The provable extent of hardness of a problem depends on this reduction, and on the structure on the original problem. Therefore, the two reductions in part (a) and part (b) of Proposition 4 are different, as they assume different structures of the cost function. The blocking problem (1) is reduced to the following problem: Given two graphs and with , can the vertices of be partitioned into disjoint sets such that, for , the subgraph induced by is isomorphic to ? Kirkpatrick and Hell (1978)

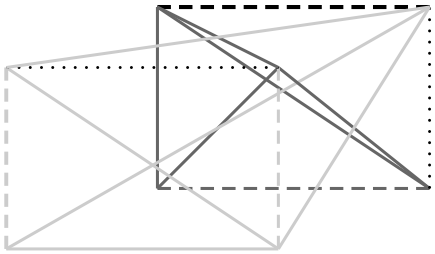


FIGURE 2. An example with two blocks of size four; total units. A solid line between two units indicates a cost of 1, a dashed line indicates a cost of , and a dotted line indicates a cost of , and absence of a line between two units indicates a cost of ; is an arbitrary constant.

proved that this is a hard problem for any graph with . A complete proof of Proposition 4 is given in the supplement.

In the remaining of this section we analyze the local performance of Algorithm 2. For a blocking , define the local neighborhood of as

$$\mathcal{LB} \quad \mathcal{B} \quad \setminus$$

$$\text{for all} \qquad \text{for some} \tag{5}$$

Here denotes the permutation group of and denotes the symmetric difference operator of two sets. A blocking in is also a feasible blocking for problem (1), i.e., consists of non-overlapping blocks of size , but it can be different from by at most one unit in each block. For simplicity, suppose in (5) is the identity permutation, for all . Then, means that the two blocks are identical, and implies that all but one unit in is in . Because , and are the only two possible values of . When , any local neighborhood is the set of all feasible blockings of problem (1). Theorem 5 analyzes the local performance of Algorithm 2.

***Theorem* 5.** *Let be the output of Algorithm 2. Suppose the cost function satisfies the triangle inequality. Then, for any ,*

$$\mathcal{B} \qquad \mathcal{B}\,\mathcal{B} \quad \mathcal{B} \tag{6}$$

Theorem 5 says that Algorithm 2 is locally a 2-approximation algorithm. Further, the constant cannot be improved as we shall show formally using the example in Figure 2. In this example, the value of is either , , or . The eight solid lines have cost , the four dashed lines have cost , the two dotted lines have cost , and all other costs are . The optimal blocking is which has the corresponding maximum cost in any block . Algorithm 2 creates the blocking . This is because it first creates the partition which minimizes its objective for pairs. Note that, (i) the left hand side of (6) for this example is , and (ii) the optimal blocking is in . Thus, in (6), equality holds for , if 2 is replaced by . Since,

is arbitrary, this example shows that for and the constant in Theorem 5 is the best possible. The example in Figure 2 can be extended to larger block sizes by adding more nodes to the structure and to larger number of blocks by introducing the structure in different positions in the space. Section 5.2 discusses local improvement of Algorithm 2.

5.2. *An extension and local improvements.* In a paired study design the difference between the units in a pair is the cost of the block. In a block with units, there are pairwise differences. It might make sense in practice to define the cost of the block as the average of these differences. The blocking problem analogous to problem (1) for average in-block differences is to find a blocking $\mathcal{B}$ that minimizes

$$\mathcal{B} \quad — \tag{7}$$

We propose to solve this problem by modifying Algorithm 2 that changes the update operation for the cost function from (3) to $\mathcal{V}$. Let be the value of (7) for $\mathcal{B}$, the blocking created by this modification of Algorithm 2.

***Theorem* 6.** *Suppose, the cost function satisfies the triangle inequality: for any and . Let be the smallest value of* (7) *attained by a design. Then,* , *where* .

***Remark* 5.** The supplement gives a more precise expression of the approximation factor in Theorem 6. When the approximation factor is , which is smaller than , the approximation factor in Theorem 3. But for larger the factor in Theorem 6 for (7) is larger than that in Theorem 3 for (1). Hochbaum and Shmoys (1986) make similar observation while comparing average versus maximum objectives.

In the following we consider further local improvements of Algorithm 2 for problem (1) and improvements of Algorithm 2's modification for problem (7). In Figure 2, it is not difficult to consider a few extra steps to improve the output of Algorithm 2, $\mathcal{B}$, to the optimal blocking, $\mathcal{B}$. These steps could be as follows, (i) take out the two units, from and from , by redefining and , (ii) define the costs and , and similarly define and , (iii) rematch, to that minimizes the maximum cost. More formally, in last step we find the onto function that minimizes . Step (iii) then chooses and , since the corresponding maximum cost is . In the following we conform steps (i)–(iii) into an algorithm for improving a blocking $\mathcal{B}$.

First recognize that step (iii) finds a perfect bipartite matching on the bipartite graph whose two parts are and . Consider any bipartite graph with parts and , with . Suppose, the cost on the edge for and is . Then, the formulation of step (iii) involves finding a perfect bipartite matching, , that solves

$$\text{is onto} \quad — \tag{8}$$

To find this efficiently, one may use either of the two algorithms proposed by Derigs (1984).

Algorithm 3. Find the cost of blocking $\mathcal{B}$, say .
while: TRUE:
$\overline{\mathcal{B}}$; $\mathcal{B}$.
do (i)
Select .
; $\overline{\mathcal{B}}$ $\overline{\mathcal{B}}$; $\mathcal{B}$ $\mathcal{B}$.
(i) **continue**.
(ii) Define between $\mathcal{B}$ and $\overline{\mathcal{B}}$.
do (iii)
Find that solves (8) with $\mathcal{B}$ and $\overline{\mathcal{B}}$.
(iii) Redefine , for in $\mathcal{B}$.
Find the cost of the blocking $\mathcal{B}$, say .
if :
break.
else: Redefine $\mathcal{B}\mathcal{B}$; .
end while.
Return $\mathcal{B}$.

Algorithm 3 works as a template for improving the output of Algorithm 2 for either problem (1) or (7). It repeats steps (i)–(iii) as long as it sees there is an improvement. The function in step (i) of the algorithm is the 'maximum' function for the first problem and is the 'average' function for the second problem. We define , in step (ii), for the first problem as and for the second problem as . Finally, and for the corresponding blockings in Algorithm 3 are calculated according to (1) or (7), for the two problems respectively.

The simulation section §7.2 (and additional results in the supplement) shows that the improvement by Algorithm 3 is beneficial.

5.3. *Blocking with blocks of size* . We can use Algorithm 2 to create a blocking where may be not equal to . This is achieved by including additional chameleon units or fake units in $\mathcal{V}$. Additional units such as these are often used to create flexible designs in observational studies, for example, optimal subset selection in a pair matching (Rosenbaum, 2012), near fine balancing (Yang et al., 2012) and matching a treated unit to a variable number of controls in different strata (Lu et al., 2011; Pimentel et al., 2015a).

Recall that Algorithm 2 calls Algorithm 1 times and after each call it updates the vertex set $\mathcal{V}$ of $\mathcal{V}$. Suppose there are units and where is either 0 or 1. Let . Suppose are such that . In that case we include types of chameleon units to $\mathcal{V}$, one type after each of the th, , and th call to Algorithm 1. More specifically, additional chameleon units of type are introduced to $\mathcal{V}$ as new vertices after the th call to Algorithm 1. The difference between two chameleon units of the same type is set to and the difference between a chameleon unit and any of the study units or any other type of chameleon units is set to 0. Algorithm 2 modified in this way creates blocks, where in each block there are chameleon units. Post

Algorithm 2 we remove all these chameleon units to create the blocking of units into blocks of size . Algorithm 3 can then be used to find an improved blocking.

To illustrate this method suppose and there are units. Algorithm 2 creates blocks of size by including chameleon units of the same type, say , at the start of the algorithm. The cost function is extended as and , for and , . When , more chameleon units of a new type are included after the study units and the units have been paired in the first loop Algorithm 2. Call these additional units . We extend the definition of the cost function as , , and , for , and , . When only the second type of chameleon units are used by the method.

6. **Analysis of a synthetic data set**

6.1. *Data description and efficiency gain in hypothesis testing.* Using a study on the effect of Morphine on mental activity score, we compare two paired randomization studies. The first creates pairs by minimizing the average of the paired differences of the covariates using MAPD, and the second creates pairs by minimizing the worst of paired differences of the covariates using Algorithm 1.

We use a synthetic data set created from the data provided by Smith and Beecher (1962). This data set was also analyzed by Snedecor and Cochran (1967) to illustrate regression adjustment. The details of the synthetic data are given in a remark later. Briefly, we created 100 individuals by sampling with replacement from the 24 subjects in the original study. In this study, there are two covariates, and , two measurements of mental activity scores of the study individuals; see Smith and Beecher (1962) for details. Both MAPD and Algorithm 1 use the same cost function, which is the rank based robust Mahalanobis distance of the two covariates (Rosenbaum, 2010, §8.3). In the respective designs, the average of the paired differences are 0.15 for MAPD and 0.17 for Algorithm 1; the largest of the pair differences are 1.72 for MAPD and 1.27 for Algorithm 1.

In each of these two sets of 50 pairs, the treatment, Morphine use, was randomized to one subject within each pair. The control unit is given a placebo. We analyze the null hypothesis of no effect twice for each design, once by testing the symmetry of treated minus control differences in the mental activity scores using the paired randomization test and also by testing the symmetry of treated minus control differences in the regression adjusted mental activity scores using the paired randomization test. In our tabulation of the results, we fix an additive effect of Morphine use (a decrease in mental activity score) and calculate the proportion of times a method rejects the null hypothesis of no effect of Morphine use in 4000 simulations, each simulation creating a paired randomized treatment assignment.

The results are given in Table 2. The power of detection of a treatment effect is consistently higher using Algorithm 1, for every simulated value of the treatment effect. Notably for a treatment effect of 0.5 the power of the paired design of Algorithm 1 is 44% higher (vs) than the paired design of MAPD and 143% higher (vs) than the completely randomized design; for treatment effect of 1, these numbers are 20% and 84% respectively.

In Table 2 a regression adjustment increases the power compared to the usual randomization test for both paired and the completely randomized designs. Results in Table 2 show

TABLE 2. The proportion of rejects of the null hypothesis of no effect for different additive treatment effects; based on 4000 randomized designs.

Treatment effect	Randomization test			Regression adjustment + Randomization test		
	Completely randomized design	Paired design		Completely randomized design	Paired design	
		MAPD	Algorithm 1		MAPD	Algorithm 1
0	0.05	0.05	0.05	0.05	0.05	0.05
0.5	0.12	0.24	**0.37**	0.16	0.27	**0.39**
0.75	0.22	0.47	**0.67**	0.31	0.53	**0.71**
1	0.36	0.75	**0.93**	0.51	0.78	**0.94**
1.25	0.54	0.91	**0.99**	0.70	0.93	**1.00**

TABLE 3. Root Mean Squared Error (RMSE) for estimating the ATE in synthetic data from the Morphine study. Lowest RMSE in each row is in bold.

	ATE	Algorithm 1	MAPD w/ caliper	MAPD	Completely randomized design
Model 1	0.25	**0.58**	0.62	0.62	0.78
Model 2	0.49	**1.02**	1.05	1.05	1.14
Model 3	0.80	**0.89**	0.92	0.92	1.03

that gain due to regression adjustment is much higher for completely randomized design (a 40% increase in power at effect size 0.75) than for the MAPD based paired design (a 13% increase in power at effect size 0.75) and the smallest for the Algorithm 1 based paired design (a 6% increase in power at effect size 0.75). This is consistent with findings related to treatment effect estimation from randomized designs that efficiency gain due to regression adjustment is higher when the design has been less than adequate to balance the covariate (Lin, 2013). But, here, MAPD based design with regression adjustment still has lower power than Algorithm 1 based design without regression adjustment.

***Remark* 6. Details of the synthetic data.** For the above analysis we simulated 100 individuals based on the data on 24 subjects in Smith and Beecher (1962). Increasing the sample size helps in presenting moderate power of both methods for the treatment effects considered. To create the 100 individuals we simulated 60 subjects with replacement from 20 subjects with , the rest were simulated with replacement from the other 4 subjects. This was done to slightly exaggerate the variability of the distribution of the covariates; variance of for the 24 subjects is and the variance is for the simulated 100 subjects, similarly for , the variance is increased from in for the 24 subjects to for the 100 subjects (see Remark 2). For each subject, the original study recorded the mental activity score two hours after a placebo treatment. We used these scores as control potential outcomes. The treated potential outcomes for Table 2 and Table 3 (discussed below) are generated following the treatment effect model specified.

6.2. *Comparison of accuracy in treatment effect estimation.* Based on the data from the above synthetic example we next consider estimation of the average treatment effect. We compare four designs. The first two are paired designs of Algorithm 1 and MAPD as before. The third method uses MAPD to create a blocking (i.e., pairing) but uses a caliper on the distance, , so that the MAPD algorithm is not allowed to form dissimilar pairs while minimizing the average in-block weights (Rosenbaum, 1989; Rubin and Thomas, 2000; Austin, 2011). The caliper value is chosen as 1.6 so that the most extreme pair is disallowed in the design. The final design is the completely randomized design.

We consider three treatment effect models in our comparison. These are labelled as Model 1, Model 2 and Model 3 respectively in Table 3. Model 1 is a constant treatment effects model where Morphine use decreases mental activity score by 0.25 for every individual in the study. Thus, the average treatment effect (ATE) for Model 1 is . In Model 2 Morphine use decreases mental activity by an amount of to a minimum of 0. Recall, is a baseline mental activity score. Hence, Model 2 implies that Morphine use decreases mental activity score more according to higher baseline activity of the individual. The ATE for Model 2 is . Model 3 assumes that decrease in mental activity score due to Morphine use is . The ATE for Model 3 is 0.80.

In Table 3, Algorithm 1 gives the smallest RMSEs for estimating the ATE using the unbiased difference-in-means estimator for these three models for each model. Also, caliper here did not improve the RMSEs at two decimal points.

7. Comparison of accuracy in treatment effect estimation: simulation studies

In this section we present simulation studies to evaluate treatment effect estimation accuracy of the difference-in-means estimator in blocked designs using the proposed methods and compare the performances of the proposed methods to their competitors.

7.1. *Paired randomization.* We studied paired designs in §6 using a synthetic data set. Using simulated data sets we again consider paired designs where a blocking corresponds to pairs. We compare Algorithm 1, MAPD and also a modification of the MAPD algorithm that uses a caliper to discourage pairing of very dissimilar units. The caliper is set so that the extreme pair is not allowed when using MAPD matching.

Simulation scenario 2 from §4.3 is used to generate the covariate data (scenario 1 is not used since the cost function is not associated with any covariate). Next, the control potential outcome for all the models is where is drawn i.i.d. from the standard normal distribution. The treatment potential outcomes for the four outcome models are for Model 1, for Model 2, for Model 3, and for Model 4.

We considered 100 simulated data sets. In each data set we used the 3 blocking methods to create paired designs. Then for each paired design we calculated the RMSE of estimating the ATE using the difference-in-means estimator in that dataset by considering 10000 randomizations. The RMSE values in Table 4 are the average RMSEs across different data sets. Table 4 also shows the ATEs for the models. As the baseline for the comparison we include the completely randomized design.

TABLE 4. Root Mean Squared Error (RMSE) for estimating the ATE in paired designs. Lowest RMSE in each row is in bold.

Outcome Model	ATE	Algorithm 1	MAPD w/ caliper	MAPD	Completely randomized design
Model 1	0.50	**2.49**	2.63	2.65	6.34
Model 2	0.99	**2.55**	2.69	2.71	6.57
Model 3	1.81	**2.51**	2.64	2.66	6.42
Model 4	2.90	**2.55**	2.69	2.71	6.61

In Table 4, the proposed algorithm consistently provides more accurate estimate of the ATE compared to any other method. Using caliper on the cost function only slightly improves the accuracy in treatment effect estimation from a MAPD based design.

7.2. *Blocked randomization with and four treatments.* Here we consider a third simulation scenario with and . Simulation scenario 3 has two binary and four continuous variables. To simulate such a structure, a 6 dimensional random vector is simulated from a mixture of three multivariate normal distributions with the same equi-correlation matrix of correlation with mixing proportions , and respectively. The normal mean vectors for the three components are , , and , respectively. Finally, the first two values of the vector, say and , are discretized as: if and if , and 0 otherwise. The cost function used is the square root of rank based robust Mahalanobis distance.

We consider a blocked randomization design that assigns 4 treatments randomly within each block. We compare six blocking methods. The first four methods are: Algorithm 2 with and without its improvement using Algorithm 3 for the two problems (1) and (7). Next, the 'Random templating and assignment' method is the randomized algorithm of Karmakar (2018) that creates many blockings by randomly selecting one template unit for each block, and outputs a blocking that is the best among those. The last method is the 'optimal greedy' heuristic blocking method of Moore (2012).

Each unit has four potential outcomes for and let denote the sample ATE of treatment over . We consider:

Outcome Model 1: In this model where is drawn i.i.d. from the standard normal distribution. The other potential outcomes are and

Outcome Model 2: In this model, same as before, where is drawn i.i.d. from the standard normal distribution. The other potential outcomes are and Recall, and are binary variables. So, a treatment changes the outcome only for some units, but when it does change the outcome, the amount of change is the same (3 units).

Results of our simulation comparison are given in Table 5. To calculate this table, we considered the difference-in-means estimator in 100 simulated data sets under Scenario 3. For each data set, the RMSE values are calculated based on 10000 randomizations for each

TABLE 5. Root Mean Squared Error (RMSE) for estimating the sample ATE in a blocked randomized design with 50 blocks of size 4. Four treatments are randomized in each block. Lowest (best) RMSE in each column is in bold.

	Outcome Model 1			Outcome Model 2		
ATE	-1.516	1.394	1.394	0.593	1.620	1.372
Completely randomized design	1.823	1.941	1.974	1.905	1.876	1.937
Algorithm 2 for (1)	1.412	1.373	1.347	1.448	1.403	1.372
Algorithm 2+3 for (1)	**1.340**	**1.164**	**1.290**	**1.366**	1.179	1.302
Algorithm 2 for (7)	1.484	1.436	1.411	1.442	1.399	1.370
Algorithm 2+3 for (7)	1.404	1.206	1.349	1.367	**1.175**	1.303
Random templating and assignment	1.405	1.438	1.417	1.375	1.410	1.378
Heuristic blocking	1.487	1.208	1.354	1.454	1.182	**1.300**

blocking method. The reported RMSE values are the average RMSEs across the data sets. Table 5 also shows the average sample ATE across the data sets.

We did not consider the threshold blocking method of Higgins et al. (2016) in this simulation study because this method does not create equal sized blocks such that we can use a randomized design with 4 treatments randomly assigned to 4 units in each block.

The simulation results in Table 5 show that all blocking strategies provide efficiency gain over the completely randomized design. The proposed methods give more accurate estimates of the ATEs. Finally, Algorithm 3 provides robust improvement over Algorithm 2.

8. Conclusion

The methods presented in this paper can be used to create blocked randomized designs, for any block size, that blocks on many covariates. These methods require that the difference, according to those covariates, of any two units be written as a cost function. Some experimental designs may require solutions of other blocking problems which impose additional constraints. One such constraint could state that the value of a covariate must be within a given range in each block. It would be possible to solve this problem using the proposed methods by appropriately defining the cost function, e.g., by using a caliper for the range of the covariate. But there could be other constraints that are more difficult to incorporate.

Acknowledgment

The work was supported by a grant from the National Science Foundation, USA. The author thanks the Editor, the Associate Editor and the three referees for their detailed comments and suggestions which led to significant improvement of the paper.

References

Austin, P. C. (2011). Optimal caliper widths for propensity-score matching when estimating differences in means and differences in proportions in observational studies. *Pharmaceutical Statistics*, **10**, 150–161.

Bailey, R. A. (1987). Restricted randomization: A practical example. *J. Am. Statist. Ass.*, **82**, 712–719.

Beck, C., Lu, B. and Greevy, R. (2016). nbpMatching: Functions for optimal non-bipartite matching. *CRAN*, package version 1.5.1.

Bondy, J. A. and Murty, U. S. R. (1976). *Graph Theory with Applications*. New York: Elsevier.

Box, G. E.P. and Hunter, J. S. (1961). The fractional factorial designs, *Technometrics*, **3**, 311–351.

Cheng, C. S. and Mukerjee, R. (2001). Blocked regular fractional factorial designs with maximum estimation capacity. *Annals of Statistics*, **29**, 530–548.

Cochran W. G. (1965). The planning of observational studies of human populations. *J. R. Statist. Soc. A*, **128**, 234–266.

Cox, D. R. (1958). *Planning of Experiments*. Oxford, England: Wiley.

Cox, D. R. and Reid, N. (2000). *The Theory of the Design of Experiments*. Chapman & Hill/CRC.

Derigs, U. (1984). Alternate strategies for solving bottleneck assignment problems – Analysis and computational results. *Computing*, **33**, 95–106.

Derigs, U. (1988). Solving nonbipartite matching problems via shortest path techniques. *Ann. Oper. Res.*, **13**, 225–261.

Ellickson, P. L., McCaffrey, D. F., Ghosh-Dastidar, B. and Longshore, D. L. (2003). New inroads in preventing adolescent drug use: Results from a large-scale trial of Project ALERT in middle schools. *Am. J. Public Health*, **93**, 1830–1836.

Federer, W. T. and King, F. (2007). *Variations on split plot and split block experiment designs*. John Wiley & Sons, Hoboken, NJ.

Fisher, R. A. (1935). *The Design of Experiments.* Edinburgh: Oliver and Boyd.

Flay, B. R., Brannon, B. R., Johnson, C. A., Hansen, W. B., Ulene, A. L. and Whitney-Saltiel, D. A. (1988). The television, school and family smoking prevention and cessation project. I Theoretical basis and program development. *Prev. Med.*, **17**, 585–607.

Fogarty, C. (2018). On mitigating the analytical limitations of finely stratified experiments. *J. R. Statist. Soc. B*, **80**, 1035–1056.

Gabow, H. N. and Tarjan, R. E. (1988). Algorithms for two bottleneck optimization problems. *Journal of Algorithms*, **9**, 411–417.

Greevy, R., Lu, B., Silber, J. H. and Rosenbaum, P. R. (2004). Optimal multivariate matching before randomization. *Biostatist.*, **5**, 263–275.

Hansen, B. B. and Klopfer, S. O. (2006). Optimal full matching and related designs via network flows. *J. Comp. Graph. Statist.*, **15**, 609–627.

Higgins, M. J., Sävjeb, F. and Sekhon, J. S. (2016). Improving massive experiments with threshold blocking. *Proc. Natl. Acad. Sci.*, **113**, 7369–7376.

Hochbaum, D. S. and Shmoys, D. B. (1986). A unified approach to approximation algorithms for bottleneck problems. *J. ACM*, **33**, 533–550.

Hu, W., Chan, C. W., Zubizarreta, J. R. and Escobar, G. J. (2018). Incorporating longitudinal comorbidity and acute physiology data in template matching for assessing hospital quality: an exploratory study in an integrated health care delivery system. *Med. Care.*, **56**, 448–454.

Karmakar, B. (2018). blockingChallenge: Create blocks or strata which are similar within. *CRAN*, R package version 1.0.

Kernan, W. N., Viscoli, C. M., Makuch, R. W., Brass, L. M. and Horwitz, R. I. (1999). Stratified randomization for clinical trials. *J. Clin. Epidemiol.*, **52**, 19–26.

Kelcey, B., Spybrook, J., Phelps, G., Jones, N. and Zhang, J. (2017). Designing large-scale multisite and cluster-randomized studies of professional development, *THe Journal of Experimental Education*, **85**, 389–410.

Kirkpatrick, D. G. and Hell, P. (1978). On the completeness of a generalized matching problem. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pp. 240–245.

Lin, W. (2013). Agnostic notes on regression adjustments to experimental data: Reexamining Freedman's critique. *Ann. Appl. Stat.*, **7**, 295–318.

Lu, B., Greevy, R., Xu, X. and Beck, C. (2011). Optimal nonbipartite matching and its statistical applications. *Am. Statist.*, **65**, 21–30.

Moore, R. T. (2012). Multivariate continuous blocking to improve political science experiments. *Political. Anal.*, **20**, 460–479.

Moore, R. T. and Schnakenberg, K. (2016). blockTools: Block, Assign, and Diagnose Potential Interference in Randomized Experiments. *CRAN*, package version 0.6-3.

Morgan, K. L. and Rubin, D. B. (2012). Rerandomization to improve covariate balance in experiments. *Ann. Stat.*, **40**, 1263–1282.

Moulton, L. H. (2004). Covariate-based constrained randomization of group-randomized trials. *Clinical Trials*, **1**, 297–305.

Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial Optimization: Algorithms and Complexity.* New York: Dover.

Pimentel, S. D., Kelz, R. R., Silber, J. H. and Rosenbaum, P. R. (2015). Large, sparse optimal matching with refined covariate balance in an observational study of the health outcomes produced by new surgeons. *J. Am. Statist. Ass.*, **110**, 515–527.

Pimentel, S. D., Page, L. C., Lenard, M. and Keele, L. (2018). Optimal multilevel matching using network flows: An application to a summer reading intervention. *Ann. Appl. Statist.*, **12**, 1479–1505.

Robinson, J. (1970). Blocking in incomplete split plot designs, *Biometrika*, **57**, 347–350.

Rosenbaum, P. R. (1989). Optimal Matching in Observational Studies. *J. Am. Statist. Ass.*, **84**, 1024–1032.

Rosenbaum, P. R. (2010). *Design of Observational Studies*. New York: Springer.

Rosenbaum, P. R. (2012). Optimal matching of an optimally chosen subset in observational studies. *J. Comp. Graph. Statist.*, **21**, 57–71.

Rosenbaum, P. R. (2017). Imposing minimax and quantile constraints on optimal matching in observational studies. *J. Comp. Graph. Statist.*, **26**, 66–78.

Rubin, D. B. (1979). Using multivariate matched sampling and regression adjustment to control bias in observational studies. *J. Am. Statist. Ass.*, **74**, 318–328.

Rubin, D. B. and Thomas, N. (2000). Combining propensity score matching with additional adjustments for prognostic covariates. *Journal of the American Statistical Association*, **95**, 573–585.

Silber, J. H., Rosenbaum, P. R., Ross, R. N., Ludwig, J. M., Wang, W., Niknam, B. A., Mukherjee, N., Saynisch, P. A., Even-Shoshan, O., Kelz, R. R. and Fleisher, L. A. (2014). Template matching for auditing hospital cost and quality. *Health Serv. Res.*, **49**, 1446–1474.

Smith, G. M. and Beecher, H. K. (1962). Subjective effects of heroin and morphine in normal subjects. *J. Pharmacol. Exp. Ther.*, **136**, 47–52.

Snedecor, G. W. and Cochran, W. G. (1967). *Statistical methods*, 6th edn., Ames., Iowa State College Press, IA.

Williamson, D. P. and Shmoys, D. B. (2011). *The Design of Approximation Algorithms.* Cambridge University Press.

Yang, D., Small, D. S., Silber, J. H. and Rosenbaum, P. R. (2012). Optimal matching with minimal deviation from fine balance in a study of obesity and surgical outcomes. *Biometrics*, **68**, 628–636.

Zubizarreta, J. R. (2012). Using mixed integer programming for matching in an observational study of kidney failure after surgery. *J. Am. Statist. Ass.*, **107**, 1360–1371.

Zubizarreta, J. R. and Keele, L. (2017). Optimal multilevel matching in clustered observational studies: a case study of the effectiveness of private schools under a large-scale voucher system. *J. Am. Statist. Ass.*, **112**, 547–560.

226 Griffin-Floyd Hall, Department of Statistics, University of Florida, Gainesville, FL 32611, USA. Email: bkarmakar@ufl.edu

Supplement: An approximation algorithm for blocking of an experimental design

Bikram Karmakar
Department of Statistics, University of Florida, Gainesville, FL 32611, USA.

1 Proof of Theorem 1

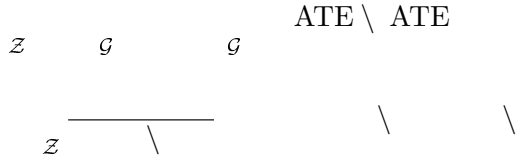
We first show that

(S1)

Start form the left hand side

Part (a).

If then using identity (S1)



Part (b).

We consider where is a fixed vector. We first check that this is in $\mathcal{G}$. We use the shorthand . Fix and . Without loss of generality assume .

\ \

\

\

The first inequality follows by triangle inequality and the second inequality follows since .

Consider values that are either or \ . There are 0s and \ . Verify that these set of values control the average distance. Using triangle inequality

\ \

\

\ \

Throughout we let \ .

Let $\mathcal{L}$ be the set of units where the treatment is assigned freely. Let $\mathcal{L}$ $\mathcal{L}$ be the set of units that are assigned treatment and let $\mathcal{L}$ $\mathcal{L}$ $\mathcal{L}$. Let $\mathcal{L}$ $\mathcal{L}$ $\mathcal{L}$, where $\mathcal{L}$ is the subset of $\mathcal{L}$ corresponding to . By (S1)

ATE \ ATE \ \

$\mathcal{L}$ $\mathcal{L}$ $\mathcal{L}$ $\mathcal{L}$

$\mathcal{L}$ \ \ $\mathcal{L}$ \

\ \

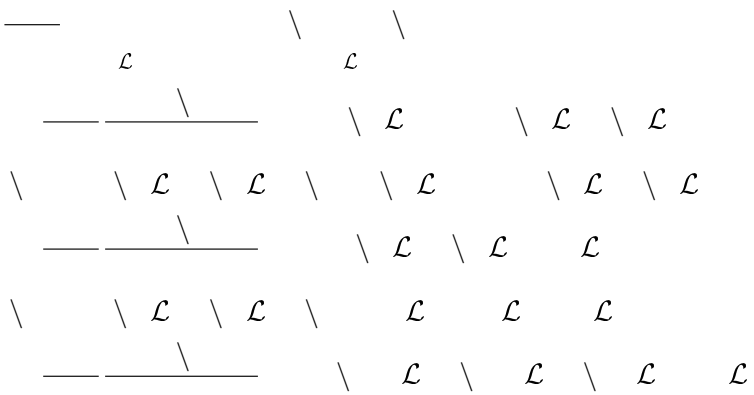
$\mathcal{L}$ $\mathcal{L}$

\ $\mathcal{L}$ \ $\mathcal{L}$

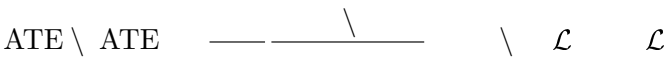
\ \

$\mathcal{L}$ $\mathcal{L}$

Among the $\mathcal{L}$ units, assign the rest possible untis to the treatment. We note that $\backslash$ $\mathcal{L}$ units remain to be treated, and $\backslash$ $\mathcal{L}$ $\backslash$ $\mathcal{L}$ of the units with $\backslash$. Thus, $\backslash$ $\mathcal{L}$ $\backslash$ $\mathcal{L}$ $\backslash$ $\mathcal{L}$ of the remaining units with $\backslash$ in $\mathcal{L}$ are assigned treatment. The other $\backslash$ $\mathcal{L}$ $\backslash$ $\mathcal{L}$ $\backslash$ $\backslash$ $\mathcal{L}$ $\backslash$ $\mathcal{L}$ $\backslash$ $\mathcal{L}$ units from $\mathcal{L}$ with $\backslash$ are assigned to control. Thus,



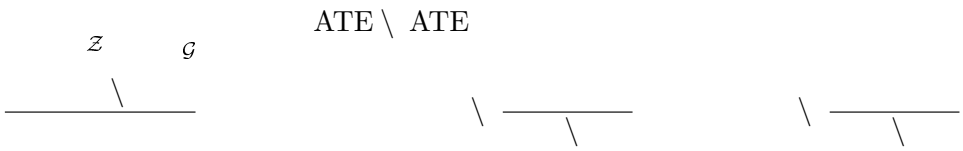
Thus, we get



By definition of the set $\mathcal{L}$'s the restriction on the sets $\mathcal{L}$ and $\mathcal{L}$ is that their sizes are at most . Hence the proof.

This inequality can be shapened by considering the other assignment where the remaining possible control units are from the remaining units with $\backslash$. The sharpened lower bound is (i) if

(ii) if



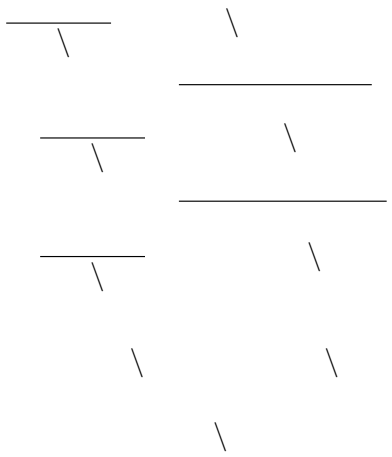
We now consider a slight modification of the inequality. Assume that s are univariate and is the absolute distance. Fix small, a point and pick rather than being exactly at be in the interval $\backslash$. Pick the other points in the interval $\backslash$ $\backslash$ $\backslash$. It is easy to verify that $\backslash$. Thus, using triangle inequality the average pairwise distance between any two points is

Notice that the difference between the two sets of points is at least \ \ \ . Next, as before let . But now notice that the error in ATE estimation is at least as much as our calculation before except now previous imbalance \ replaced by \ \ \ . Hence we get the proof of the Theorem.

Lemma S1 *This lemma gives a lower bound for the conditional probability of getting a set of 's so that the lower bound of part (b) of Theorem 1 is attained given that the average pairwise distance is at most when the \ and 's are sampled i.i.d. from a standard normal distribution.*

Proof. We assume that are sampled randomly from the standard normal distribution. We want to find the conditional probability that we observe of the units are in an interval of length and the other units are also in another interval of length , and the maximum distance between any two points is at most \ given that the average pairwise distance of any pointst is at most

First we start by computing the denominator of the conditional expectation.



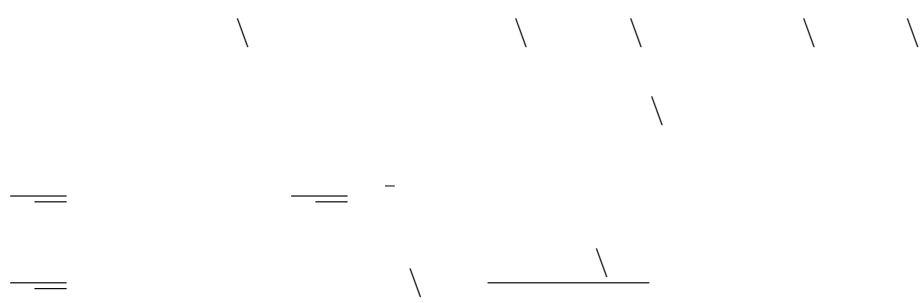
Now consider the numerator which is the probability of observing random samples of a particular pattern. We calculate the probability that are in an interval of length and the other units are in another interval of length , and the maximum distance between any two points is at most \ . Then the numerator will be times this probability.

The probability in question is calculated in four parts: (a) if all the points are in the positive real line, (b) if all the points are in the negative real line, (c) the first set of points are negative and the second set of points are positive, and (d) the first set of points are postive and the next set of points are negative,

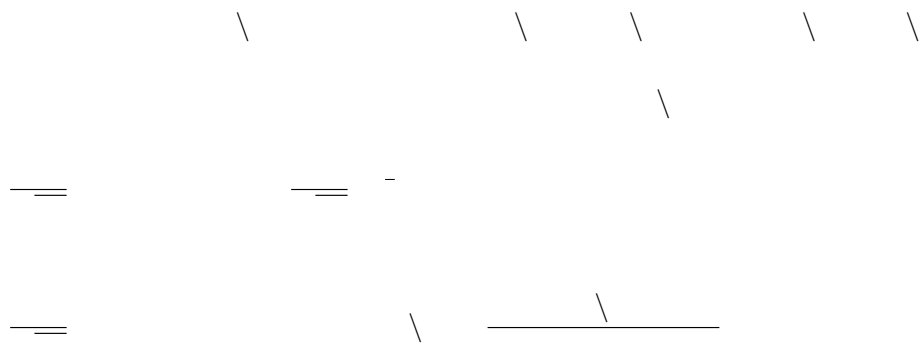
Notice that probability of (a) and (b) are the same since the distribution is symmetric around 0. Let's consider (a) then. We can split (a) in two parts: (a1) points come first

and then the points, and (a2) the points come first then the points. Throughout we use and denote the standard normal density and distribution function respectively.

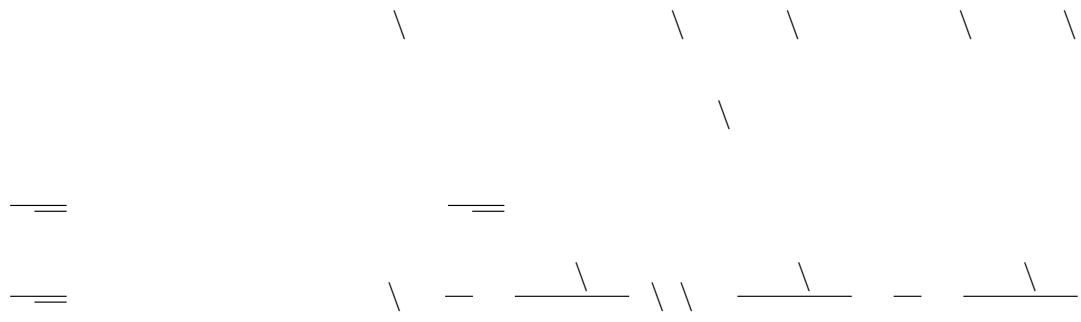
We start by calculating probabilities for (a1) and (a2). Probability of (a1) is



Probability of (a2) is



It remains to calculate the lower bounds of the probabilities of (c) and (d).



Finally, we have the probability of (d)

Finally, we get the lower bound of the conditional probability by adding the lower bounds for (a1), (a2), (b), (c) and (d) and then diving it by .

We can calculate this lower bound numerically. Using this lemma we calculate, when , and the probability is 3.2%, when , and the probability is at least 5%.

2 Proof of Theorem 2

(a) We first see the only-if part of the theorem. If is a matching and it permits a negative alternating cycle, , then is another matching and . Thus, cannot be an optimal matching.

Now we prove the if part of the statement. Suppose is a matching that does not permit any negative alternating cycle. We start by observing that any other matching of the same set of matched vertices can be created from by composing it with alternating cycles. More specifically, if is any other matching on the same set of vertices as and , then there are non-overlapping alternating cycles with respect to so that .[1] Since, does not permit any negative alternating cycles, we have .

We write, , where we denote by . Also, . Let , and . Thus implies, either , or . Further,

[1] Note though that it is possible for two maximal matchings and for the complete graph, for any alternating cycle , that is, for two matchings on the same set of vertices composition with more than one alternating cycles may be necessary to transform one matching to the other.

Now,

(S2)

For any , we have either or . In the second case, we also have, . Hence, in this case, . Using these relations, thus, in (S2) we have

\

This completes our proof of part (a).

(b) Let be a matching of size that permits no negative alternating cycle. Let be a shortest augmenting path of , and define .

The proof is by contradiction. If possible suppose, is a negative alternating cycle of , i.e., is an alternating cycle with respect to the matching and .

First consider the case when \ . Note that

(S3)

Where we use the notation to shorthand . When \ , we have the following

\ \
\ \ \

\ \
\
\ \
\

Where we have used De-Morgan's law and the distributive law for the second and the third

equality respectively. Also

\ \
\ \
\

Rewrite (S3) as

(S4)

Note that, as \

\ (S5)

Also

\ (S6)

and

(S7)

Since is a negative alternating cycle with , from (S4) and (S7)

(S8)

This implies,

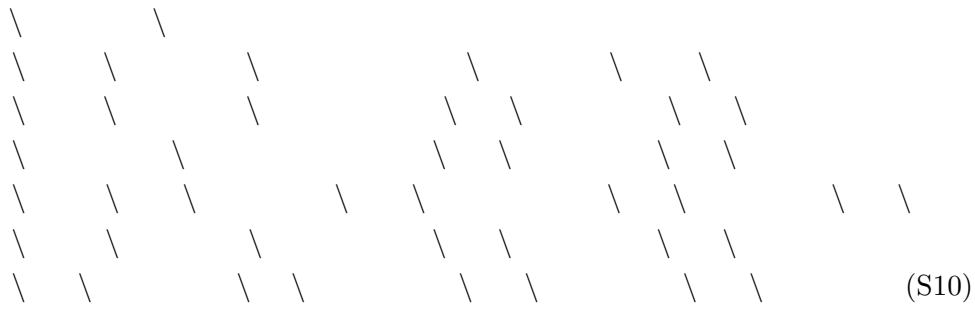
\ \

The left hand side term above is at least by (S5) and the right hand side term is by (S6). Thus . Which gives us a contradiction since, by our assumption, does not permit any alternating cycle.

Now consider the other case, \ . Define for this case \ \ . First we observe that . To see this, write

\ \
\ \ \ \ \ \ \
\ \ \ \ \ \
\ \ \ \ \ \ \ \
\ \ \ \ \ \ \
\ \ \ \ \ \ \ \ (S9)

Also,



(S10)

Notice that (S9) and (S10) are identical. As is a negative alternating cycle of we have

Finally, we shall show that is an augmenting path with respect to that starts at , i.e., . This will contradict the fact that is the minimal augmenting path with respect to .

We first argue that starts with . Let , an unmatched edge of . Consequently, is a matched edge in the augmented match . Also, no other edge in includes . There are now two possibilities, either there is an edge in that includes , or there is not. If there is no such edge, then . Otherwise, , and since is an alternating cycle with respect to , there is an edge in where is matched in and in . Thus, starts at . The same argument shows that if ends at , which is an unmatched node in then also ends at .

Lastly, we show that is an augmenting path with respect to . To show this, it suffices to show that if then (i) both and are matched in , else one is matched in and the other is unmatched in , i.e., or , and (ii) when and are not and , there is exactly one and one , both of which are matched nodes in so that and .

We argue in two cases, and assume that neither of is or .

Case 1; suppose but . Consider two sub-cases. First, suppose . Then, as is an alternating cycle, neither or are in . Also, the nodes and are not in . Hence, and are in . Next, suppose, . Since , . If possible, suppose and . One is matched in and the other is unmatched in . Wlog, suppose is matched in . As is the augmented match , must be in . Which implies that . So, and . Similarly, there is exactly one edge in , other than with the node .

Case 2; but . Since, is a cycle, there is exactly one and one so that and . At most one of them are in . For, implies either is a matched edge in both and , or it is an unmatched edge in both and . We consider these two sub-cases separately. If is matched in both and , neither nor are in . Because, if say was in , as

it is an unmatched edge in , it would be matched in . But, as is an augmenting path, there is also another such that is in and matched in . Which is a contradiction to the fact that the matched edge of in is not in .

Now suppose is unmatched in both and . Then consider the two edges and . Suppose, if possible, there is an edge in . Then there is also another edge . One of them, say the first one, is matched in and the other is unmatched in . As is an alternating cycle and is unmatched in , it is true that is in . So, and .

Thus, we have established that $\mathcal{P}$, an augmenting path with respect to that starts at . Thus, if is a negative alternating cycle with respect to , we have established a contradiction to the fact that is the shortest augmenting path with respect to by showing .

3 Algorithm to find the shortest augmenting path

In this appendix we present the algorithm of finding the shortest augmenting path $\mathcal{P}$ used in Algorithm 1 of the main text.

1. Input: a set of nodes; the cost function between any two vertices and given by ; node in that is unmatched in the existing match .

2. Initialization: Define vectors, of size each, `l`, `m`, `p`, `p_org`, `dm`, `dp` and `Pseudo`, and a scalar `Bls`. Make a non-redundant copy of as that would be modified within the algorithm. Set

- `l[ ]` = `'uncolored'` for and `l[]` = `'colored'`.
- `m[ ]` = `NULL` for all .
- `p[ ]` = for all ; `p_org` = `p`.
- `dm[ ]` = for all and `dm[]` = `-Inf`.
- `dp[ ]` = `Inf` for all .
- `Pseudo[ ]` = `FALSE` for all .
- `Bls` = .

Interpretation of the variables:

The interpretation of these objects in the algorithm is as follows.

- This algorithm builds a tree rooted at , and stops when it finds another node that is also unmatched in and is the next best option to be added to the tree.
- `l`: As more nodes are added to the tree those nodes are labeled as colored by the vector `l`.

- `p`: In the tree if a node is added before its matched node is added, vector `p` is used to find its parent node in the tree. If a node has not been added to the tree yet, thus `l[ ] = 'uncolored'`, `p[ ]` labels the node in the tree so that if were to be added to the tree attaching it to `p[ ]` would be the best.
- `m`: When a node is added that already has its matched node in the tree, the value of the vector `m[ ]` is used to identify that matched node of . The value of `m[ ]` is `NULL` for other nodes. Specifically, `m[]` = `NULL` and `m[Bls]` = `NULL` (see the final point about `Bls`).
- `dm`: For an uncolored node the value `dm[ ]` stores the best cost if were to be added to the tree at `p[ ]`.
- `dp`: When a node is added to the tree, `dp[ ]` is the best cost of the path traversed from to along the tree.
- `Pseudo`: In the process of building the tree, the algorithm might find a cycle which can be traversed any way with and the cost would remain the same. These would be a cycle of an odd number of edges. The algorithm then collapses the nodes in the cycle to one pseudo-node. When a pseudo-node is created the nodes in it are removed from . Initially, none of the nodes are pseudo nodes; thus, `Pseudo[ ]` = `FALSE` for all .
- `p_org`: `p_org` tracks the parent nodes of in the original nodes.
- `Bls`: `Bls` denotes the node or a pseudo-node in that includes the root node . At anytime, the tree is rooted at `Bls`.

In the following we use the notation to denote .

3. A while(`TRUE`) loop, that exists at .

Define:

`dm[ ]` `l[ ] 'uncolored'`

`dp[ ]` `dm[ ]` `l[ ] 'colored'`

Consider two switch cases — **Case 1:** and **Case 2:** .

First, for these two switch cases set

`dm[ ]` `l[ ] 'uncolored'` when **Case 1**,

or, `dp[ ]` `dm[ ]` `l[ ] 'colored'` when **Case 2**

Now for these two cases we do the following operations.

Case 1:

If is unmatched in then exit the while loop, go to **4.**

Otherwise, do the following.

- Find that is the matched vertex of in .
- Set `l[ ]='colored'`, `m[ ]=` , `dp[ ]` , and `l[ ]='inline'`.
- Check vertex for possible reassignment of `'uncolored'` nodes.
  - Scan for each `l[ ]` `'uncolored'` ,
  - if `dp[]`, `dm[ ]` set `dm[ ] = dp[]`, , and `p[ ] =` .

Case 2: Find two paths, using a backward search method, `Path 1` and `Path 2` in .

- `Path 1` is a path that starts at and ends at `Bls` so that when in the path, for all and `= p[ ]` or `= m[ ]`.
- `Path 2` is a path that starts at `p[ ]` and ends at `Bls` so that, as before, when in the path, for all and `= p[ ]` or `= m[ ]`.

In the path find the vertex so that is in both the paths and no other node that proceeds in `Path 1` is a node that proceeds in `Path 2`. Define a new path that first traverses `Path 1` backwards from to and then jumps to `Path 2` and traverses it forwards from `p[ ]` to the node preceding (so is traversed). Call this path . Notice that can also be thought of as a cycle. Also, starting from any node in we can find an alternating path to that needs an odd number of edges.

The following steps are used to collapse the nodes in into a new pseudo-node called `newnode`. It assigns appropriate values to the vectors `l`, `m`, `p`, `dm`, `dp` and `Pseudo` for the new node. The nodes in are removed from and adds `newnode`.

- Define a new node `newnode` and include it in . Set `B[newnode]` = .
- Remove the nodes in from .
- `l[newnode]` = `'colored'`.
- `m[newnode]` = `m[ ]`.
- `p[newnode]` = `p[m[ ]]`.
- `dm[newnode]` = `Inf`.
- `dp[newnode]` = .
- `Pseudo[newnode]` = `TRUE`.
- If `m[newnode]` = `NULL` set `Bls = newnode`.

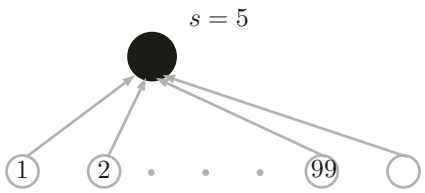


Figure S1: **Initialization.**

- `p[`t`]` = `newnode` for all t so that `p[`t`]` is a node in b.

The final step for **Case 2** is a reassignment step for the `'uncolored'` nodes within the new pseudo-node. The steps, which is reminiscent of the reassignment step at the end of the operations in **Case 1**, runs as follows

- For all $v' \in \{v' \mid v' \in V', \texttt{l[}v'\texttt{]} = \texttt{'uncolored'}\}$ do the following.
  - For $t \in V \cap \{\text{nodes in } b\}$ define $\texttt{dtemp[}t\texttt{]} = \max\{\texttt{dp[newnode]}, c[t, v']\}$.
  - If $dt := \min_{t \in V \cap \{\text{nodes in } b\}} \texttt{dtemp[}t\texttt{]} < \texttt{dm[}v'\texttt{]}$, set
    - $\texttt{dm[}v'\texttt{]} = dt$ and $\texttt{p[}v'\texttt{]} = \texttt{newnode}$.
    - $\texttt{p\_org[}v'\texttt{]} = \arg\min_{t \in V \cap \{\text{nodes in } b\}} \texttt{dtemp[}t\texttt{]}$.

4. Expanding the path from v to s along the tree:

Upon exiting **3** at $(\star)$, the final step of the algorithm is to find the path from v to s in the tree. This can be done in two steps. First, one can find the path from v to `Bls` through the nodes in V'. This is done easily using the vectors `p` and `m`. Second, as many of the nodes in V' might be pseudo-nodes, this path needs to be expanded to a path in V. For the second step one can use a recursive algorithm similar one in Derigs (1988). All the information required are stored in the variables `Bls` (the node/pseudo-node containing s), `p` (the parents in V' for the nodes in the tree), `p_org` (the parents in V for the nodes in the tree), `m` (the matched node for a node added to the tree after), and `B` (the structure of each pseudo-node). The resulting path is the shortest augmenting path $P_0 \in \mathcal{P}_s(M)$ from s to v.

The proposed algorithm has been implemented in `R`. This has been made available through the repository https://github.com/bikram12345k/BlockingAlgo.

3.1 An illustration.

In the following we give an illustration of a possible sequence of finding the shortest augmenting path. The illustration is drawn from a simulated data in an intermediate step of Algorithm 1 when the existing matched structure M has 35 matched pairs.

In Figure S1 the initialization step starts with $s = 5$, which is the root node of the tree. Figures S2 and S3 show the following three steps. Each step correspond to one round of

the while loop in the algorithm. In these two steps the tree grows from the root node to 3 nodes, from 3 nodes to 5 nodes and from 5 nodes to 7 nodes. In all these steps case 1 prevails over case 2. Figure S2, showing sub processes of step 1, highlights the growth of the tree, on the left, and then, on the right, the reassignment of the `'uncolored'` nodes to new colored node $u = 80$. The colored nodes are shown in the filled black circles. The nodes `'inline'` are shown in hollowed black circles. The `'uncolored'` nodes, that are not in the tree, can be attached later only to a `'colored'` node.

Later, in Figure S4, skipping a few steps, we show the tree after Step 7. The next step, Step 8 in Figure S5, processes Case 2 with $v = 44$ and `p[`v`]` $= 42$. `Path 1` is $44 \to 21 \to 84 \to 4 \to 42 \to 33 \to 24 \to 59 \to 80 \to 1 \to 5$, and `Path 2` is $42 \to 33 \to 24 \to 59 \to 80 \to 1 \to 5$. So, $b = 42 \to 4 \to 84 \to 21 \to 44$. In the center figure, the `'uncolored'` nodes are reassigned to the nodes $\{42, 4, 84, 21, 44\}$. These links are stored in `p_org`. Finally, in the right-most figure, the nodes in b are removed from V' and b is shrunk to a new pseudo-node.

The remaining steps of the algorithm are not shown. The algorithm creates eight pseudo nodes. They are `B1` $= 42 \to 4 \to 84 \to 21 \to 44$, `B2` $=$ `B1` $\to 32 \to 8 \to 55 \to 1$, `B3` $=$ `B2` $\to 19 \to 2 \to 12 \to 28$, `B4` $=$ `B3` $\to \to 72 \to 23 \to 50$, `B5` $=$ `B4` $\to 3 \to 39 \to 3 \to 47$, `B6` $=$ `B537` $\to 49$, `B7` $= 24 \to$ `B6` $\to 38 \to 88$, and `B8` $=$ `B7` $\to 34 \to 20 \to 4 \to 53$. The algorithm ends with finding the shortest path along the nodes/pseudo-nodes in V' as $57 \to B8 \to 59 \to 80 \to 1 \to 5$. This path is expanded to find the path along the original nodes V as $57 \to 49 \to 37 \to 21 \to 44 \to 42 \to 33 \to 24 \to 59 \to 80 \to 1 \to 5 = s$. For this expansion, three vectors `m`, `p`, and `p_org` are used. In this specific example these vectors are shown below. Notice that in expanding the path, if a pseudo-node is traversed, it is done in a direction that an even number the nodes/pseudo-nodes, equivalently, an odd number of edges are traversed.

```
 m =
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24   25   26
  NA "52"   NA   NA   NA   NA   NA  "9"   NA "30" "43" "28"   NA   NA   NA   NA   NA   NA   NA "34"   NA   NA   NA "59"   NA   NA
  27   28   29   30   31   32   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48   49   50   51   52
"18"   NA "40"   NA   NA   NA   NA   NA "13"   NA   NA   NA "36"   NA  "7" "33"   NA "21"   NA   NA  "3" "14" "37" "23"   NA   NA
  53   54   55   56   57   58   59   60   61   62   63   64   65   66   67   68   69   70   71   72   73   74   75   76   77   78
"46" "25"  "1"   NA   NA   NA   NA   NA   NA "19"   NA   NA   NA "26"   NA "32"   NA   NA   NA  "6"   NA   NA "45"   NA   NA   NA
  79   80   81   82   83   84   85   86   87   88   89   90   91   92   93   94   95   96   97   98   99  100   B1   B2   B3   B4
"22" "16"   NA   NA   NA  "4"   NA   NA   NA "38"   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA "33" "33" "33" "33"
  B5   B6   B7   B8
"33" "33" "59" "59"

p =
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24   25   26
"B8" "B8" "B8" "B8" "35" "B8" "B8" "80" "B8" "B8" "B8" "80" "B8" "B8" "B8" "56" "B8" "80" "B8" "B8" "B8" "B8" "B8" "27" "79" "79"
  27   28   29   30   31   32   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48   49   50   51   52
"80" "B8" "B8" "B8" "29" "B8" "B8" "B8" "80" "B8" "B8" "B8" "B8" "B8" "35" "80" "B8" "B8" "B8" "B8" "B8" "B8" "B8" "B8" "B8" "B8" "B8"
  53   54   55   56   57   58   59   60   61   62   63   64   65   66   67   68   69   70   71   72   73   74   75   76   77   78
"B8" "B8" "B8"   NA "B8" "66" "80"  "2" "B8" "B8" "11" "66" "79" "B8" "79" "B8" "10" "B8" "27" "B8" "B8" "B8"  "8" "B8" "B8" "79"
  79   80   81   82   83   84   85   86   87   88   89   90   91   92   93   94   95   96   97   98   99  100   B1   B2   B3   B4
"B8" "56" "B8" "10" "B8" "B8" "B8" "54" "66" "B8" "B8" "54" "B8" "75" "41" "B8" "B8" "B8" "B8" "B8" "B8" "B8" "B8" "B8" "B8" "B8"
  B5   B6   B7   B8
"B8" "B8" "80" "80"

p_org =
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24   25   26
 "4" "23" "39" "42" "35" "62" "21" "80" "39" "21" "50" "80" "42" "55" "68" "56" "47" "80" "68" "72" "84" "50" "72" "27" "79" "79"
  27   28   29   30   31   32   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48   49   50   51   52
"80" "21" "24" "12" "29" "84" "24" "44" "80" "32" "21"  "3" "44" "23" "35" "80" "24" "42" "12" "20" "21"  "4"  "4" "42" "28" "50"
  53   54   55   56   57   58   59   60   61   62   63   64   65   66   67   68   69   70   71   72   73   74   75   76   77   78
"21"  "6"  "4"   NA "49" "66" "80"  "2" "19" "12" "11" "66" "79" "19" "79" "55" "10" "21" "27" "62" "28" "47"  "8" "62" "68" "79"
  79   80   81   82   83   84   85   86   87   88   89   90   91   92   93   94   95   96   97   98   99  100
 "4" "56" "88" "10" "42" "24" "20" "54" "66" "24"  "1" "54" "68" "75" "41" "49" "12" "47" "68" "20" "36" "47"
```

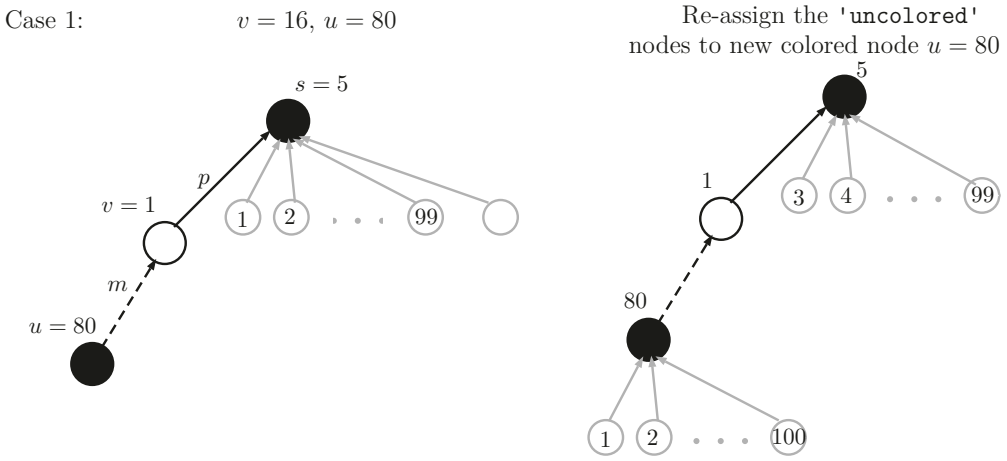


Figure S2: **Step 1.** The `'colored'` nodes are shown in solid black circle. The `'inline'` nodes are shown in hollowed black circle. The `'uncolored'` nodes are shown in hollowed gray circle. The tree built has a solid edge encoded in `p`, and a dashed edge encoded in `m`.

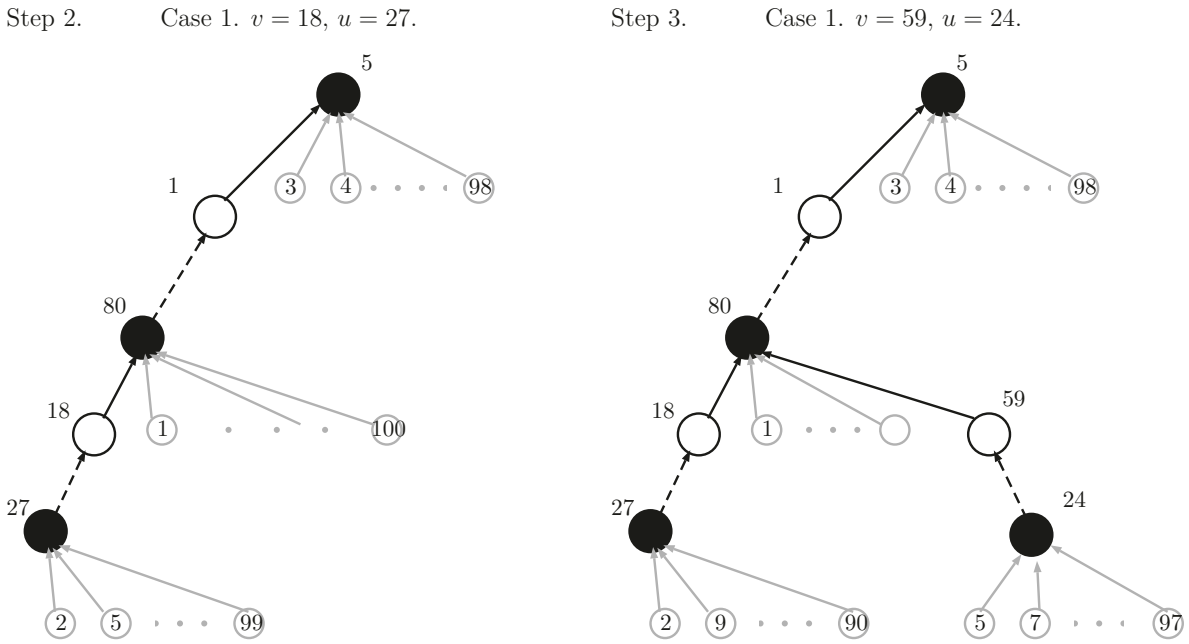


Figure S3: **Step 2 and 3.**

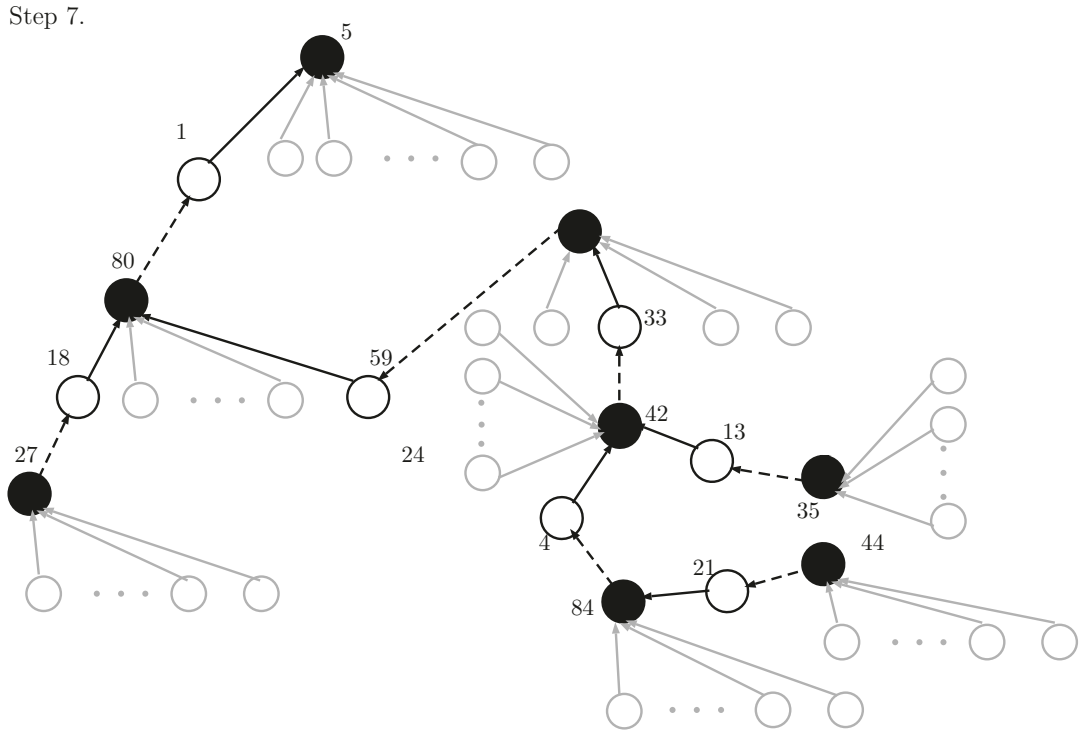


Figure S4: **Step 7.**

4 Proofs of other results

Proof of Theorem 3. Consider the following characterization of Algorithm 1 and Algorithm 2. Starting with cost matrix, square matrix of size $2m$, Algorithm 1 creates the matching M. If the nodes were labeled $1, \ldots, 2m$ then a matching is characterized by a bijective function, say ζ on $\{1, \ldots, 2m\}$ so that ζ creates a permutation of $\{1, \ldots, 2m\}$, i is matched to $\zeta(i)$, and $\zeta \circ \zeta =$Id; ζ has m transpositions.

The input to Algorithm 2 be the cost function on pairs of $N = n2^J$ units. Algorithm 2 calls Algorithm 1 J times. Let these J calls output the J mappings $\zeta_1, \ldots, \zeta_J$. Then, ζ_1 permutes $\{1, \ldots, N\}$ so that i is matched to $\zeta_1(i)$ and $\zeta_1 \circ \zeta_1 =$Id. For $x \in \{1, \ldots, N\}$, write

$$\{x\}_{\zeta_1} := \{x, \zeta_1(x)\}.$$

Suppose, ζ_2, the output to the second call to Algorithm 1, induces a permutation of $\{\{x\}_{\zeta_1} : x \in \{1, \ldots, N\}\}$. This permutation has degree N and consists of $N/2$ transpositions. Write

$$\{x\}_{\zeta_2 \circ \zeta_1} = \{x\}_{\zeta_1} \cup \zeta_2(\{x\}_{\zeta_1}).$$

Continuing this way suppose, ζ_j, the output to the second call to Algorithm 1, induces a permutation to $\{\{x\}_{\zeta_{j-1} \circ \ldots \circ \xi_1} : x \in \{1, \ldots, N\}\}$. The degree of ζ_j is $N/2^{j-1}$ so that

Step 8: Case 2, $v = 44$, $\mathtt{p}[v] = 42$

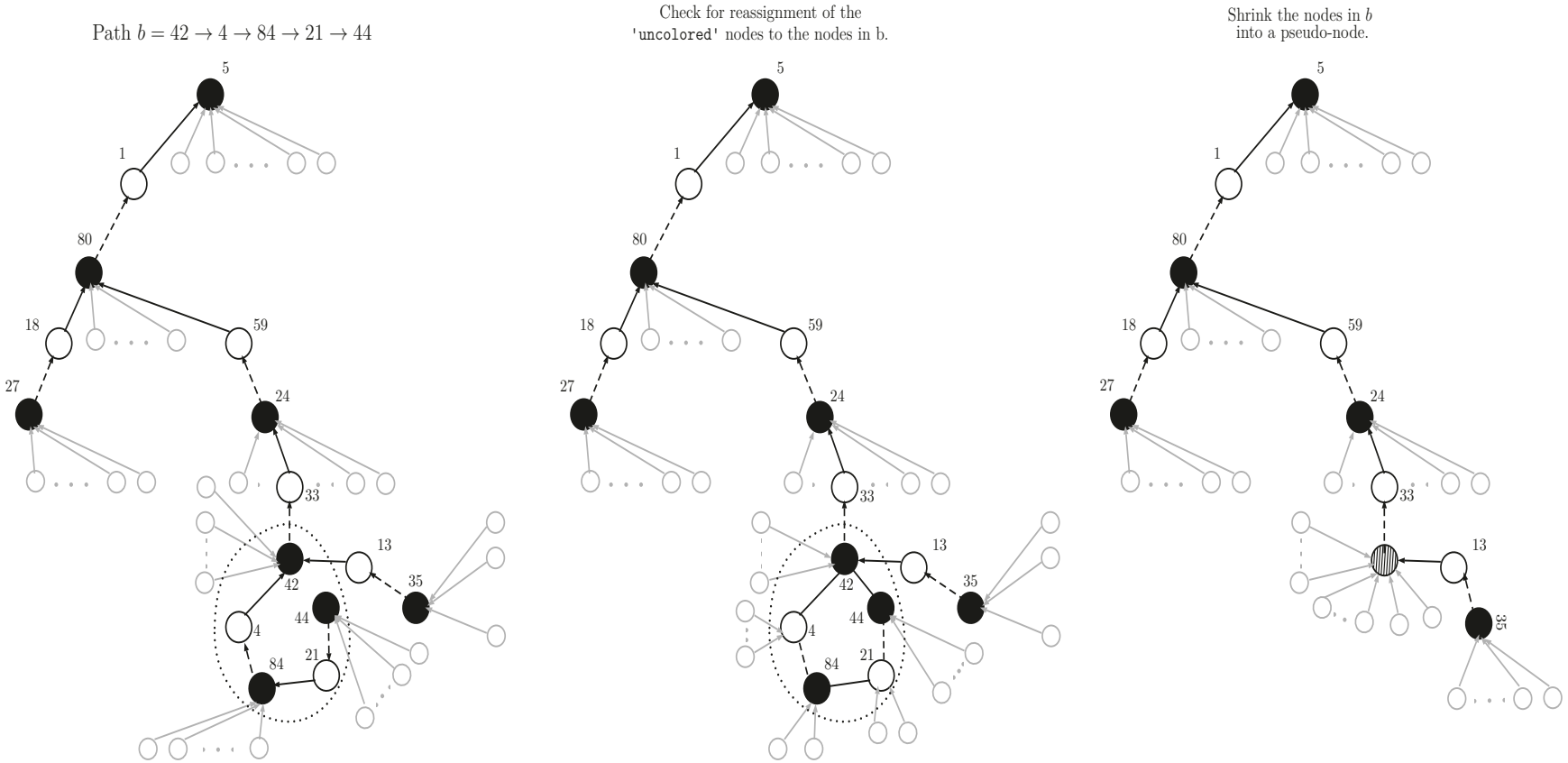


Figure S5: Step 8

is matched to . Write

For a set , define

Selection step. Let $\mathcal{B}$ be the optimal design that minimizes (1). For each $\mathcal{B}$, we do the following set of operations. Select distinct units so that for any and . It is possible to make this selection. First select arbitrarily from . Select from , and so on. Note that \ , so number of choices for is \ \ . Let . Define a one-to-one function

(S11)

arbitrarily. Notice that, by construction, .

Select a subset of in the following way. There are units in . For any two units we have . This selection can be done in a similar way as we did for . Select one unit of arbitrarily and select the second unit from . Notice that, for any , since for any other , by construction, , \ has at most one unit. Thus, after the selection of units for from there are ( )th unit is selected from at least \ units. Continue next to select from of units so that for any two units we have . Having selected , select so that

(1) ,

(2) , and

(3) for with , we have .

The sequence of subsets

creates a connection between the optimal design $\mathcal{B}$ and the output of Algorithm 2, $\mathcal{B}$.

Comparison step. for all $\mathcal{B}$. Design $\mathcal{B}$ has the form $\mathcal{B}$. Thus, the cost of design $\mathcal{B}$ is

$\mathcal{B}$ $\mathcal{B}$

To prove the theorem we need the following lemma.

Lemma S2 *For any*

$\mathcal{B}$ $\mathcal{B}$ $\mathcal{B}$

We complete the proof of the theorem using this lemma. Proof of Lemma S2 is given after. Using the lemma repeatedly we have that

$\mathcal{B}$

$\mathcal{B}$ $\mathcal{B}$

$\mathcal{B}$ $\mathcal{B}$ $\mathcal{B}$

$\mathcal{B}$ $\mathcal{B}$

Recall from (S12). Consider a pairing of units $\mathcal{B}$. Since, maps into and is one to one from , is a matching of the units . Now, as the first call to Algorithm 1 by Algorithm 2 finds the matching that is minimizes (1) and this matching is characterized by , we get that

$\mathcal{B}$ $\mathcal{B}$

Since , if is the cost of the optimal matching

$\mathcal{B}$ $\mathcal{B}$

$\mathcal{B}$ $\mathcal{B}$

$\mathcal{B}$

\ \

This completes the proof.

Proof of Lemma S2. Fix , . Note that, creates an optimal matching of $\mathcal{B}$. This is true by construction of , that are nonoverlapping for different and for , . In $\mathcal{B}$, there is another matching of $\mathcal{B}$. Within each , simply arbitrarily pair the units of . Call this function , which is a bijection on , so that is matched to . Thus, we get, by the optimality of ,

$\mathcal{B}$ $\mathcal{B}$

$\mathcal{B}$

Where

with . The term is the term that does not control for directly. It is the maximum difference across units that are in different blocks after the (\ )th call to Algorithm 1 by Algorithm 2.

Now, we control this term using the triangle inequality of the cost function.

$$\begin{aligned} B(\widetilde{b}) &\leq \max_{(x,x')\in T_{\widetilde{b}}^{(j-1)}} \max_{\substack{y\in\{x\}_{\zeta_{j-1}\circ\cdots\circ\zeta_1} \\ y'\in\{x'\}_{\zeta_{j-1}\circ\cdots\circ\zeta_1}}} (c_{xy} + c_{xx'} + c_{x'y'}) \\ &\leq 2 \max_{x\in S_{\widetilde{b}}^{(j-1)}} c(\{x\}_{\zeta_{j-1}\circ\cdots\circ\zeta_1}) + \max_{(x,x')\in T_{\widetilde{b}}^{(j-1)}} c_{xx'} \\ &\leq 2A(\widetilde{b}) + \max_{x,x'\in S_{\widetilde{b}}^{(j-1)}} c_{xx'}. \end{aligned}$$

The proof of the lemma is thus complete by noting that the cost function is nonnegative, thus, $A(\widetilde{b}) \leq 2A(\widetilde{b}) + \max_{x,x'\in S_{\widetilde{b}}^{(j-1)}} c_{xx'}$.

Proof of Proposition 4.

(a) First consider the blocking problem without and structure on the costs. We'll show a reduction of to the following problem.

Q1. Suppose $G = (V, E)$ is any graph with $|V| = nk$ nodes, and $H = (V', E')$ is another graph with $|V'| = k$. Can the vertices of G be partitioned into k disjoint sets $V_1, \ldots, V_n$ such that, for each i, the subgraph of G induced by V_i is isomorphic to H?

This is a problem whose answer is either 'Yes' or 'No'. Kirkpatrick and Hell (1978) showed that for any fixed H with $|V'| = k > 3$, the problem is NP-complete. In other words, given any fixed H there is no polynomial time algorithm that can answer this binary question for all graphs G.

Let H be a complete graph of k vertices. For a graph G we consider the following blocking problem of nk units. These nk units are identified with the nk vertices of G. Let the cost c_{ij} between two units i and j be 1 is the corresponding nodes are connected in E and be $(1 + 2\alpha)$ is they are not connected in E. Notice that, the solution to the blocking problem is 1, the maximum in-block differences, when and only when the answer to Q1 is 'Yes'. Only only other possible value of the blocking problem is $(1 + 2\alpha)$.

Suppose, there is a polynomial time algorithm that can solve the blocking problem at an approximation factor $(1 + \alpha)$. Suppose $\tilde{c} = 1$, i.e., the answer to Q1 is 'Yes', then $c^{\star\star} \leq (1 + \alpha) \times 1$. Since only two possible values of $c^{\star\star}$ are 1 and $(1 + 2\alpha)$, when $\tilde{c} = 1$ the approximation algorithm finds $c^{\star\star} = 1$. Also, when $\tilde{c} = (1 + 2\alpha)$, we cannot have $c^{\star\star} = 1$. Thus, the approximation algorithm will output $c^{\star\star} = 1$ if and only if the answer to Q1 is 'Yes'. Now using NP-completeness of the problem of finding the answer to Q1, we conclude that the blocking problem cannot be solved by a polynomial time algorithm, with an approximation factor $(1 + \alpha)$.

(b) Now consider the blocking problem with structure (4). As before, let H be a complete graph with k vertices. For any graph $G = (V, E)$ with $|V| = nk$ vertices, consider the correspondences of the nk vertices to nk units. The cost, c_{ij}, between two units i and j is $2\beta/\alpha$ if they are connected in E and it is β/α be they are not connected in E. As before note that answer to Q1 is affirmative if and only if the solution to this blocking problem is $\tilde{c} = 2\beta/\alpha$. Only two possible values are $2\beta/\alpha$ and β/α.

Now suppose an approximation is available so that the solution to the blocking problem by the approximation algorithm is $c^{\star\star} \leq (3-\alpha)\tilde{c} + \beta$. Then, $\tilde{c} = 2\beta/\alpha$ implies

$$c^{\star\star} \leq (3-\alpha)\tilde{c} + \beta = (3-\alpha)2\beta/\alpha + \beta = \ \beta/\alpha - \beta < \ \beta/\alpha.$$

Hence, when the answer to Q1 is 1 or equivalently, $\tilde{c} = 2\beta/\alpha$, the approximation algorithm will also achieve the solution $c^{\star\star} = 2\beta/\alpha$. On the other hand, $c^{\star\star} \neq 2\beta/\alpha$ when $\tilde{c} \neq 2\beta/\alpha$. Thus, if such an approximation algorithm exits to solve the blocking problem, then using the above reduction, we can answer Q1 for any graph G. But, Q1 is a NP-complete problem. Hence the proof.

Proof of Theorem 5. We borrow the notation in the proof of Theorem 3. Suppose the locally optimal design is $\mathcal{B}' \in \mathcal{L}(\mathcal{B})$, where $\mathcal{B}$ is the output of Algorithm 2. Suppose $x_{b'} \in \mathcal{B}'$ is such that $x_{b'}$ is not in the same block as the rest of the units of b' in $\mathcal{B}$. Make the selection of $S_{b'}^{(1)}$ such that $x_{b'} \notin S_{b'}^{(1)}$. This is always possible, since, as discussed in the proof of Theorem 3, there are two choices for the selection of the last unit of the set $S_{b'}^{(1)}$.

$$\max_{b\in\mathcal{B}} \max_{i,j\in b; i\neq j} c_{ij} = \max_{b'\in\mathcal{B}'} \max\{ \max_{x\in S_{b'}^{(1)}} c_{x,\zeta_1(x)}, \max_{x,x'\in S_{b'}^{(1)}} c_{x,x'}, \max_{x,x'\in S_{b'}^{(1)}} c_{\zeta_1(x),\zeta_1(x')}\}.$$

Suppose, $x_{b'}^{\star} \in S_{b'}^{(1)}$ is such that $\zeta_1(x_{b'}^{\star}) \notin b'$. We have, by the triangle inequality

$$c_{x,\zeta_1(x_{b'}^{\star})} \leq c_{x,x_{b'}^{\star}} + c_{x_{b'}^{\star},\zeta_1(x_{b'}^{\star})}.$$

Finally, using this fact and the optimality of ζ_1 we get that

$$\max_{b\in\mathcal{B}} \max_{i,j\in b; i\neq j} c_{ij} \leq 2 \cdot \max_{b'\in\mathcal{B}'} \max_{i,j\in b'; i\neq j} c_{ij}.$$

Proof of Theorem 6. The first step to prove the theorem is to follow the 'selection step' of the proof of Theorem 3. Following that discussion let $\xi_1, \ldots, \xi_J$ be the functions that characterizes the outputs to the J calls to Algorithm 1 by Algorithm 2 for (7).

Let, $N = n2^J$. Write for $x \in \{1, \ldots, N\}$

$$\{x\}_{\xi_1} := \{x, \xi_1(x)\},$$

$$\{x\}_{\xi_j \circ \ldots \circ \xi_1} = \{x\}_{\xi_{j-1}\circ\ldots\circ\xi_1} \cup \xi_j(\{x\}_{\xi_{j-1}\circ\ldots\circ\xi_1}).$$

For a set $A \subseteq \{1, \ldots, N\}$, define

$$c_s(A) := \sum_{i,j\in A} c_{ij}.$$

Selection Step. Let $\widetilde{\mathcal{B}} = \{\widetilde{b}_1, \ldots, \widetilde{b}_n\}$ be the optimal design that minimizes (7). For

each $\mathcal{B}$, we do the following set of operations. Select distinct units so that for any and . Let . Define a one-to-one function

(S12)

arbitrarily. Notice that, by construction, . Having selected , for , select so that

(1) ,

(2) , and

(3) for with , we have .

Optimality of . For each , arbitrarily pair the units of . Call this function , which is a bijection on , so that is paired to . Using these 's define , a pairing of the units of $_{\mathcal{B}}$ so that pairs units in for all $\mathcal{B}$ according to .

Since optimaly pairs the units in $_{\mathcal{B}}$ we have

$_{\mathcal{B}}$

$_{\mathcal{B}}$

(S13)

$_{\mathcal{B}}$

Comparison step. Since and for $_{\mathcal{B}}$ with , by construction . Thus, if $\mathcal{B}$ is the output of Algorithm 2 for (7) then

$_{\mathcal{B}}$ — — $_{\mathcal{B}}$ (S14)

This establishes the connection between the cost of the design $\mathcal{B}$ and the optimal design

$\mathcal{B}$. By (S14) and (S13)

$$\mathcal{B}$$

$$\mathcal{B}$$

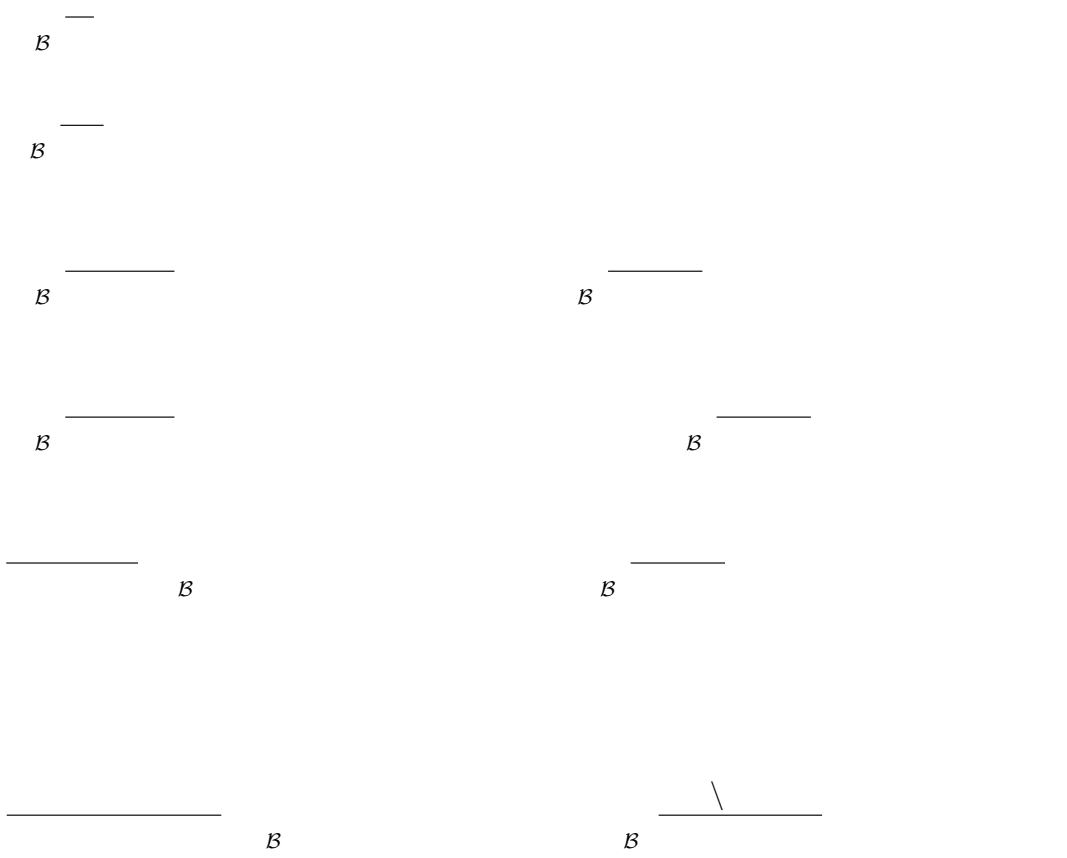
$$-$$

$$\mathcal{B}$$

$$\mathcal{B}$$

By the triangle inequality for ,

Thus



(S15)

where , and . In the first term now use (S13) to replace by .

For the coefficient for the first term in (S15) we write

$$\begin{aligned}
&\prod_{j=1}^{J-1}(2^j + 2^{J-j}) \\
&\leq \Big[\frac{1}{J-1}\sum_{j=1}^{J-1}(2^{J-j} + 2^j)\Big]^{J-1} \\
&= \Big[\frac{1}{J-1}\{2^J(1-2^{-J+1}) + 2(2^{J-1}-1)\}\Big]^{J-1} \\
&= \Big[\frac{1}{J-1}\{k(1-2/k) + 2(k/2-1)\}\Big]^{J-1} \\
&= (k-2)^{J-1}(2/(J-1))^{J-1} \leq 8(k-2)^{J-1}/k
\end{aligned}$$

The inequality above follows from the fact that geometric mean $\leq$ arithmetic mean. The last inequality follows since $(2/(J-1))^{J-1} \leq 8 \cdot 2^{-J}$. This term can by dispersed into $(k-1)$ terms, which gives us a factor

$$8(k-2)^{J-1}/(k(k-1)) \leq 8(k-2)^{J-3}.$$

The jth term in (S15) in the sum can be dispersed into $(2^J - 2^{J-j} + 1)/(2^{J-j} - 1) = 2^{J+j}/(2^J - 2^j) - 1 \geq 2^j - 1$ terms. Use the GM $\leq$ AM inequality to write

$$p(j;J)^{1/(j-1)} \leq \frac{1}{j-1}(2^j - 2 + 2^J - 2^{J-j+1}) \leq \frac{1}{j-1}(3 \cdot 2^{J-1} - \ ) = \frac{1}{j-1}(3/2)(k-4). \quad \text{(S16)}$$

To get the second inequality above we have used $j = J-1$.

Using (S16) and that $(1/(j-1))^{j-1} \leq 2 \cdot 2^{-(j-1)}$ we get

$$\begin{aligned}
\sum_{j=1}^{J-1}\frac{p(J-j;J)4^j}{2^j-1} &\leq \sum_{j=1}^{J-1} 2 \cdot 2^J\{(3/8)(k-4)\}^{j-1} \\
&\leq 2 \cdot 2^J \frac{(3/8)^{J-1}(k-4)^{J-1}}{(3/8)(k-4)-1} \\
&= 2 \cdot 2^J (3/8)^{J-2}\frac{(k-4)^{J-1}}{(k-20/3)} \\
&\leq 2 \cdot 4 \cdot (3/4)^{J-2}\frac{(k-4)^{J-1}}{(k-7)} \\
&\leq 2 \cdot 4 \cdot (3/4)^2 \cdot 1 \cdot \frac{(k-4)^{J-1}}{(k-7)} \\
&= 9/2\frac{(k-4)^{J-1}}{(k-7)} \\
&\leq 9/2 \cdot k(k-4)^{J-3}
\end{aligned}$$

For , from (S15) and with the dispersion terms, the approximation factor is

\ \

For ,

\ \

5 Comparison using simulations

In this section we evaluate the performance of the proposed methods and compare them to the available methods in different simulation scenarios. Including Scenario 1 and Scenario 2 of §4.3, three simulation scenarios are considered. The third simulation scenario has two binary and four continuous variables, and these variables are correlated. To simulate such a structure a 6 dimensional random vector is simulated from a mixture of three multivariate normal distributions with mixing proportions 1/2, 3/8 and 1/8 respectively. Each mixing component has the same covariance structure: the variance of each variable is 1 and the covariance of any two variables is 0.3. The normal mean vectors for the three components are $(0,0,0,0,0,0)^\top$, $(0,0,2,2,-2,-2)^\top$, and $(0,0,5,5,-\ ,-\ )^\top$, respectively. Finally, the first two values of the vector, say x_1 and x_2, are discretized as: $x_1^* = 1$ if $x_1 < 0.1$ and $x_2^* = 1$ if $x_2 > 0.1$, and 0 otherwise. The cost function used in this scenario is the square root of rank based robust Mahalanobis distance (Rosenbaum, 2010, §8.3).

We compare eight methods for $k = 4$, as listed in Table S5 and Table S6. The first four are: Algorithm 2 with and without its improvement using Algorithm 3 for the two problems (1) and (7). The other four methods are the available methods in the literature, see §1.3. The 'Random templating and assignment' method is the randomized algorithm of that creates many blockings by randomly selecting one template unit for each block, and outputs a blocking that is the best among those. The number of blockings it creates is a parameter of the algorithm. In the simulations, we set this number to 30, a suggestion by Karmakar (2018). The next two methods, method 6 and 7 in the tables, are the 'threshold blocking' method. For a given threshold k, this method finds a blocking of the units into blocks of at least k units. In the simulation, the threshold blocking method is evaluated for two threshold values, $k = 3$ and $k = 4$. The last of the methods in the tables is the 'optimal greedy' method. As of the latest available implementation of this method, unless a data set is provided this method cannot create a blocking. Thus, for Scenario 1 the results of this method are not available.

In the tables, two measures are reported. The first measure is the maximum of largest in-block paired differences. Second is the maximum of the average in-block paired differences. They correspond to the objective functions of problems (1) and (7) respectively.

In Table S5, the numbers are percentages with respect to the baseline algorithm, Algorithm 2 for problem (1), in the first row. A smaller value in Table S5 represents a better performance by the method. Consider first Scenario 1. In this scenario, compared to Algorithm 2 for (1) we do not see any improvement by additionally using Algorithm 3. Although, we do observe improvements in both the measures for Algorithm 2 for (7) by using Algorithm 3. The threshold blocking method shows identical performance for both choices of the threshold. Further, they are considerably poorer compared to the random templating method, which is closer to the methods proposed in this paper, but still inferior to all of them.

Scenarios 2 and 3 create Mahalanobis distances from two different kinds of multivariate data sets. In these two scenarios using Algorithm 3 shows a marginal improvement over Algorithm 2 for (1) — a comparatively better improvement for Scenario 3 over Scenario 2. The ranking of the last four methods by their performances is the same order in which they

Table S1: Table of the values. Shows the performance in percentage terms, compared to the baseline, the first row averaged over 500 simulations in each scenario. Two performance measurements are considered, and three simulation scenarios are considered (described in the text). The pair of numbers in parenthesis are the 10% and the 90% quantiles. A smaller value in the table represents better performance; the smallest one(s) in each column is highlighted in bold.

Measurement	Max largest in-block differences			Max average in-block differences		
	Scenario			Scenario		
	1	2	3	1	2	3
Algorithm 2 for (1) (baseline)	**100**	100	100	100	100	100
Algorithm 2+3 for (1)	**100** (100, 100)	**99** (98, 100)	**97** (91, 100)	100 (100, 100)	100 (100, 100)	100 (96, 103)
Algorithm 2 for (7)	121 (100, 131)	104 (100, 114)	111 (101, 121)	85 (79, 90)	96 (88, 100)	94 (86, 100)
Algorithm 2+3 for (7)	119 (110, 130)	106 (100, 119)	116 (104, 127)	**82** (77, 88)	**94** (83, 100)	**89** (81, 96)
Random templating and assignment	143 (127, 184)	133 (101, 170)	131 (114, 148)	103 (88, 136)	123 (97, 158)	117 (103, 132)
Threshold blocking with	202 (194, 209)	139 (96, 193)	139 (114, 166)	265 (248, 284)	139 (97, 189)	136 (112, 164)
Threshold blocking with	202 (194, 209)	168 (119, 225)	152 (127, 180)	265 (248, 284)	146 (104, 189)	138 (118, 163)
Heuristic blocking	–	320 (206, 431)	194 (158, 233)	–	309 (198, 435)	181 (148, 218)

Karmakar (2018); Higgins et al. (2016); Moore (2012).

are listed in the table. The very last method, the heuristic blocking method (the optimal greedy method of Moore, 2012), performs very poorly; in certain instances the values are 430% of the baseline, on an average 320% and 309% of the baseline for the maximum largest in-block difference and maximum average in-block difference respectively. The randomized blocking method again performs better than the threshold blocking method — only marginally better in the first but considerably better in the second measurement.

In the simulation, for the three scenarios, the threshold blocking method with threshold has average block sizes 5.8, 3.9 and 4 respectively, and with has average block sizes 9.1, 5.4 and 5.7 respectively.

Algorithm 2 for (7) with improvements using Algorithm 3, gives the best performance in the simulation, when we look at the maximum average in-block difference. This method also fares well for the maximum largest in-block difference, compared to the best algorithm.

Table S6 complements Table S5. In each scenario, this table reports the percentage of the simulation instances a method is the winner, in the corresponding measure, among the 8 methods compared. The columns in the table do not sum to 100% because multiple methods can have the best value of a measurement in a simulated instance. It is clear from Table S6 that proposed Algorithm 2 with 3 for (1), the second method in the table, has the most wins for the maximum of the largest in-block differences and the proposed Algorithm

Table S2: Table of the winners. In each scenario and for the two measurements, the percentage of times the method performs the best among all the methods. In each column the best one(s) are highlighted in bold.

Measurement	Max. largest in-block difference			Max. average in-block difference		
	Scenario			Scenario		
	1	2	3	1	2	3
Algorithm 2 for (1)	**100**	70	56	0	28	2
Algorithm 2+3 for (1)	**100**	**82**	**99**	0	29	2
Algorithm 2 for (7)	0	41	4	15	58	22
Algorithm 2+3 for (7)	0	34	2	**100**	**88**	**100**
Random templating	0	7	0	0	6	0
Threshold blocking with	0	15	0	0	8	0
Threshold blocking with	0	1	0	0	3	0
Heuristic blocking	–	0	0	–	0	0

2 with 3 for (7), the fourth method in the table, has the most wins for the maximum of the average in-block differences. Only in scenario 2, we see a few wins for the random templating method and the threshold blocking method; no wins for the heuristic blocking method. In scenarios 2 and 3, we can see noticeable improvements by using Algorithm 3. In scenario 2, Algorithm 3 improves Algorithm 2 for (1) in at least (82\ 70)=12% of the instances, and in scenario 3, in at least (99\ 56)=43% of the instances. Algorithm 3 also improves Algorithm 2 for (7), in at least (100\ 22)=78% of the instances in Scenario 3, in at least (88\ 58)=30% of the instances in Scenario 2, and in at least (100\ 15)=85% of the instances in Scenario 1.

6 Additional simulation results

This section compares the different methods of blocking to create blocks of size and , while the main text presented the results for . Three simulation scenarios are considered and they have been discussed in the main text. The conclusions of the simulation results of for still holds. Algorithm 3 provides a sufficient improvement over Algorithm 2 in all three simulation scenarios, more for Scenarios 2 and 3. For minimizing the maximum of the block-wise worst pair, Algorithm 2+3 for (1) outperforms all other methods in all but one situation. The randomized templating method of Karmakar (2018) performs best for and scenario 2. In this case, the random templating method also outperforms other methods in the measure of maximum of the average in-block differences. This simulation is an exception in some of the remarks below as well.

The heuristic blocking method Moore (2012) fairs very poorly in every simulation instance and measure of comparison. Thresholds blocking method comes close second in all but the above one exception. We note that for the threshold blocking algorithm with threshold created blockings with average block size , and for the three simulation scenarios respectively. For the other set of simulations with , the threshold

blocking algorithm with threshold 8 creates blockings with average block size 31.3, 11. and 13.1 for the three simulation scenarios respectively. In the same simulations, a threshold of 7 creates blockings with average block size 25.3, 10 and 11.2 for the three simulation scenarios respectively. These averages are over 500 simulated instances for each scenario.

Table S3: k=3. Table of the values. Compared to the baseline, the first row, the value of the measurement in percentage terms, averaged over 500 simulations in each scenario. Two performance measurements are considered, and three simulation scenarios are considered (described in the text). The pair of numbers inside a parenthesis represents the 10% and the 90% quantile of the value out of the 500 iterations. A smaller value in the table represents better performance; the smallest one(s) in each column is highlighted in bold.

Measurement	Max largest in-block difference			Max average in-block difference		
	Scenario			Scenario		
	1	2	3	1	2	3
Algorithm 2 for (1) (baseline)	**100**	100	100	100	100	100
Algorithm 2+3 for (1)	**98** (93, 100)	91 (69, 100)	**95** (83, 100)	98 (94, 100)	91 (69, 100)	97 (87, 100)
Algorithm 2 for (7)	108 (100, 121)	104 (85, 124)	101 (89, 109)	100 (94, 107)	103 (85, 125)	97 (87, 105)
Algorithm 2+3 for (7)	107 (99, 120)	91 (69, 101)	98 (85, 106)	**97** (92, 100)	91 (68, 100)	**93** (83, 100)
Random templating and assignment†	138 (116, 181)	**80** (53, 109)	112 (92, 133)	124 (105, 177)	**84** (56, 115)	119 (99, 137)
Threshold blocking†† with $k = 3$	192 (181, 201)	94 (58, 136)	127 (104, 155)	272 (252, 284)	92 (56, 134)	133 (108, 164)
Heuristic blocking†††	–	187 (108, 276)	169 (130, 209)	–	205 (112, 303)	185 (144, 228)

† Karmakar (2018); †† Higgins et al. (2016); ††† Moore (2012).

Table S4: k=3. Table of the winners. In each scenario and for the two measurement criterion, the percentage of times the method performs the best among all the methods. A larger value implies better performance, and in each column the best one(s) are highlighted.

Measurement	Max. largest in-block difference			Max. average in-block difference		
	Scenario			Scenario		
	1	2	3	1	2	3
Algorithm 2 for (1)	68	9	51	15	9	20
Algorithm 2+3 for (1)	**100**	16	**91**	19	15	20
Algorithm 2 for (7)	9	6	17	52	10	63
Algorithm 2+3 for (7)	10	11	18	**100**	20	**98**
Random templating	0	**60**	8	0	**46**	2
Threshold blocking with	0	27	1	0	37	0
Heuristic blocking	–	0	0	–	0	0

Table S5: k=8. Table of the values. Compared to the baseline, the first row, the value of the measurement in percentage terms, averaged over 500 simulations in each scenario. Two performance measurements are considered, and three simulation scenarios are considered (described in the text). The pair of numbers inside a parenthesis represents the 10% and the 90% quantile of the value out of the 500 iterations. A smaller value in the table represents better performance; the smallest one(s) in each column is highlighted in bold.

Measurement	Max largest in-block difference			Max average in-block difference		
	Scenario			Scenario		
	1	2	3	1	2	3
Algorithm 2 for (1) (baseline)	**100**	100	100	100	100	100
Algorithm 2+3 for (1)	**100** (100, 100)	**98** (92, 100)	**97** (92, 100)	100 (100, 100)	99 (95, 100)	100 (98, 104)
Algorithm 2 for (7)	121 (111, 131)	105 (96, 117)	117 (107, 128)	85 (80, 90)	94 (81, 103)	90 (84, 97)
Algorithm 2+3 for (7)	122 (113, 131)	129 (100, 158)	130 (117, 146)	**84** (79, 89)	**87** (74, 100)	**84** (78, 91)
Random templating and assignment	129 (120, 135)	129 (100, 164)	126 (112, 142)	99 (90, 107)	113 (90, 141)	111 (100, 123)
Threshold blocking with	175 (163, 188)	169 (123, 220)	153 (132, 178)	147 (87, 198)	146 (106, 193)	141 (123, 163)
Threshold blocking with	175 (162, 189)	180 (133, 232)	158 (137, 183)	134 (85, 181)	150 (113, 194)	143 (126, 163)
Heuristic blocking	–	308 (231, 382)	192 (162, 220)	–	297 (200, 389)	178 (149, 206)

Karmakar (2018); Higgins et al. (2016); Moore (2012).

Table S6: k=8. Table of the winners. In each scenario and for the two measurement criterion, the percentage of times the method performs the best among all the methods. A larger value implies better performance, and in each column the best one(s) are highlighted in bold.

Measurement	Max. largest in-block difference			Max. average in-block difference		
	Scenario			Scenario		
	1	2	3	1	2	3
Algorithm 2 for (1)	**100**	59	49	0	11	2
Algorithm 2+3 for (1)	**100**	**81**	**100**	0	12	0
Algorithm 2 for (7)	0	34	0	46	26	5
Algorithm 2+3 for (7)	0	12	0	**91**	**92**	**100**
Random templating	0	9	0	0	4	0
Threshold blocking with	0	0	0	5	0	0
Threshold blocking with	0	0	0	4	1	0
Heuristic blocking	–	0	0	–	0	0

References

Bind, M. A. and Rubin, D. B. (2020). When possible, report a Fisher-exact P value and display its underlying null randomization distribution. . *Proceedings of the National Academy of Sciences*, **117**, 19151–19158. (Not cited.)

Box, G. E.P. and Hunter, J. S., (1961). The fractional factorial designs, *Technometrics*, **3**, 311-351. (Not cited.)

Cheng, C. S. and Mukerjee, R. (2001). Blocked regular fractional factorial designs with maximum estimation capacity. *Annals of Statistics*, **29**, 530–548. (Not cited.)

Derigs, U. (1988). Solving nonbipartite matching problems via shortest path techniques. *Annals of Operations Research,* **13**, 225–261. [13].

Edmonds, J. (1965a). Paths, trees, and flowers. *Canadian Journal of Mathematics*, **17**, 449–467. (Not cited.)

Edmonds, J. (1965b). Maximum matching and a polyhedron with 0, 1-vertices. *Journal of Research of the National Bureau of Standards B. Mathematics and Mathematical Physics*, **69**, 125–130. (Not cited.)

Federer, W. T. and King, F. (2007). *Variations on split plot and split block experiment designs* . John Wiley & Sons, Hoboken, NJ. (Not cited.)

Higgins, M. J., Sävjeb, F. and Sekhon, J. S. (2016). Improving massive experiments with threshold blocking. *Proceedings of the National Academy of Sciences*, **113**, 7369–7376. [29, 31, and 32].

Karmakar, B. (2018). blockingChallenge: Create blocks or strata which are similar within. *CRAN*, R package version 1.0. [28, 29, 30, 31, and 32].

Kelcey, B., Spybrook, J., Phelps, G., Jones, N. and Zhang, J. (2017). Designing large-scale multisite and cluster-randomized studies of professional development, *THe Journal of Experimental Education*, **85**, 389–410. (Not cited.)

Kirkpatrick, D. G. and Hell, P. (1978). On the completeness of a generalized matching problem. In *Proceedings of the tenth annual ACM symposium on Theory of computing* pp. 240–245. [21].

Moore, R. T. (2012). Multivariate continuous blocking to improve political science experiments. *Political Analysis*, **20**, 460–479. [29, 30, 31, and 32].

Robinson, J. (1970). Blocking in incomplete split plot designs, *Biometrika*, **57**, 347–350. (Not cited.)

Rosenbaum, P. R. (2010). *Design of Observational Studies*. New York: Springer. [28].

Rubin, D. B. and Thomas, N. (2000). Combining propensity score matching with additional adjustments for prognostic covariates. *Journal of the American Statistical Association*, **95**, 573–585. (Not cited.)

Supplement: An approximation algorithm for blocking of an experimental design

Bikram Karmakar
Department of Statistics, University of Florida, Gainesville, FL 32611, USA.

1 Proof of Theorem 1

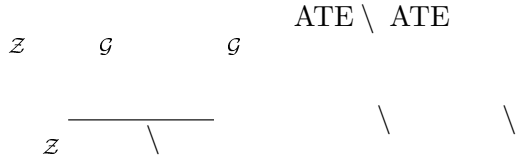
We first show that

(S1)

Start form the left hand side

Part (a).

If then using identity (S1)



Part (b).

We consider where is a fixed vector. We first check that this is in $\mathcal{G}$. We use the shorthand . Fix and . Without loss of generality assume .

\ \

\

\

The first inequality follows by triangle inequality and the second inequality follows since .

Consider values that are either or \ . There are 0s and \ . Verify that these set of values control the average distance. Using triangle inequality

\ \

\

\ \

Throughout we let \ .

Let $\mathcal{L}$ be the set of units where the treatment is assigned freely. Let $\mathcal{L}$ $\mathcal{L}$ be the set of units that are assigned treatment and let $\mathcal{L}$ $\mathcal{L}$ $\mathcal{L}$. Let $\mathcal{L}$ $\mathcal{L}$ $\mathcal{L}$, where $\mathcal{L}$ is the subset of $\mathcal{L}$ corresponding to . By (S1)

ATE \ ATE \ \

$\mathcal{L}$ $\mathcal{L}$ $\mathcal{L}$ $\mathcal{L}$

\ $\mathcal{L}$ \ \ $\mathcal{L}$ \

\ \

$\mathcal{L}$ $\mathcal{L}$

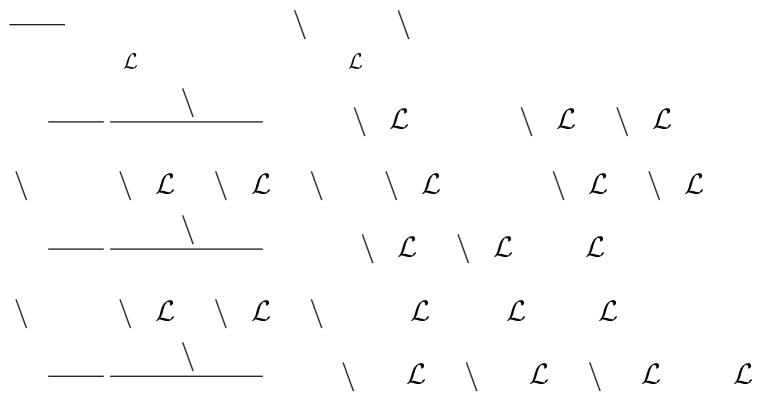
\ \

$\mathcal{L}$ \ $\mathcal{L}$

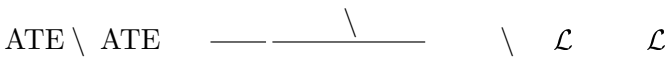
\ \

$\mathcal{L}$ $\mathcal{L}$

Among the $\mathcal{L}$ units, assign the rest possible untis to the treatment. We note that $\backslash$ $\mathcal{L}$ units remain to be treated, and $\backslash$ $\mathcal{L}$ $\backslash$ $\mathcal{L}$ of the units with $\backslash$. Thus, $\backslash$ $\mathcal{L}$ $\backslash$ $\mathcal{L}$ $\backslash$ $\mathcal{L}$ of the remaining units with $\backslash$ in $\mathcal{L}$ are assigned treatment. The other $\backslash$ $\mathcal{L}$ $\backslash$ $\mathcal{L}$ $\backslash$ $\backslash$ $\mathcal{L}$ $\backslash$ $\mathcal{L}$ $\backslash$ $\mathcal{L}$ units from $\mathcal{L}$ with $\backslash$ are assigned to control. Thus,



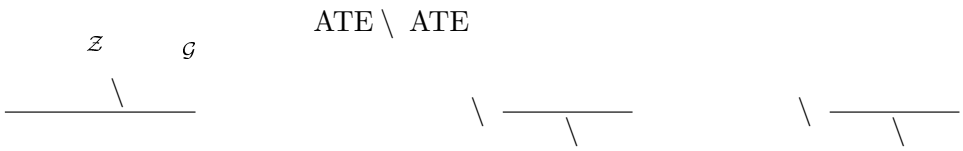
Thus, we get



By definition of the set $\mathcal{L}$'s the restriction on the sets $\mathcal{L}$ and $\mathcal{L}$ is that their sizes are at most . Hence the proof.

This inequality can be shapened by considering the other assignment where the remaining possible control units are from the remaining units with $\backslash$. The sharpened lower bound is (i) if

(ii) if



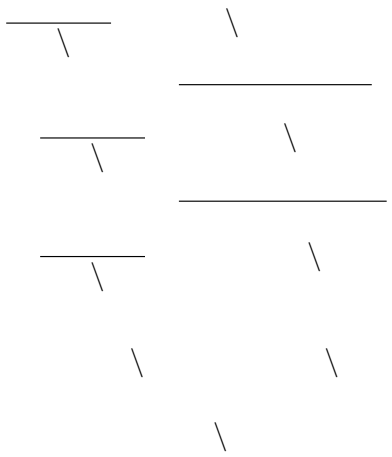
We now consider a slight modification of the inequality. Assume that s are univariate and is the absolute distance. Fix small, a point and pick rather than being exactly at be in the interval $\backslash$. Pick the other points in the interval $\backslash$ $\backslash$ $\backslash$. It is easy to verify that $\backslash$. Thus, using triangle inequality the average pairwise distance between any two points is

Notice that the difference between the two sets of points is at least \ \ \ . Next, as before let . But now notice that the error in ATE estimation is at least as much as our calculation before except now previous imbalance \ replaced by \ \ \ . Hence we get the proof of the Theorem.

Lemma S1 *This lemma gives a lower bound for the conditional probability of getting a set of 's so that the lower bound of part (b) of Theorem 1 is attained given that the average pairwise distance is at most when the \ and 's are sampled i.i.d. from a standard normal distribution.*

Proof. We assume that are sampled randomly from the standard normal distribution. We want to find the conditional probability that we observe of the units are in an interval of length and the other units are also in another interval of length , and the maximum distance between any two points is at most \ given that the average pairwise distance of any pointst is at most

First we start by computing the denominator of the conditional expectation.



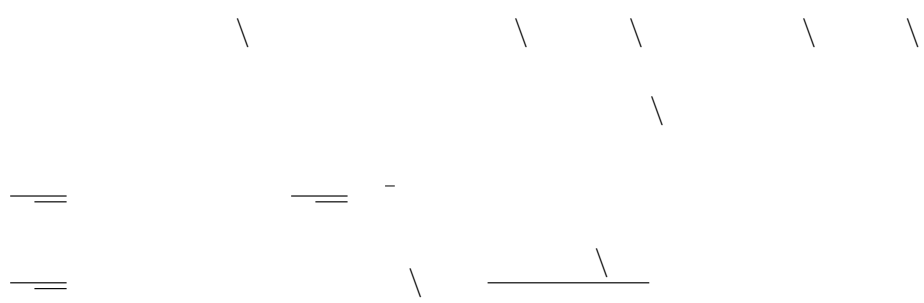
Now consider the numerator which is the probability of observing random samples of a particular pattern. We calculate the probability that are in an interval of length and the other units are in another interval of length , and the maximum distance between any two points is at most \ . Then the numerator will be times this probability.

The probability in question is calculated in four parts: (a) if all the points are in the positive real line, (b) if all the points are in the negative real line, (c) the first set of points are negative and the second set of points are positive, and (d) the first set of points are postive and the next set of points are negative,

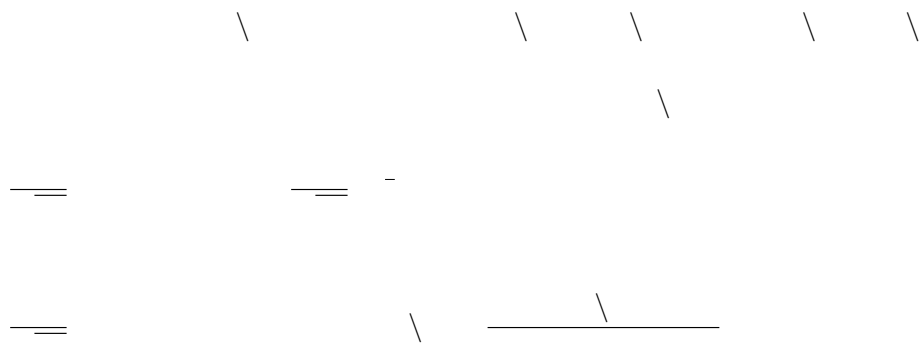
Notice that probability of (a) and (b) are the same since the distribution is symmetric around 0. Let's consider (a) then. We can split (a) in two parts: (a1) points come first

and then the points, and (a2) the points come first then the points. Throughout we use and denote the standard normal density and distribution function respectively.

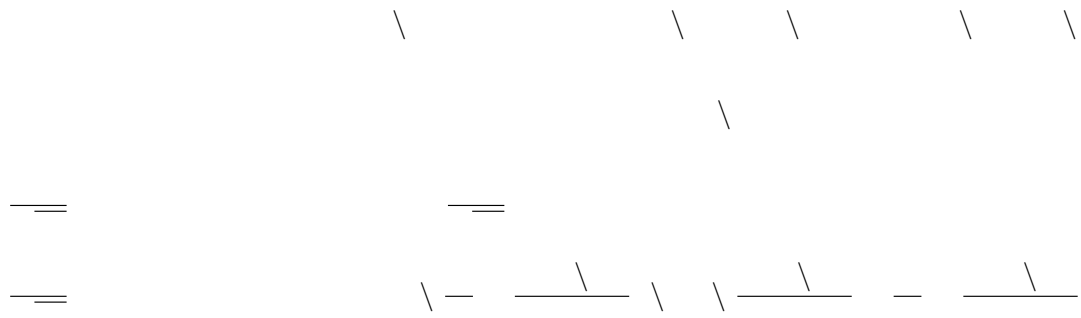
We start by calculating probabilities for (a1) and (a2). Probability of (a1) is



Probability of (a2) is



It remains to calculate the lower bounds of the probabilities of (c) and (d).



Finally, we have the probability of (d)

Finally, we get the lower bound of the conditional probability by adding the lower bounds for (a1), (a2), (b), (c) and (d) and then diving it by .

We can calculate this lower bound numerically. Using this lemma we calculate, when , and the probability is 3.2%, when , and the probability is at least 5%.

2 Proof of Theorem 2

(a) We first see the only-if part of the theorem. If is a matching and it permits a negative alternating cycle, , then is another matching and . Thus, cannot be an optimal matching.

Now we prove the if part of the statement. Suppose is a matching that does not permit any negative alternating cycle. We start by observing that any other matching of the same set of matched vertices can be created from by composing it with alternating cycles. More specifically, if is any other matching on the same set of vertices as and , then there are non-overlapping alternating cycles with respect to so that .[1] Since, does not permit any negative alternating cycles, we have .

We write, , where we denote by . Also, . Let , and . Thus implies, either , or . Further,

[1] Note though that it is possible for two maximal matchings and for the complete graph, for any alternating cycle , that is, for two matchings on the same set of vertices composition with more than one alternating cycles may be necessary to transform one matching to the other.

Now,

(S2)

For any , we have either or . In the second case, we also have, . Hence, in this case, . Using these relations, thus, in (S2) we have

\

This completes our proof of part (a).

(b) Let be a matching of size that permits no negative alternating cycle. Let be a shortest augmenting path of , and define .

The proof is by contradiction. If possible suppose, is a negative alternating cycle of , i.e., is an alternating cycle with respect to the matching and .

First consider the case when \ . Note that

(S3)

Where we use the notation to shorthand . When \ , we have the following

\ \

\ \ \

\ \

\

\ \

\

Where we have used De-Morgan's law and the distributive law for the second and the third

equality respectively. Also

\ \
\ \
\

Rewrite (S3) as

(S4)

Note that, as \

\ (S5)

Also

\ (S6)

and

(S7)

Since is a negative alternating cycle with , from (S4) and (S7)

(S8)

This implies,

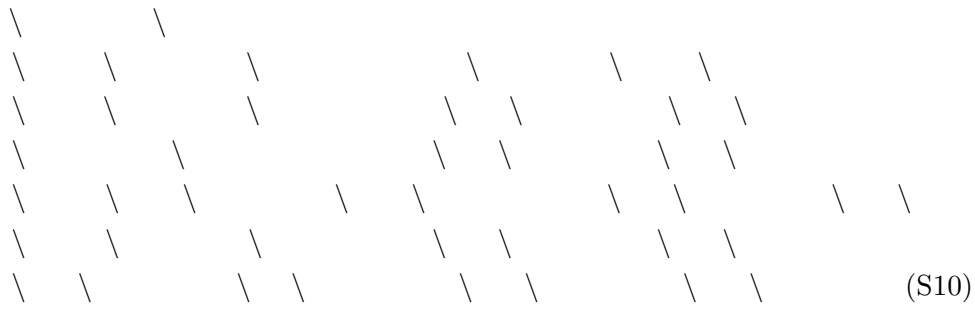
\ \

The left hand side term above is at least by (S5) and the right hand side term is by (S6). Thus . Which gives us a contradiction since, by our assumption, does not permit any alternating cycle.

Now consider the other case, \ . Define for this case \ \ . First we observe that . To see this, write

\ \
\ \ \ \ \ \ \
\ \ \ \ \ \
\ \ \ \ \ \ \ \
\ \ \ \ \ \ \
\ \ \ \ \ \ \ \ (S9)

Also,



(S10)

Notice that (S9) and (S10) are identical. As is a negative alternating cycle of we have

Finally, we shall show that is an augmenting path with respect to that starts at , i.e., $\mathcal{P}$. This will contradict the fact that is the minimal augmenting path with respect to .

We first argue that starts with . Let , an unmatched edge of . Consequently, is a matched edge in the augmented match . Also, no other edge in includes . There are now two possibilities, either there is an edge in that includes , or there is not. If there is no such edge, then . Otherwise, , and since is an alternating cycle with respect to , there is an edge in where is matched in and in . Thus, starts at . The same argument shows that if ends at , which is an unmatched node in then also ends at .

Lastly, we show that is an augmenting path with respect to . To show this, it suffices to show that if then (i) both and are matched in , else one is matched in and the other is unmatched in , i.e., or , and (ii) when and are not and , there is exactly one and one , both of which are matched nodes in so that and .

We argue in two cases, and assume that neither of is or .

Case 1; suppose but . Consider two sub-cases. First, suppose . Then, as is an alternating cycle, neither or are in . Also, the nodes and are not in . Hence, and are in . Next, suppose, . Since , . If possible, suppose and . One is matched in and the other is unmatched in . Wlog, suppose is matched in . As is the augmented match , must be in . Which implies that . So, and . Similarly, there is exactly one edge in , other than with the node .

Case 2; but . Since, is a cycle, there is exactly one and one so that and . At most one of them are in . For, implies either is a matched edge in both and , or it is an unmatched edge in both and . We consider these two sub-cases separately. If is matched in both and , neither nor are in . Because, if say was in , as

it is an unmatched edge in , it would be matched in . But, as is an augmenting path, there is also another such that is in and matched in . Which is a contradiction to the fact that the matched edge of in is not in .

Now suppose is unmatched in both and . Then consider the two edges and . Suppose, if possible, there is an edge in . Then there is also another edge . One of them, say the first one, is matched in and the other is unmatched in . As is an alternating cycle and is unmatched in , it is true that is in . So, and .

Thus, we have established that $\mathcal{P}$, an augmenting path with respect to that starts at . Thus, if is a negative alternating cycle with respect to , we have established a contradiction to the fact that is the shortest augmenting path with respect to by showing .

3 Algorithm to find the shortest augmenting path

In this appendix we present the algorithm of finding the shortest augmenting path $\mathcal{P}$ used in Algorithm 1 of the main text.

1. Input: a set of nodes; the cost function between any two vertices and given by ; node in that is unmatched in the existing match .

2. Initialization: Define vectors, of size each, `l`, `m`, `p`, `p_org`, `dm`, `dp` and `Pseudo`, and a scalar `Bls`. Make a non-redundant copy of as that would be modified within the algorithm. Set

- `l[ ]` = `'uncolored'` for and `l[ ]` = `'colored'`.
- `m[ ]` = `NULL` for all .
- `p[ ]` = for all ; `p_org` = `p`.
- `dm[ ]` = for all and `dm[ ]` = `-Inf`.
- `dp[ ]` = `Inf` for all .
- `Pseudo[ ]` = `FALSE` for all .
- `Bls` = .

Interpretation of the variables:

The interpretation of these objects in the algorithm is as follows.

- This algorithm builds a tree rooted at , and stops when it finds another node that is also unmatched in and is the next best option to be added to the tree.
- `l`: As more nodes are added to the tree those nodes are labeled as colored by the vector `l`.

- `p`: In the tree if a node is added before its matched node is added, vector `p` is used to find its parent node in the tree. If a node has not been added to the tree yet, thus `l[ ] = 'uncolored'`, `p[ ]` labels the node in the tree so that if were to be added to the tree attaching it to `p[ ]` would be the best.
- `m`: When a node is added that already has its matched node in the tree, the value of the vector `m[ ]` is used to identify that matched node of . The value of `m[ ]` is `NULL` for other nodes. Specifically, `m[ ] = NULL` and `m[Bls] = NULL` (see the final point about `Bls`).
- `dm`: For an uncolored node the value `dm[ ]` stores the best cost if were to be added to the tree at `p[ ]`.
- `dp`: When a node is added to the tree, `dp[ ]` is the best cost of the path traversed from to along the tree.
- `Pseudo`: In the process of building the tree, the algorithm might find a cycle which can be traversed any way with and the cost would remain the same. These would be a cycle of an odd number of edges. The algorithm then collapses the nodes in the cycle to one pseudo-node. When a pseudo-node is created the nodes in it are removed from . Initially, none of the nodes are pseudo nodes; thus, `Pseudo[ ] = FALSE` for all .
- `p_org`: `p_org` tracks the parent nodes of in the original nodes.
- `Bls`: `Bls` denotes the node or a pseudo-node in that includes the root node . At anytime, the tree is rooted at `Bls`.

In the following we use the notation to denote .

3. A while(`TRUE`) loop, that exists at .

Define:

`dm[ ]` `l[ ] 'uncolored'`

`dp[ ]` `dm[ ]` `l[ ] 'colored'`

Consider two switch cases — **Case 1:** and **Case 2:** .

First, for these two switch cases set

`dm[ ]` `l[ ] 'uncolored'` when **Case 1**,

or, `dp[ ]` `dm[ ]` `l[ ] 'colored'` when **Case 2**

Now for these two cases we do the following operations.

Case 1:

If is unmatched in then exit the while loop, go to **4.**

Otherwise, do the following.

- Find that is the matched vertex of in .
- Set `l[ ]='colored'`, `m[ ]=` , `dp[ ]` , and `l[ ]='inline'`.
- Check vertex for possible reassignment of `'uncolored'` nodes.
  - Scan for each `l[ ]` `'uncolored'` ,
  - if `dp[ ]`, `dm[ ]` set `dm[ ] =` `dp[ ]`, , and `p[ ] =` .

Case 2: Find two paths, using a backward search method, `Path 1` and `Path 2` in .

- `Path 1` is a path that starts at and ends at `Bls` so that when in the path, for all and `= p[ ]` or `= m[ ]`.
- `Path 2` is a path that starts at `p[ ]` and ends at `Bls` so that, as before, when in the path, for all and `= p[ ]` or `= m[ ]`.

In the path find the vertex so that is in both the paths and no other node that proceeds in `Path 1` is a node that proceeds in `Path 2`. Define a new path that first traverses `Path 1` backwards from to and then jumps to `Path 2` and traverses it forwards from `p[ ]` to the node preceding (so is traversed). Call this path . Notice that can also be thought of as a cycle. Also, starting from any node in we can find an alternating path to that needs an odd number of edges.

The following steps are used to collapse the nodes in into a new pseudo-node called `newnode`. It assigns appropriate values to the vectors `l`, `m`, `p`, `dm`, `dp` and `Pseudo` for the new node. The nodes in are removed from and adds `newnode`.

- Define a new node `newnode` and include it in . Set `B[newnode] =` .
- Remove the nodes in from .
- `l[newnode] = 'colored'`.
- `m[newnode] = m[ ]`.
- `p[newnode] = p[m[ ]]`.
- `dm[newnode] = Inf`.
- `dp[newnode] =` .
- `Pseudo[newnode] = TRUE`.
- If `m[newnode] = NULL` set `Bls = newnode`.

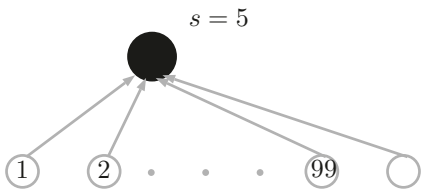


Figure S1: **Initialization.**

- `p[`t`]` = `newnode` for all t so that `p[`t`]` is a node in b.

The final step for **Case 2** is a reassignment step for the `'uncolored'` nodes within the new pseudo-node. The steps, which is reminiscent of the reassignment step at the end of the operations in **Case 1**, runs as follows

- For all $v' \in \{v' \mid v' \in V', \texttt{l[}v'\texttt{]} = \texttt{'uncolored'}\}$ do the following.
  - For $t \in V \cap \{\text{nodes in } b\}$ define $\texttt{dtemp[}t\texttt{]} = \max\{\texttt{dp[newnode]}, c[t, v']\}$.
  - If $dt := \min_{t \in V \cap \{\text{nodes in } b\}} \texttt{dtemp[}t\texttt{]} < \texttt{dm[}v'\texttt{]}$, set
    - $\texttt{dm[}v'\texttt{]} = dt$ and $\texttt{p[}v'\texttt{]} = \texttt{newnode}$.
    - $\texttt{p\_org[}v'\texttt{]} = \arg\min_{t \in V \cap \{\text{nodes in } b\}} \texttt{dtemp[}t\texttt{]}$.

4. Expanding the path from v to s along the tree:

Upon exiting **3** at $(\star)$, the final step of the algorithm is to find the path from v to s in the tree. This can be done in two steps. First, one can find the path from v to `Bls` through the nodes in V'. This is done easily using the vectors `p` and `m`. Second, as many of the nodes in V' might be pseudo-nodes, this path needs to be expanded to a path in V. For the second step one can use a recursive algorithm similar one in Derigs (1988). All the information required are stored in the variables `Bls` (the node/pseudo-node containing s), `p` (the parents in V' for the nodes in the tree), `p_org` (the parents in V for the nodes in the tree), `m` (the matched node for a node added to the tree after), and `B` (the structure of each pseudo-node). The resulting path is the shortest augmenting path $P_0 \in \mathcal{P}_s(M)$ from s to v.

The proposed algorithm has been implemented in `R`. This has been made available through the repository https://github.com/bikram12345k/BlockingAlgo.

3.1 An illustration.

In the following we give an illustration of a possible sequence of finding the shortest augmenting path. The illustration is drawn from a simulated data in an intermediate step of Algorithm 1 when the existing matched structure M has 35 matched pairs.

In Figure S1 the initialization step starts with $s = 5$, which is the root node of the tree. Figures S2 and S3 show the following three steps. Each step correspond to one round of

the while loop in the algorithm. In these two steps the tree grows from the root node to 3 nodes, from 3 nodes to 5 nodes and from 5 nodes to 7 nodes. In all these steps case 1 prevails over case 2. Figure S2, showing sub processes of step 1, highlights the growth of the tree, on the left, and then, on the right, the reassignment of the `'uncolored'` nodes to new colored node $u = 80$. The colored nodes are shown in the filled black circles. The nodes `'inline'` are shown in hollowed black circles. The `'uncolored'` nodes, that are not in the tree, can be attached later only to a `'colored'` node.

Later, in Figure S4, skipping a few steps, we show the tree after Step 7. The next step, Step 8 in Figure S5, processes Case 2 with $v = 44$ and `p[`v`]` $= 42$. `Path 1` is $44 \to 21 \to 84 \to 4 \to 42 \to 33 \to 24 \to 59 \to 80 \to 1 \to 5$, and `Path 2` is $42 \to 33 \to 24 \to 59 \to 80 \to 1 \to 5$. So, $b = 42 \to 4 \to 84 \to 21 \to 44$. In the center figure, the `'uncolored'` nodes are reassigned to the nodes $\{42, 4, 84, 21, 44\}$. These links are stored in `p_org`. Finally, in the right-most figure, the nodes in b are removed from V' and b is shrunk to a new pseudo-node.

The remaining steps of the algorithm are not shown. The algorithm creates eight pseudo nodes. They are `B1` $= 42 \to 4 \to 84 \to 21 \to 44$, `B2` $=$ `B1` $\to 32 \to 8 \to 55 \to 1$, `B3` $=$ `B2` $\to 19 \to 2 \to 12 \to 28$, `B4` $=$ `B3` $\to \to 72 \to 23 \to 50$, `B5` $=$ `B4` $\to 3 \to 39 \to 3 \to 47$, `B6` $=$ `B537` $\to 49$, `B7` $= 24 \to$ `B6` $\to 38 \to 88$, and `B8` $=$ `B7` $\to 34 \to 20 \to 4 \to 53$. The algorithm ends with finding the shortest path along the nodes/pseudo-nodes in V' as $57 \to B8 \to 59 \to 80 \to 1 \to 5$. This path is expanded to find the path along the original nodes V as $57 \to 49 \to 37 \to 21 \to 44 \to 42 \to 33 \to 24 \to 59 \to 80 \to 1 \to 5 = s$. For this expansion, three vectors `m`, `p`, and `p_org` are used. In this specific example these vectors are shown below. Notice that in expanding the path, if a pseudo-node is traversed, it is done in a direction that an even number the nodes/pseudo-nodes, equivalently, an odd number of edges are traversed.

```
 m =
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24   25   26
  NA "52"   NA   NA   NA   NA   NA  "9"   NA "30" "43" "28"   NA   NA   NA   NA   NA   NA   NA "34"   NA   NA   NA "59"   NA   NA
  27   28   29   30   31   32   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48   49   50   51   52
"18"   NA "40"   NA   NA   NA   NA   NA "13"   NA   NA   NA "36"   NA  "7" "33"   NA "21"   NA   NA  "3" "14" "37" "23"   NA   NA
  53   54   55   56   57   58   59   60   61   62   63   64   65   66   67   68   69   70   71   72   73   74   75   76   77   78
"46" "25"  "1"   NA   NA   NA   NA   NA   NA "19"   NA   NA   NA "26"   NA "32"   NA   NA   NA  "6"   NA   NA "45"   NA   NA   NA
  79   80   81   82   83   84   85   86   87   88   89   90   91   92   93   94   95   96   97   98   99  100   B1   B2   B3   B4
"22" "16"   NA   NA   NA  "4"   NA   NA   NA "38"   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA "33" "33" "33" "33"
  B5   B6   B7   B8
"33" "33" "59" "59"

p =
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24   25   26
"B8" "B8" "B8" "B8" "35" "B8" "B8" "80" "B8" "B8" "B8" "80" "B8" "B8" "B8" "56" "B8" "80" "B8" "B8" "B8" "B8" "B8" "27" "79" "79"
  27   28   29   30   31   32   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48   49   50   51   52
"80" "B8" "B8" "B8" "29" "B8" "B8" "B8" "80" "B8" "B8" "B8" "B8" "B8" "35" "80" "B8" "B8" "B8" "B8" "B8" "B8" "B8" "B8" "B8" "B8" "B8"
  53   54   55   56   57   58   59   60   61   62   63   64   65   66   67   68   69   70   71   72   73   74   75   76   77   78
"B8" "B8" "B8"   NA "B8" "66" "80"  "2" "B8" "B8" "11" "66" "79" "B8" "79" "B8" "10" "B8" "27" "B8" "B8" "B8"  "8" "B8" "B8" "79"
  79   80   81   82   83   84   85   86   87   88   89   90   91   92   93   94   95   96   97   98   99  100   B1   B2   B3   B4
"B8" "56" "B8" "10" "B8" "B8" "B8" "54" "66" "B8" "B8" "54" "B8" "75" "41" "B8" "B8" "B8" "B8" "B8" "B8" "B8" "B8" "B8" "B8" "B8"
  B5   B6   B7   B8
"B8" "B8" "80" "80"

p_org =
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24   25   26
 "4" "23" "39" "42" "35" "62" "21" "80" "39" "21" "50" "80" "42" "55" "68" "56" "47" "80" "68" "72" "84" "50" "72" "27" "79" "79"
  27   28   29   30   31   32   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48   49   50   51   52
"80" "21" "24" "12" "29" "84" "24" "44" "80" "32" "21"  "3" "44" "23" "35" "80" "24" "42" "12" "20" "21"  "4"  "4" "42" "28" "50"
  53   54   55   56   57   58   59   60   61   62   63   64   65   66   67   68   69   70   71   72   73   74   75   76   77   78
"21"  "6"  "4"   NA "49" "66" "80"  "2" "19" "12" "11" "66" "79" "19" "79" "55" "10" "21" "27" "62" "28" "47"  "8" "62" "68" "79"
  79   80   81   82   83   84   85   86   87   88   89   90   91   92   93   94   95   96   97   98   99  100
 "4" "56" "88" "10" "42" "24" "20" "54" "66" "24"  "1" "54" "68" "75" "41" "49" "12" "47" "68" "20" "36" "47"
```

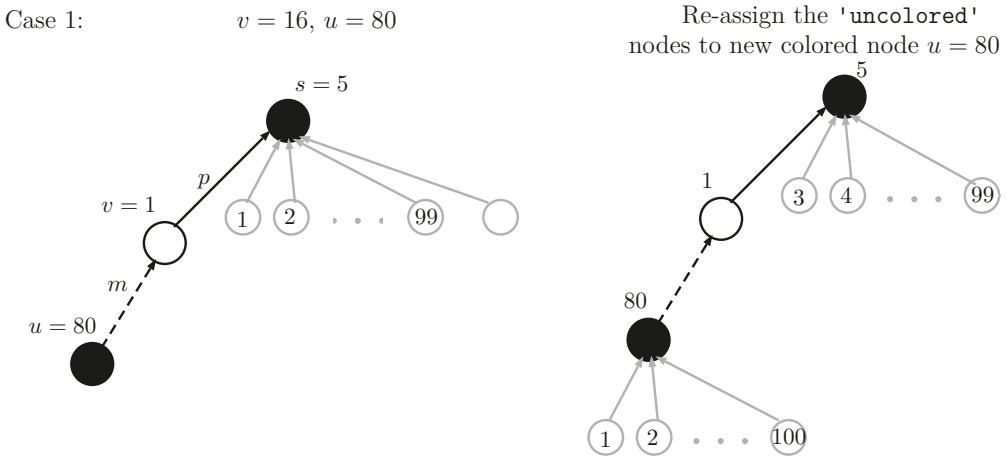


Figure S2: **Step 1.** The `'colored'` nodes are shown in solid black circle. The `'inline'` nodes are shown in hollowed black circle. The `'uncolored'` nodes are shown in hollowed gray circle. The tree built has a solid edge encoded in `p`, and a dashed edge encoded in `m`.

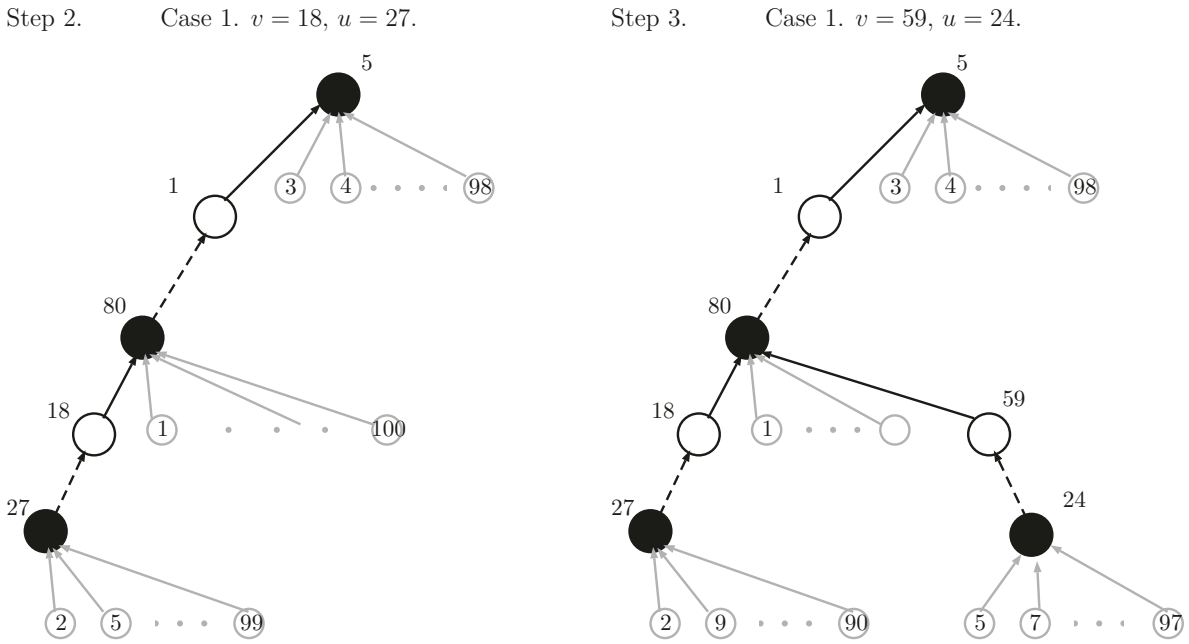


Figure S3: **Step 2 and 3.**

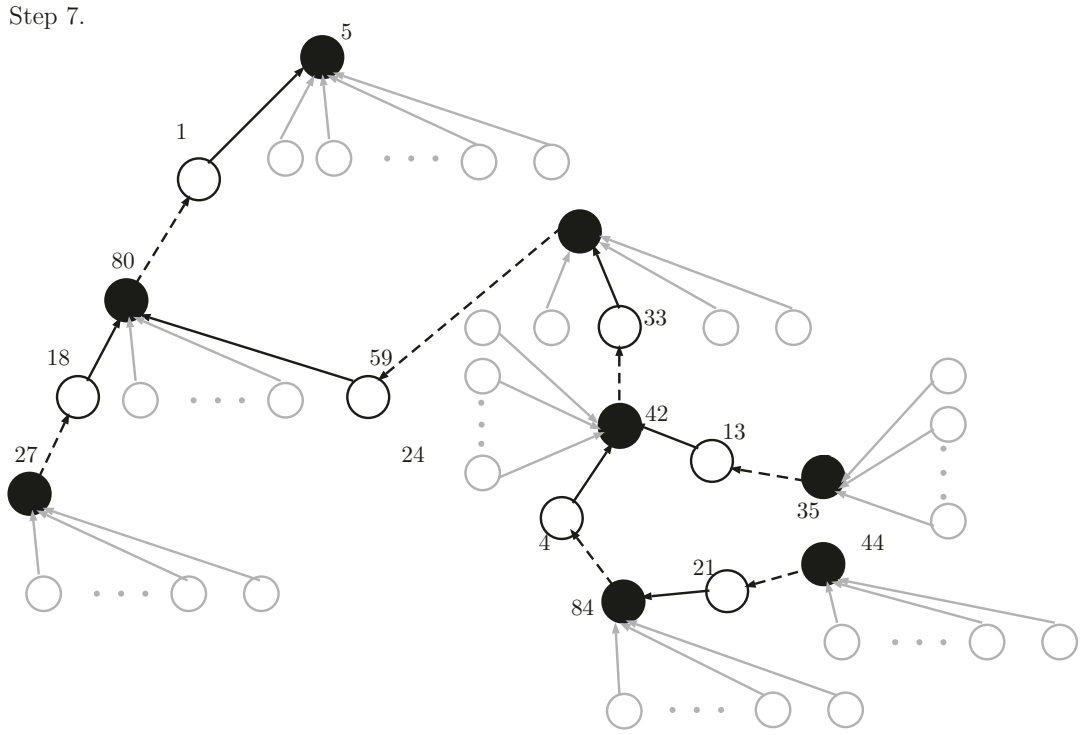


Figure S4: **Step 7.**

4 Proofs of other results

Proof of Theorem 3. Consider the following characterization of Algorithm 1 and Algorithm 2. Starting with cost matrix, square matrix of size $2m$, Algorithm 1 creates the matching M. If the nodes were labeled $1, \ldots, 2m$ then a matching is characterized by a bijective function, say ζ on $\{1, \ldots, 2m\}$ so that ζ creates a permutation of $\{1, \ldots, 2m\}$, i is matched to $\zeta(i)$, and $\zeta \circ \zeta =$Id; ζ has m transpositions.

The input to Algorithm 2 be the cost function on pairs of $N = n2^J$ units. Algorithm 2 calls Algorithm 1 J times. Let these J calls output the J mappings $\zeta_1, \ldots, \zeta_J$. Then, ζ_1 permutes $\{1, \ldots, N\}$ so that i is matched to $\zeta_1(i)$ and $\zeta_1 \circ \zeta_1 =$Id. For $x \in \{1, \ldots, N\}$, write

$$\{x\}_{\zeta_1} := \{x, \zeta_1(x)\}.$$

Suppose, ζ_2, the output to the second call to Algorithm 1, induces a permutation of $\{\{x\}_{\zeta_1} : x \in \{1, \ldots, N\}\}$. This permutation has degree N and consists of $N/2$ transpositions. Write

$$\{x\}_{\zeta_2 \circ \zeta_1} = \{x\}_{\zeta_1} \cup \zeta_2(\{x\}_{\zeta_1}).$$

Continuing this way suppose, ζ_j, the output to the second call to Algorithm 1, induces a permutation to $\{\{x\}_{\zeta_{j-1} \circ \ldots \circ \xi_1} : x \in \{1, \ldots, N\}\}$. The degree of ζ_j is $N/2^{j-1}$ so that

Step 8: Case 2, $v = 44$, $\mathtt{p}[v] = 42$

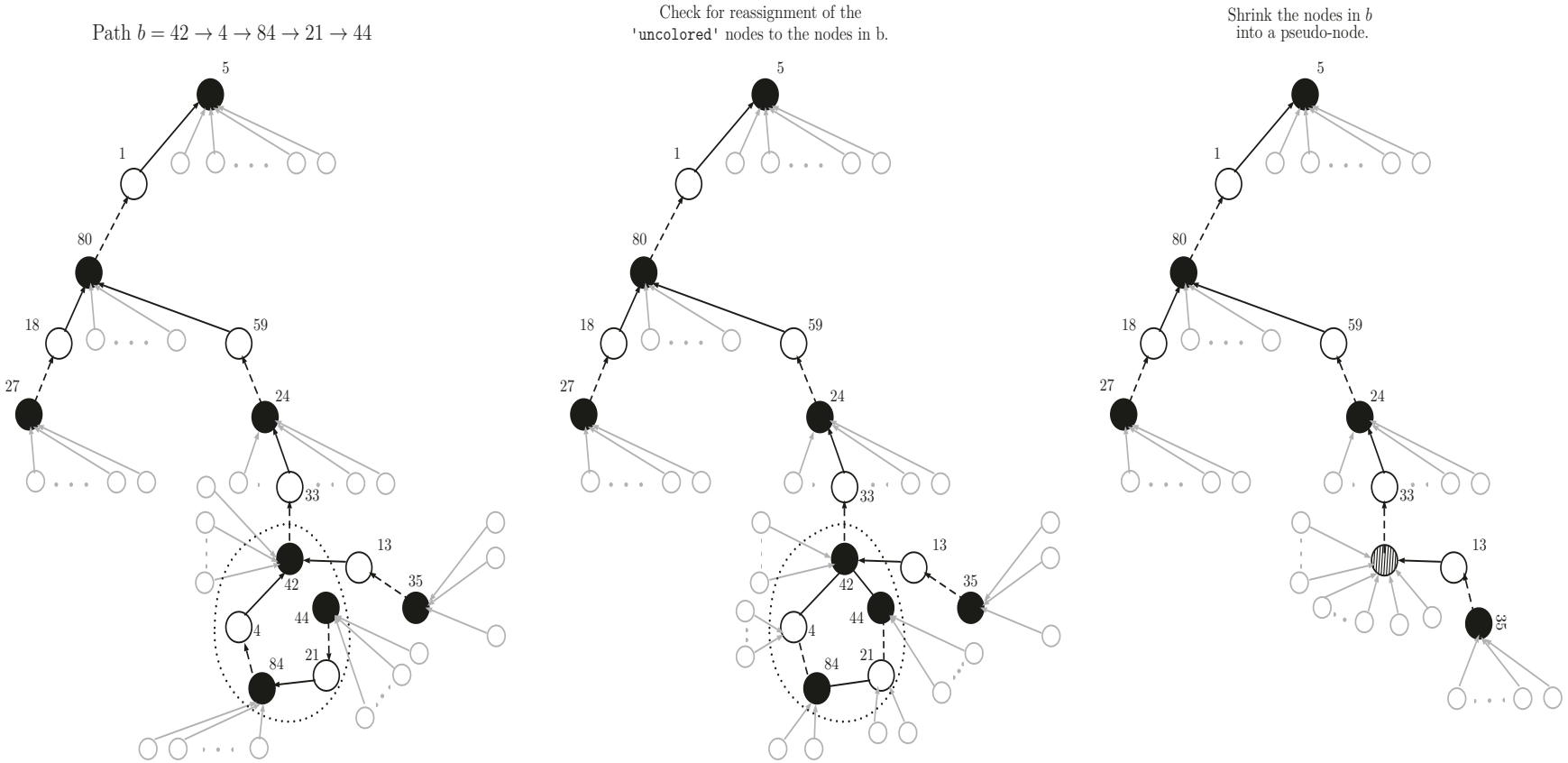


Figure S5: Step 8

is matched to . Write

For a set , define

Selection step. Let $\mathcal{B}$ be the optimal design that minimizes (1). For each $\mathcal{B}$, we do the following set of operations. Select distinct units so that for any and . It is possible to make this selection. First select arbitrarily from . Select from , and so on. Note that \ , so number of choices for is \ \ . Let . Define a one-to-one function

(S11)

arbitrarily. Notice that, by construction, .

Select a subset of in the following way. There are units in . For any two units we have . This selection can be done in a similar way as we did for . Select one unit of arbitrarily and select the second unit from . Notice that, for any , since for any other , by construction, , \ has at most one unit. Thus, after the selection of units for from there are ( )th unit is selected from at least \ units. Continue next to select from of units so that for any two units we have . Having selected , select so that

(1) ,

(2) , and

(3) for with , we have .

The sequence of subsets

creates a connection between the optimal design $\mathcal{B}$ and the output of Algorithm 2, $\mathcal{B}$.

Comparison step. for all $\mathcal{B}$. Design $\mathcal{B}$ has the form $\mathcal{B}$. Thus, the cost of design $\mathcal{B}$ is

$\mathcal{B}$ $\mathcal{B}$

To prove the theorem we need the following lemma.

Lemma S2 *For any*

$\mathcal{B}$ $\mathcal{B}$ $\mathcal{B}$

We complete the proof of the theorem using this lemma. Proof of Lemma S2 is given after. Using the lemma repeatedly we have that

$\mathcal{B}$

$\mathcal{B}$ $\mathcal{B}$

$\mathcal{B}$ $\mathcal{B}$ $\mathcal{B}$

$\mathcal{B}$ $\mathcal{B}$

Recall from (S12). Consider a pairing of units $\mathcal{B}$. Since, maps into and is one to one from , is a matching of the units . Now, as the first call to Algorithm 1 by Algorithm 2 finds the matching that is minimizes (1) and this matching is characterized by , we get that

$\mathcal{B}$ $\mathcal{B}$

Since , if is the cost of the optimal matching

$\mathcal{B}$ $\mathcal{B}$

$\mathcal{B}$ $\mathcal{B}$

$\mathcal{B}$

\ \

This completes the proof.

Proof of Lemma S2. Fix , . Note that, creates an optimal matching of $\mathcal{B}$. This is true by construction of , that are nonoverlapping for different and for , . In $\mathcal{B}$, there is another matching of $\mathcal{B}$. Within each , simply arbitrarily pair the units of . Call this function , which is a bijection on , so that is matched to . Thus, we get, by the optimality of ,

$\mathcal{B}$ $\mathcal{B}$

$\mathcal{B}$

Where

with . The term is the term that does not control for directly. It is the maximum difference across units that are in different blocks after the (\ )th call to Algorithm 1 by Algorithm 2.

Now, we control this term using the triangle inequality of the cost function.

$$\begin{aligned} B(\widetilde{b}) &\leq \max_{(x,x')\in T_{\widetilde{b}}^{(j-1)}} \max_{\substack{y\in\{x\}_{\zeta_{j-1}\circ\cdots\circ\zeta_1} \\ y'\in\{x'\}_{\zeta_{j-1}\circ\cdots\circ\zeta_1}}} (c_{xy} + c_{xx'} + c_{x'y'}) \\ &\leq 2 \max_{x\in S_{\widetilde{b}}^{(j-1)}} c(\{x\}_{\zeta_{j-1}\circ\cdots\circ\zeta_1}) + \max_{(x,x')\in T_{\widetilde{b}}^{(j-1)}} c_{xx'} \\ &\leq 2A(\widetilde{b}) + \max_{x,x'\in S_{\widetilde{b}}^{(j-1)}} c_{xx'}. \end{aligned}$$

The proof of the lemma is thus complete by noting that the cost function is nonnegative, thus, $A(\widetilde{b}) \leq 2A(\widetilde{b}) + \max_{x,x'\in S_{\widetilde{b}}^{(j-1)}} c_{xx'}$.

Proof of Proposition 4.

(a) First consider the blocking problem without and structure on the costs. We'll show a reduction of to the following problem.

Q1. Suppose $G = (V, E)$ is any graph with $|V| = nk$ nodes, and $H = (V', E')$ is another graph with $|V'| = k$. Can the vertices of G be partitioned into k disjoint sets $V_1, \ldots, V_n$ such that, for each i, the subgraph of G induced by V_i is isomorphic to H?

This is a problem whose answer is either 'Yes' or 'No'. Kirkpatrick and Hell (1978) showed that for any fixed H with $|V'| = k > 3$, the problem is NP-complete. In other words, given any fixed H there is no polynomial time algorithm that can answer this binary question for all graphs G.

Let H be a complete graph of k vertices. For a graph G we consider the following blocking problem of nk units. These nk units are identified with the nk vertices of G. Let the cost c_{ij} between two units i and j be 1 is the corresponding nodes are connected in E and be $(1 + 2\alpha)$ is they are not connected in E. Notice that, the solution to the blocking problem is 1, the maximum in-block differences, when and only when the answer to Q1 is 'Yes'. Only only other possible value of the blocking problem is $(1 + 2\alpha)$.

Suppose, there is a polynomial time algorithm that can solve the blocking problem at an approximation factor $(1 + \alpha)$. Suppose $\tilde{c} = 1$, i.e., the answer to Q1 is 'Yes', then $c^{\star\star} \leq (1 + \alpha) \times 1$. Since only two possible values of $c^{\star\star}$ are 1 and $(1 + 2\alpha)$, when $\tilde{c} = 1$ the approximation algorithm finds $c^{\star\star} = 1$. Also, when $\tilde{c} = (1 + 2\alpha)$, we cannot have $c^{\star\star} = 1$. Thus, the approximation algorithm will output $c^{\star\star} = 1$ if and only if the answer to Q1 is 'Yes'. Now using NP-completeness of the problem of finding the answer to Q1, we conclude that the blocking problem cannot be solved by a polynomial time algorithm, with an approximation factor $(1 + \alpha)$.

(b) Now consider the blocking problem with structure (4). As before, let H be a complete graph with k vertices. For any graph $G = (V, E)$ with $|V| = nk$ vertices, consider the correspondences of the nk vertices to nk units. The cost, c_{ij}, between two units i and j is $2\beta/\alpha$ if they are connected in E and it is β/α be they are not connected in E. As before note that answer to Q1 is affirmative if and only if the solution to this blocking problem is $\tilde{c} = 2\beta/\alpha$. Only two possible values are $2\beta/\alpha$ and β/α.

Now suppose an approximation is available so that the solution to the blocking problem by the approximation algorithm is $c^{\star\star} \leq (3-\alpha)\tilde{c} + \beta$. Then, $\tilde{c} = 2\beta/\alpha$ implies

$$c^{\star\star} \leq (3-\alpha)\tilde{c} + \beta = (3-\alpha)2\beta/\alpha + \beta = \ \beta/\alpha - \beta < \ \beta/\alpha.$$

Hence, when the answer to Q1 is 1 or equivalently, $\tilde{c} = 2\beta/\alpha$, the approximation algorithm will also achieve the solution $c^{\star\star} = 2\beta/\alpha$. On the other hand, $c^{\star\star} \neq 2\beta/\alpha$ when $\tilde{c} \neq 2\beta/\alpha$. Thus, if such an approximation algorithm exits to solve the blocking problem, then using the above reduction, we can answer Q1 for any graph G. But, Q1 is a NP-complete problem. Hence the proof.

Proof of Theorem 5. We borrow the notation in the proof of Theorem 3. Suppose the locally optimal design is $\mathcal{B}' \in \mathcal{L}(\mathcal{B})$, where $\mathcal{B}$ is the output of Algorithm 2. Suppose $x_{b'} \in \mathcal{B}'$ is such that $x_{b'}$ is not in the same block as the rest of the units of b' in $\mathcal{B}$. Make the selection of $S_{b'}^{(1)}$ such that $x_{b'} \notin S_{b'}^{(1)}$. This is always possible, since, as discussed in the proof of Theorem 3, there are two choices for the selection of the last unit of the set $S_{b'}^{(1)}$.

$$\max_{b\in\mathcal{B}} \max_{i,j\in b; i\neq j} c_{ij} = \max_{b'\in\mathcal{B}'} \max\{ \max_{x\in S_{b'}^{(1)}} c_{x,\zeta_1(x)}, \max_{x,x'\in S_{b'}^{(1)}} c_{x,x'}, \max_{x,x'\in S_{b'}^{(1)}} c_{\zeta_1(x),\zeta_1(x')}\}.$$

Suppose, $x_{b'}^{\star} \in S_{b'}^{(1)}$ is such that $\zeta_1(x_{b'}^{\star}) \notin b'$. We have, by the triangle inequality

$$c_{x,\zeta_1(x_{b'}^{\star})} \leq c_{x,x_{b'}^{\star}} + c_{x_{b'}^{\star},\zeta_1(x_{b'}^{\star})}.$$

Finally, using this fact and the optimality of ζ_1 we get that

$$\max_{b\in\mathcal{B}} \max_{i,j\in b; i\neq j} c_{ij} \leq 2 \cdot \max_{b'\in\mathcal{B}'} \max_{i,j\in b'; i\neq j} c_{ij}.$$

Proof of Theorem 6. The first step to prove the theorem is to follow the 'selection step' of the proof of Theorem 3. Following that discussion let $\xi_1, \ldots, \xi_J$ be the functions that characterizes the outputs to the J calls to Algorithm 1 by Algorithm 2 for (7).

Let, $N = n2^J$. Write for $x \in \{1, \ldots, N\}$

$$\{x\}_{\xi_1} := \{x, \xi_1(x)\},$$

$$\{x\}_{\xi_j \circ \ldots \circ \xi_1} = \{x\}_{\xi_{j-1}\circ\ldots\circ\xi_1} \cup \xi_j(\{x\}_{\xi_{j-1}\circ\ldots\circ\xi_1}).$$

For a set $A \subseteq \{1, \ldots, N\}$, define

$$c_s(A) := \sum_{i,j\in A} c_{ij}.$$

Selection Step. Let $\widetilde{\mathcal{B}} = \{\widetilde{b}_1, \ldots, \widetilde{b}_n\}$ be the optimal design that minimizes (7). For

each $\mathcal{B}$, we do the following set of operations. Select distinct units so that for any and . Let . Define a one-to-one function

(S12)

arbitrarily. Notice that, by construction, . Having selected , for , select so that

(1) ,

(2) , and

(3) for with , we have .

Optimality of . For each , arbitrarily pair the units of . Call this function , which is a bijection on , so that is paired to . Using these 's define , a pairing of the units of $_{\mathcal{B}}$ so that pairs units in for all $\mathcal{B}$ according to .

Since optimaly pairs the units in $_{\mathcal{B}}$ we have

$_{\mathcal{B}}$

$_{\mathcal{B}}$

(S13)

$_{\mathcal{B}}$

Comparison step. Since and for $_{\mathcal{B}}$ with , by construction . Thus, if $\mathcal{B}$ is the output of Algorithm 2 for (7) then

$_{\mathcal{B}}$ — — $_{\mathcal{B}}$ (S14)

This establishes the connection between the cost of the design $\mathcal{B}$ and the optimal design

$\mathcal{B}$. By (S14) and (S13)

$$\mathcal{B}$$

$$\mathcal{B}$$

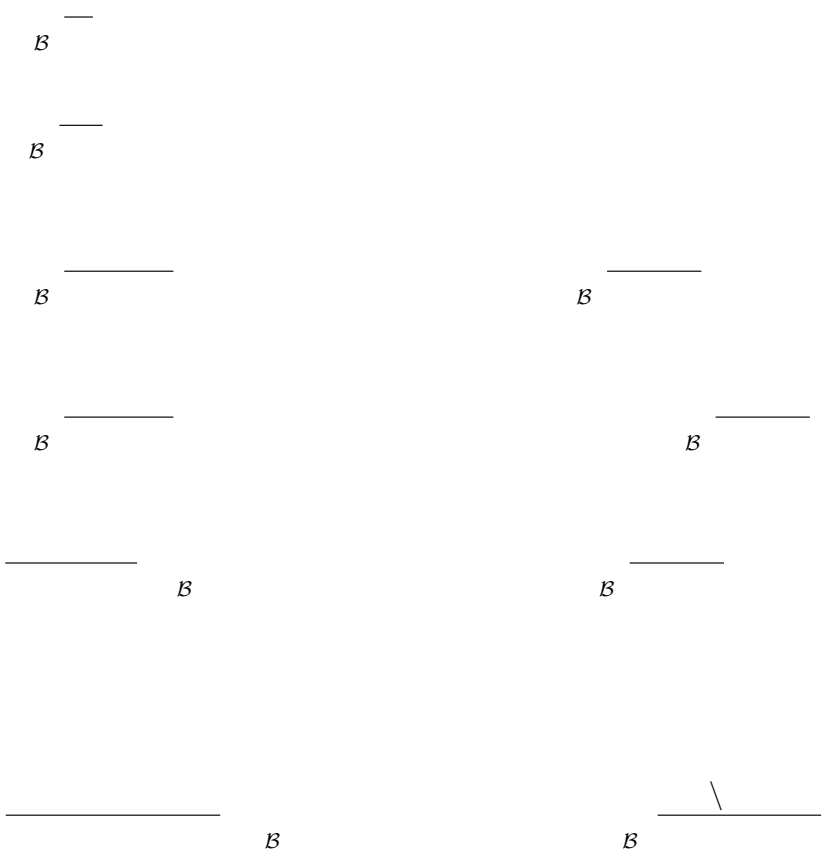
$$-$$

$$\mathcal{B}$$

$$\mathcal{B}$$

By the triangle inequality for ,

Thus



(S15)

where , and . In the first term now use (S13) to replace by .

For the coefficient for the first term in (S15) we write

$$\begin{aligned}
&\prod_{j=1}^{J-1}(2^j + 2^{J-j}) \\
&\leq \Big[\frac{1}{J-1}\sum_{j=1}^{J-1}(2^{J-j} + 2^j)\Big]^{J-1} \\
&= \Big[\frac{1}{J-1}\{2^J(1-2^{-J+1}) + 2(2^{J-1}-1)\}\Big]^{J-1} \\
&= \Big[\frac{1}{J-1}\{k(1-2/k) + 2(k/2-1)\}\Big]^{J-1} \\
&= (k-2)^{J-1}(2/(J-1))^{J-1} \leq 8(k-2)^{J-1}/k
\end{aligned}$$

The inequality above follows from the fact that geometric mean $\leq$ arithmetic mean. The last inequality follows since $(2/(J-1))^{J-1} \leq 8 \cdot 2^{-J}$. This term can by dispersed into $(k-1)$ terms, which gives us a factor

$$8(k-2)^{J-1}/(k(k-1)) \leq 8(k-2)^{J-3}.$$

The jth term in (S15) in the sum can be dispersed into $(2^J - 2^{J-j} + 1)/(2^{J-j} - 1) = 2^{J+j}/(2^J - 2^j) - 1 \geq 2^j - 1$ terms. Use the GM $\leq$ AM inequality to write

$$p(j;J)^{1/(j-1)} \leq \frac{1}{j-1}(2^j - 2 + 2^J - 2^{J-j+1}) \leq \frac{1}{j-1}(3 \cdot 2^{J-1} - \ ) = \frac{1}{j-1}(3/2)(k-4). \quad \text{(S16)}$$

To get the second inequality above we have used $j = J-1$.

Using (S16) and that $(1/(j-1))^{j-1} \leq 2 \cdot 2^{-(j-1)}$ we get

$$\begin{aligned}
\sum_{j=1}^{J-1}\frac{p(J-j;J)4^j}{2^j-1} &\leq \sum_{j=1}^{J-1} 2 \cdot 2^J\{(3/8)(k-4)\}^{j-1} \\
&\leq 2 \cdot 2^J \frac{(3/8)^{J-1}(k-4)^{J-1}}{(3/8)(k-4)-1} \\
&= 2 \cdot 2^J (3/8)^{J-2}\frac{(k-4)^{J-1}}{(k-20/3)} \\
&\leq 2 \cdot 4 \cdot (3/4)^{J-2}\frac{(k-4)^{J-1}}{(k-7)} \\
&\leq 2 \cdot 4 \cdot (3/4)^2 \cdot 1 \cdot \frac{(k-4)^{J-1}}{(k-7)} \\
&= 9/2\frac{(k-4)^{J-1}}{(k-7)} \\
&\leq 9/2 \cdot k(k-4)^{J-3}
\end{aligned}$$

For , from (S15) and with the dispersion terms, the approximation factor is

\ \

For ,

\ \

5 Comparison using simulations

In this section we evaluate the performance of the proposed methods and compare them to the available methods in different simulation scenarios. Including Scenario 1 and Scenario 2 of §4.3, three simulation scenarios are considered. The third simulation scenario has two binary and four continuous variables, and these variables are correlated. To simulate such a structure a 6 dimensional random vector is simulated from a mixture of three multivariate normal distributions with mixing proportions 1/2, 3/8 and 1/8 respectively. Each mixing component has the same covariance structure: the variance of each variable is 1 and the covariance of any two variables is 0.3. The normal mean vectors for the three components are $(0,0,0,0,0,0)^\top$, $(0,0,2,2,-2,-2)^\top$, and $(0,0,5,5,-\ ,-\ )^\top$, respectively. Finally, the first two values of the vector, say x_1 and x_2, are discretized as: $x_1^* = 1$ if $x_1 < 0.1$ and $x_2^* = 1$ if $x_2 > 0.1$, and 0 otherwise. The cost function used in this scenario is the square root of rank based robust Mahalanobis distance (Rosenbaum, 2010, §8.3).

We compare eight methods for $k = 4$, as listed in Table S5 and Table S6. The first four are: Algorithm 2 with and without its improvement using Algorithm 3 for the two problems (1) and (7). The other four methods are the available methods in the literature, see §1.3. The 'Random templating and assignment' method is the randomized algorithm of that creates many blockings by randomly selecting one template unit for each block, and outputs a blocking that is the best among those. The number of blockings it creates is a parameter of the algorithm. In the simulations, we set this number to 30, a suggestion by Karmakar (2018). The next two methods, method 6 and 7 in the tables, are the 'threshold blocking' method. For a given threshold k, this method finds a blocking of the units into blocks of at least k units. In the simulation, the threshold blocking method is evaluated for two threshold values, $k = 3$ and $k = 4$. The last of the methods in the tables is the 'optimal greedy' method. As of the latest available implementation of this method, unless a data set is provided this method cannot create a blocking. Thus, for Scenario 1 the results of this method are not available.

In the tables, two measures are reported. The first measure is the maximum of largest in-block paired differences. Second is the maximum of the average in-block paired differences. They correspond to the objective functions of problems (1) and (7) respectively.

In Table S5, the numbers are percentages with respect to the baseline algorithm, Algorithm 2 for problem (1), in the first row. A smaller value in Table S5 represents a better performance by the method. Consider first Scenario 1. In this scenario, compared to Algorithm 2 for (1) we do not see any improvement by additionally using Algorithm 3. Although, we do observe improvements in both the measures for Algorithm 2 for (7) by using Algorithm 3. The threshold blocking method shows identical performance for both choices of the threshold. Further, they are considerably poorer compared to the random templating method, which is closer to the methods proposed in this paper, but still inferior to all of them.

Scenarios 2 and 3 create Mahalanobis distances from two different kinds of multivariate data sets. In these two scenarios using Algorithm 3 shows a marginal improvement over Algorithm 2 for (1) — a comparatively better improvement for Scenario 3 over Scenario 2. The ranking of the last four methods by their performances is the same order in which they

Table S1: Table of the values. Shows the performance in percentage terms, compared to the baseline, the first row averaged over 500 simulations in each scenario. Two performance measurements are considered, and three simulation scenarios are considered (described in the text). The pair of numbers in parenthesis are the 10% and the 90% quantiles. A smaller value in the table represents better performance; the smallest one(s) in each column is highlighted in bold.

Measurement	Max largest in-block differences			Max average in-block differences		
	Scenario			Scenario		
	1	2	3	1	2	3
Algorithm 2 for (1) (baseline)	**100**	100	100	100	100	100
Algorithm 2+3 for (1)	**100** (100, 100)	**99** (98, 100)	**97** (91, 100)	100 (100, 100)	100 (100, 100)	100 (96, 103)
Algorithm 2 for (7)	121 (100, 131)	104 (100, 114)	111 (101, 121)	85 (79, 90)	96 (88, 100)	94 (86, 100)
Algorithm 2+3 for (7)	119 (110, 130)	106 (100, 119)	116 (104, 127)	**82** (77, 88)	**94** (83, 100)	**89** (81, 96)
Random templating and assignment	143 (127, 184)	133 (101, 170)	131 (114, 148)	103 (88, 136)	123 (97, 158)	117 (103, 132)
Threshold blocking with	202 (194, 209)	139 (96, 193)	139 (114, 166)	265 (248, 284)	139 (97, 189)	136 (112, 164)
Threshold blocking with	202 (194, 209)	168 (119, 225)	152 (127, 180)	265 (248, 284)	146 (104, 189)	138 (118, 163)
Heuristic blocking	–	320 (206, 431)	194 (158, 233)	–	309 (198, 435)	181 (148, 218)

Karmakar (2018); Higgins et al. (2016); Moore (2012).

are listed in the table. The very last method, the heuristic blocking method (the optimal greedy method of Moore, 2012), performs very poorly; in certain instances the values are 430% of the baseline, on an average 320% and 309% of the baseline for the maximum largest in-block difference and maximum average in-block difference respectively. The randomized blocking method again performs better than the threshold blocking method — only marginally better in the first but considerably better in the second measurement.

In the simulation, for the three scenarios, the threshold blocking method with threshold has average block sizes 5.8, 3.9 and 4 respectively, and with has average block sizes 9.1, 5.4 and 5.7 respectively.

Algorithm 2 for (7) with improvements using Algorithm 3, gives the best performance in the simulation, when we look at the maximum average in-block difference. This method also fares well for the maximum largest in-block difference, compared to the best algorithm.

Table S6 complements Table S5. In each scenario, this table reports the percentage of the simulation instances a method is the winner, in the corresponding measure, among the 8 methods compared. The columns in the table do not sum to 100% because multiple methods can have the best value of a measurement in a simulated instance. It is clear from Table S6 that proposed Algorithm 2 with 3 for (1), the second method in the table, has the most wins for the maximum of the largest in-block differences and the proposed Algorithm

Table S2: Table of the winners. In each scenario and for the two measurements, the percentage of times the method performs the best among all the methods. In each column the best one(s) are highlighted in bold.

Measurement	Max. largest in-block difference			Max. average in-block difference		
	Scenario			Scenario		
	1	2	3	1	2	3
Algorithm 2 for (1)	**100**	70	56	0	28	2
Algorithm 2+3 for (1)	**100**	**82**	**99**	0	29	2
Algorithm 2 for (7)	0	41	4	15	58	22
Algorithm 2+3 for (7)	0	34	2	**100**	**88**	**100**
Random templating	0	7	0	0	6	0
Threshold blocking with	0	15	0	0	8	0
Threshold blocking with	0	1	0	0	3	0
Heuristic blocking	–	0	0	–	0	0

2 with 3 for (7), the fourth method in the table, has the most wins for the maximum of the average in-block differences. Only in scenario 2, we see a few wins for the random templating method and the threshold blocking method; no wins for the heuristic blocking method. In scenarios 2 and 3, we can see noticeable improvements by using Algorithm 3. In scenario 2, Algorithm 3 improves Algorithm 2 for (1) in at least (82\ 70)=12% of the instances, and in scenario 3, in at least (99\ 56)=43% of the instances. Algorithm 3 also improves Algorithm 2 for (7), in at least (100\ 22)=78% of the instances in Scenario 3, in at least (88\ 58)=30% of the instances in Scenario 2, and in at least (100\ 15)=85% of the instances in Scenario 1.

6 Additional simulation results

This section compares the different methods of blocking to create blocks of size and , while the main text presented the results for . Three simulation scenarios are considered and they have been discussed in the main text. The conclusions of the simulation results of for still holds. Algorithm 3 provides a sufficient improvement over Algorithm 2 in all three simulation scenarios, more for Scenarios 2 and 3. For minimizing the maximum of the block-wise worst pair, Algorithm 2+3 for (1) outperforms all other methods in all but one situation. The randomized templating method of Karmakar (2018) performs best for and scenario 2. In this case, the random templating method also outperforms other methods in the measure of maximum of the average in-block differences. This simulation is an exception in some of the remarks below as well.

The heuristic blocking method Moore (2012) fairs very poorly in every simulation instance and measure of comparison. Thresholds blocking method comes close second in all but the above one exception. We note that for the threshold blocking algorithm with threshold created blockings with average block size , and for the three simulation scenarios respectively. For the other set of simulations with , the threshold

blocking algorithm with threshold 8 creates blockings with average block size 31.3, 11. and 13.1 for the three simulation scenarios respectively. In the same simulations, a threshold of 7 creates blockings with average block size 25.3, 10 and 11.2 for the three simulation scenarios respectively. These averages are over 500 simulated instances for each scenario.

Table S3: k=3. Table of the values. Compared to the baseline, the first row, the value of the measurement in percentage terms, averaged over 500 simulations in each scenario. Two performance measurements are considered, and three simulation scenarios are considered (described in the text). The pair of numbers inside a parenthesis represents the 10% and the 90% quantile of the value out of the 500 iterations. A smaller value in the table represents better performance; the smallest one(s) in each column is highlighted in bold.

Measurement	Max largest in-block difference			Max average in-block difference		
	Scenario			Scenario		
	1	2	3	1	2	3
Algorithm 2 for (1) (baseline)	**100**	100	100	100	100	100
Algorithm 2+3 for (1)	**98** (93, 100)	91 (69, 100)	**95** (83, 100)	98 (94, 100)	91 (69, 100)	97 (87, 100)
Algorithm 2 for (7)	108 (100, 121)	104 (85, 124)	101 (89, 109)	100 (94, 107)	103 (85, 125)	97 (87, 105)
Algorithm 2+3 for (7)	107 (99, 120)	91 (69, 101)	98 (85, 106)	**97** (92, 100)	91 (68, 100)	**93** (83, 100)
Random templating and assignment†	138 (116, 181)	**80** (53, 109)	112 (92, 133)	124 (105, 177)	**84** (56, 115)	119 (99, 137)
Threshold blocking†† with $k = 3$	192 (181, 201)	94 (58, 136)	127 (104, 155)	272 (252, 284)	92 (56, 134)	133 (108, 164)
Heuristic blocking†††	–	187 (108, 276)	169 (130, 209)	–	205 (112, 303)	185 (144, 228)

† Karmakar (2018); †† Higgins et al. (2016); ††† Moore (2012).

Table S4: k=3. Table of the winners. In each scenario and for the two measurement criterion, the percentage of times the method performs the best among all the methods. A larger value implies better performance, and in each column the best one(s) are highlighted.

Measurement	Max. largest in-block difference			Max. average in-block difference		
	Scenario			Scenario		
	1	2	3	1	2	3
Algorithm 2 for (1)	68	9	51	15	9	20
Algorithm 2+3 for (1)	**100**	16	**91**	19	15	20
Algorithm 2 for (7)	9	6	17	52	10	63
Algorithm 2+3 for (7)	10	11	18	**100**	20	**98**
Random templating	0	**60**	8	0	**46**	2
Threshold blocking with	0	27	1	0	37	0
Heuristic blocking	–	0	0	–	0	0

Table S5: k=8. Table of the values. Compared to the baseline, the first row, the value of the measurement in percentage terms, averaged over 500 simulations in each scenario. Two performance measurements are considered, and three simulation scenarios are considered (described in the text). The pair of numbers inside a parenthesis represents the 10% and the 90% quantile of the value out of the 500 iterations. A smaller value in the table represents better performance; the smallest one(s) in each column is highlighted in bold.

Measurement	Max largest in-block difference			Max average in-block difference		
	Scenario			Scenario		
	1	2	3	1	2	3
Algorithm 2 for (1) (baseline)	**100**	100	100	100	100	100
Algorithm 2+3 for (1)	**100** (100, 100)	**98** (92, 100)	**97** (92, 100)	100 (100, 100)	99 (95, 100)	100 (98, 104)
Algorithm 2 for (7)	121 (111, 131)	105 (96, 117)	117 (107, 128)	85 (80, 90)	94 (81, 103)	90 (84, 97)
Algorithm 2+3 for (7)	122 (113, 131)	129 (100, 158)	130 (117, 146)	**84** (79, 89)	**87** (74, 100)	**84** (78, 91)
Random templating and assignment	129 (120, 135)	129 (100, 164)	126 (112, 142)	99 (90, 107)	113 (90, 141)	111 (100, 123)
Threshold blocking with	175 (163, 188)	169 (123, 220)	153 (132, 178)	147 (87, 198)	146 (106, 193)	141 (123, 163)
Threshold blocking with	175 (162, 189)	180 (133, 232)	158 (137, 183)	134 (85, 181)	150 (113, 194)	143 (126, 163)
Heuristic blocking	–	308 (231, 382)	192 (162, 220)	–	297 (200, 389)	178 (149, 206)

Karmakar (2018); Higgins et al. (2016); Moore (2012).

Table S6: k=8. Table of the winners. In each scenario and for the two measurement criterion, the percentage of times the method performs the best among all the methods. A larger value implies better performance, and in each column the best one(s) are highlighted in bold.

Measurement	Max. largest in-block difference			Max. average in-block difference		
	Scenario			Scenario		
	1	2	3	1	2	3
Algorithm 2 for (1)	**100**	59	49	0	11	2
Algorithm 2+3 for (1)	**100**	**81**	**100**	0	12	0
Algorithm 2 for (7)	0	34	0	46	26	5
Algorithm 2+3 for (7)	0	12	0	**91**	**92**	**100**
Random templating	0	9	0	0	4	0
Threshold blocking with	0	0	0	5	0	0
Threshold blocking with	0	0	0	4	1	0
Heuristic blocking	–	0	0	–	0	0

References

Bind, M. A. and Rubin, D. B. (2020). When possible, report a Fisher-exact P value and display its underlying null randomization distribution. . *Proceedings of the National Academy of Sciences*, **117**, 19151–19158. (Not cited.)

Box, G. E.P. and Hunter, J. S., (1961). The fractional factorial designs, *Technometrics*, **3**, 311-351. (Not cited.)

Cheng, C. S. and Mukerjee, R. (2001). Blocked regular fractional factorial designs with maximum estimation capacity. *Annals of Statistics*, **29**, 530–548. (Not cited.)

Derigs, U. (1988). Solving nonbipartite matching problems via shortest path techniques. *Annals of Operations Research,* **13**, 225–261. [13].

Edmonds, J. (1965a). Paths, trees, and flowers. *Canadian Journal of Mathematics*, **17**, 449–467. (Not cited.)

Edmonds, J. (1965b). Maximum matching and a polyhedron with 0, 1-vertices. *Journal of Research of the National Bureau of Standards B. Mathematics and Mathematical Physics*, **69**, 125–130. (Not cited.)

Federer, W. T. and King, F. (2007). *Variations on split plot and split block experiment designs* . John Wiley & Sons, Hoboken, NJ. (Not cited.)

Higgins, M. J., Sävjeb, F. and Sekhon, J. S. (2016). Improving massive experiments with threshold blocking. *Proceedings of the National Academy of Sciences*, **113**, 7369–7376. [29, 31, and 32].

Karmakar, B. (2018). blockingChallenge: Create blocks or strata which are similar within. *CRAN*, R package version 1.0. [28, 29, 30, 31, and 32].

Kelcey, B., Spybrook, J., Phelps, G., Jones, N. and Zhang, J. (2017). Designing large-scale multisite and cluster-randomized studies of professional development, *THe Journal of Experimental Education*, **85**, 389–410. (Not cited.)

Kirkpatrick, D. G. and Hell, P. (1978). On the completeness of a generalized matching problem. In *Proceedings of the tenth annual ACM symposium on Theory of computing* pp. 240–245. [21].

Moore, R. T. (2012). Multivariate continuous blocking to improve political science experiments. *Political Analysis*, **20**, 460–479. [29, 30, 31, and 32].

Robinson, J. (1970). Blocking in incomplete split plot designs, *Biometrika*, **57**, 347–350. (Not cited.)

Rosenbaum, P. R. (2010). *Design of Observational Studies*. New York: Springer. [28].

Rubin, D. B. and Thomas, N. (2000). Combining propensity score matching with additional adjustments for prognostic covariates. *Journal of the American Statistical Association*, **95**, 573–585. (Not cited.)