Optimal Matched Block Design For Multi-Arm Experiments*

Nathan Brixius**

Airbnb, Inc.
San Francisco, CA

Abstract. We introduce new techniques for matched block design in multi-arm experiments. In matched block design, units with similar covariate values are grouped into blocks, with one unit per treatment in each block. Existing methods for unit-level block design fail to produce optimal matches for multi-arm experiments. We present a mixed integer programming (MIP) formulation that guarantees optimal solutions for the general multi-arm matched blocking problem using a clique-based equipartitioning approach. For cases where the MIP is computationally infeasible, we introduce heuristics that decompose large problems into tractable subproblems while providing explicit quality-runtime tradeoffs. We demonstrate that our methods significantly outperform existing techniques on a diverse test suite, achieving consistent improvements in block balance quality. Additionally, we show how these methods can be adapted to improve covariate balance across treatment groups. We demonstrate that matched block design presents an interesting application area for metaheuristic and exact optimization methods.

Keywords: matched block design, optimization, mixed integer programming, randomized experiments

1 Introduction

Determining causal effects is fundamental to evidence-based decision making. Randomized experiments, where subjects are assigned to different treatment groups, remain the gold standard for establishing causality.

However, in many real-world settings, subject-level treatment assignment is often impractical or impossible. Instead, subjects must be collected into groups called units, which then become the level of assignment. In a retail experiment, individual stores might serve as units rather than employees [18]; in educational research, schools rather than students [17]; and on digital platforms, geographic regions rather than individual users.

A key goal in experiment design is to achieve balance in unit covariates across treatment groups. In randomized experiments, balance is ensured if there are sufficient units. When sample sizes are limited or when greater precision is desired,

* Submitted February 13, 2025, revised May 14, 2025.
** nathan.brixius@airbnb.com

alternative approaches are needed. Block design addresses this challenge by explicitly considering unit covariates to form matched groups across treatments. In this approach, units with similar characteristics are grouped into blocks, with each block containing one unit for each treatment.

Software tools for block design include `blocktools` [23], `mipmatch` [32], and `brsmatch` [20]. Such implementations account for for balance within and across blocks, as well as computational runtime. These considerations in turn depend on factors such as the number of treatments, experimental units, and covariates.

Let n be the number of experimental units, and t be the number of treatments, i.e. block size. Without loss of generality we assume $n \mod t = 0$ and block count $m = \frac{n}{t}$. Figure 1 shows the relationship between subjects, units, blocks, and treatments.

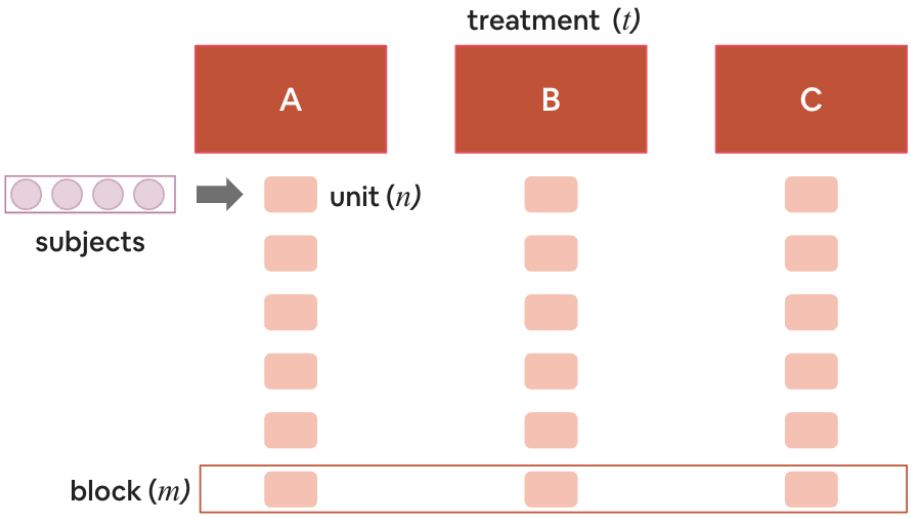

Fig. 1. Subjects, Units, Blocks, and Treatments

Balance can be assessed by comparing unit covariates within and across blocks. Two types of measures for block design are block balance and treatment balance. By block balance we mean that units in a block are similar, as determined by a covariate distance metric, e.g. Mahalanobis distance. By treatment balance we mean that the covariate distributions among treatment groups are similar.

Our primary focus in this work is block balance, as assessed by the sum of all pairs of distances between matched pairs within blocks. We refer to this problem as Matched Blocking (MB), and the special case of $t = 2$ as Pair Blocking (PB).

The primary contribution of this work is a new mixed integer programming formulation of MB. This formulation can be solved to optimality in many scenarios, as demonstrated across a set of test instances from the literature. While

our method produces optimal matches, it is not computationally feasible in all situations. Therefore a second contribution is a heuristic framework capable of balancing runtime and block balance considerations, leveraging the strengths of our MIP formulation. We show that our techniques outperform commonly used methods over a test suite that includes both previously considered and newly created instances.

The remainder of this work is organized as follows. We first consider PB, where efficient methods for optimal matches are well known. We then turn to MB, where such is not the case. While MB approaches often rely on heuristics that do not provide guarantees of match quality, we show how an extension of a mixed integer programming (MIP) formulation for PB can be extended to MB. This formulation can be solved to optimality across most of the instances in our test suite. We then consider a heuristic framework that trades optimality for faster runtime.

While block balance is our primary concern, we address treatment balance in Section 5. We show that even a simple postprocessing heuristic approach to treatment balance yields significantly improved covariate balance.

2 Block Design

2.1 Applications

Experimental blocking was introduced in 1935 by Ronald Fisher. The term "blocking" comes from Chapter 22 of [6] concerning an agricultural experiment where an "area is divided into eight blocks...and each of these is divided into five plots". In Fisher's setting, blocks were predefined and not "designed". Subsequently, block design has been applied to a range of domain areas.

Greevy et al. described several medical applications and supporting methods in [8], in particular the efficacy of ACE-inhibitor drugs in pediatric cancer survivors. Rosenbaum considered cadmium and lead in the blood of smokers in the matching study described in [26]. The same author in [27] provides a comprehensive overview of matching algorithms inside and outside the context of block design, using the possible impact of antidepressent medication on bone density as a motivating example.

There are an array of social science applications of matched block design. Bruhn and McKenzie [4] survey several applications. First, Sri Lankan microenterprises from [5] are used to simulate a program boosting firm profits. Second, a sub-sample of the Mexican ENE survey in [14] focuses on heads of household receiving training or nutrition intervention to raise income. Third, the Indonesian Family Life Survey (IFLS) [30] examines children at risk of dropping out of school and households facing increases in expenditure per capita. Finally, the LEAPS data in Pakistan [1] considers youth education and nutrition. The microenterprise dataset is described in [31].

Keele et al. applied match block design to assess the effectiveness of a computer-aided instruction program for improving reading comprehension in

[17]. Moore described several political science applications in [24]. Panagopoulos and Green applied MB to a campaign finance analysis in [25].

2.2 Solution Approaches

Various algorithmic approaches for block design have been proposed.

Rosenbaum et al. [28] considered greedy and regression-based techniques for matched block design, with the aim of reducing covariate imbalance. Greevy et al provided a more formal treatment of such techniques, using Mahalanobis distance and bipartite matching for the $t = 2$ case [8]. In 2009, Bruhn and McKenzie [4] established that matched pair design yielded more balanced covariates than randomized methods, as measured on four panel data sets. This work roughly coincides with the introduction of the `blocktools` package in 2008, see [23].

Imai, King et al. advocate for the use of PB with block randomized experiments in [11], noting: "Since matching prior to random treatment assignment can greatly improve the efficiency of causal effect estimation ([3]; [8]), and matching in pairs can be substantially more efficient than matching in larger blocks, matched-pair, cluster-randomization (PB) would appear to be an attractive design for field experiments [13]." Matched pair design methodologies were employed and enhanced by in a universal health insurance experiment in Mexico, see [11]. In [12] the authors extended their analysis, considering the impact of the choice of distance metric in matched block design.

Mixed integer programming (MIP) was first applied to matched block design by Zubizarreta in [32], who considered "the total sum of distances and a weighted sum of specific measures of covariate imbalance." The method described in [32] differs from our scenario because it considers a) PB only, b) modeling covariate imbalance directly, c) linearized objectives only. The author's `mipmatch` package is the only previous matched block design implementation to use MIP of which we are aware. Zubizarreta and Keele further considered both block and treatment balance in multilevel matching, again with $t = 2$ [33].

In 2016, Higgins et al [10] proposed a heuristic approach to the full $t \geq 2$ m atched block design problem, with a guaranteed bound of no worse than 4x from optimal match quality. Further, this approach scales to large problem sizes. This approach could be a useful alternative to those presented here, but we are not aware of a library implementing the described approach.

There is also literature on closely related problems. Balanced risk set matching is the problem of matching treated subjects with one or more control subjects. This may occur across multiple groups, but the key difference from block design is that pairs across groups are matched, rather than forming strata (blocks). Li et al provided the `brsmatch` package as outlined in [20]. Lu and Rosenbaum in [22] gave a globally optimal risk set matching algorithm for three experimental groups, which does not carry over to MB since it involves pairwise matching.

Lu, Greevy et al consider the related problem of optimally assigning blocked units to treatments in [21]. Their results can be applied once blocks have been determined, i.e. as a postprocessing step to block design, as we discuss in Section 5.

3 Pair Blocking

Assume a set of units $u_1, \ldots, u_n$. Let $f \in \mathbb{R}^{n \times v}$ represent unit covariates with $f_{i\alpha}$ the value of covariate α for unit u_i. Let $c \in \mathbb{R}^{n \times n}$ with c_{ij} representing the distance between units i and j, with distances not necessarily satisfying the triangle inequality. Since a unit cannot be matched with itself, $c_{ii} = \infty$. It is common to use the Mahalanobis distance, but we do not depend on this assumption in our analysis.

Assuming $t = 2$, let $x_{ij} \in \{0, 1\}$ represent whether items i and j are matched, that is $x_{ij} = 1$ iff item i and item j form a matched pair. Pair Blocking is then the problem of minimizing total distance between matched pairs, i.e.

$$\min \quad \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{ij} x_{ij}.$$

3.1 Greedy Pair Blocking

Greedy algorithms are an intuitive approach for solving PB. A greedy approach selects available (i, j) with minimal distance, matches them, and repeats.

Algorithm 1 PB-G(c)

let $s = \emptyset, h = \text{HEAP}(\{((i,j), c_{ij}) \mid i = 1, \ldots, n;\ i < j\})$ {heap ordered by c_{ij} values}
while not EMPTY(h) **do**
  $(i, j) := \text{POP}(h)$ {extract pair with minimum distance}
  $s := s \cup \{(i, j)\}$
  REMOVE$(h, \{((k, l), c_{kl}) \mid k \in \{i, j\}$ or $l \in \{i, j\}\})$ {no longer consider i, j}
end while
return s

PB-G corresponds to `algorithm=optGreedy`[1] in `blocktools`. Observe that PB-G is not guaranteed to yield optimal solutions. Consider:

$$c = \begin{bmatrix} \infty & 1 & 2 & 2 \\ 1 & \infty & 2 & 2 \\ 2 & 2 & \infty & 11 \\ 2 & 2 & 11 & \infty \end{bmatrix}$$

PB-G yields solution $\{(1, 2), (3, 4)\}$ with total cost = 12, but $\{(1, 3), (2, 4)\}$ has cost 8.

[1] Allowing for differences in how ties are broken.

3.2 Optimal Pair Blocking

Defining x as before, we formulate PB as:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1, \ldots, n, \\ & \sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1, \ldots, n, \\ & x_{ij} \in \{0, 1\}. \end{aligned} \tag{PB-O}$$

The formulation (PB-O) is equivalent to a linear assignment problem (LAP) with cost matrix c, equivalent to the well-known minimum cost bipartite matching problem, see [21]. Though x is binary, the integrality condition can be relaxed due to Birkhoff [2]. LAP is easily solved for problem sizes considered in this work; the Hungarian method of Kuhn [19] and Jonker-Volgenant approach [16] each have $O(n^3)$ complexity.

In Section 6 we compare the match quality of PB-G and PB-O. Despite the guaranteed globally optimal solution quality of (PB-O), there may be legitimate reasons for using a heuristic approach. First, while PB-G and PB-O are both polynomial time algorithms, PB-G is faster in practice. Second, as noted by [11], if it is anticipated that items will be added or removed prior to experimentation, an online algorithm may be more appropriate. In light of such considerations, `blocktools` offers the choice between PB-O and PB-G as algorithm options `optGreedy` and `optimal`.

In certain cases, minimizing the sum of distances between matched pairs may not be the desired objective, see [32]. The formulation (PB-O) can be adapted for such considerations, but we will not present them here for brevity.

4 Matched Blocking

We now consider extending PB solution methods to the general $t \geq 2$ case. A simple approach for MB, which we refer to as MB-N, is to scan items in order of index, add the unmatched item of least distance to the current block, and repeat. MB-N corresponds to corresponds to `algorithm=naiveGreedy` in `blocktools`.

PB-G is trivially extended to MB by making $t-1$ greedy choices per iteration, which we refer to as MB-G. We omit the details for brevity. Since MB-G also solves PB, MB-G is not guaranteed to produce optimal solutions as previously demonstrated.

4.1 Clique Formulation

Let $G = (V, E)$ be a graph with vertices representing items and edges item distances. MB is equivalent to partitioning G into $m = \frac{n}{t}$ subcliques (blocks) of

size t, referred to in [15] as t-way equipartitioning. We seek the equipartitioning that minimizes total distance within blocks.

Formulation (PB-O) can be extended to model t-way equipartitioning. Letting binary decision variable x_{ij} where $x_{ij} = 1$ iff item i and item j are assigned to the same block, we have:

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{ij} x_{ij} & & \\
\text{s.t.} \quad & \sum_{j} x_{ij} = t - 1 & & i = 1, \ldots, n, \\
& \sum_{i} x_{ij} = t & & j = 1, \ldots, n, \\
& x_{ij} = x_{ji} & & i, j = 1, \ldots, n, \\
& x_{ij} + x_{jk} - x_{ik} \leq 1 & & \forall i < j < k, \\
& x_{ij} - x_{jk} + x_{ik} \leq 1 & & \forall i < j < k, \\
& -x_{ij} + x_{jk} + x_{ik} \leq 1 & & \forall i < j < k, \\
& x_{ij} \in \{0, 1\}. & &
\end{aligned}
\tag{MB-O}
$$

This formulation closely resembles that provided by in [9] in the context of a generic clustering framework. Note formulation (MB-O) has n^2 variables and $O(n^3)$ constraints. The large number of constraints in this model, necessary to ensure assigned items are in cliques, can make (MB-O) computationally infeasible for large n, as shown in Section 6. Alternatively (MB-O) can be reformulated as a bilinear model, but such a reformulation is less amenable to efficient solution by standard solvers.

4.2 Divide and Conquer

We now turn to approaches that offer explicit tradeoffs between block balance and computational time. Unlike (MB-O), these approaches do not guarantee optimal solutions to MB, however they do leverage the strengths of the formulation presented in Section 3.2.

Our divide-and-conquer metaheuristic separates the n items into subpartitions, solves each subpartition (a smaller MB), and returns the union of results. Suppose $s \leq n$ is the desired upper bound on subpartition size. Then clearly $q = s - (s \mod t)$ is the largest subpartition size with equally sized subcliques, and $p = \left\lceil \frac{n}{q} \right\rceil$ is the subpartition count. The size of each subpartition is $s_i = q$ for $i = 1 \ldots p-1$ and $s_p = q - (s \mod t)$.

Let MB-S be any solution method for MB. Let PARTITION be a function that separates cluster indexes into disjoint g_i, $i \in 1 \ldots p$. Finally, let $c[g_i, g_i]$ denote the submatrix with row and column indexes g_i. The resulting metaheuristic is given in Algorithm 2.

Algorithm 2 MB-DC(c, t, s)

1: $q = s - (s \mod t)$, $p = \left\lceil \frac{n}{q} \right\rceil$
2: $s_i = q$ for $i = 1 \dots p-1$, $s_p = q - (s \mod t)$
3: $g = \text{PARTITION}(c, s)$
4: **for** $i = 1 \dots p$ **do**
5: $\quad x_i = \text{MB-S}(c[g_i, g_i])$
6: **end for**
7: **return** $\bigcup_i^r x_i$

The resulting block balance yielded by MB-DC depends on the choice of partition, the choice of MB-S, and maximum partition size s. We now present two MB-DC variants with different quality-runtime tradeoffs. Observe that MB-DC reduces to MB-G when $s = t$ and to MB-O when $s = n$.

Greedy Partitioning A simple choice for PARTITION is to use MB-G and let $g_i = \{j \mid j \in \sum_{k=1}^{p-1} s_k \cdots \sum_{k=1}^{i} s_k\}$, that is, segment the results of the greedy algorithm along block boundaries. If these partitions are then solved using MB-S = MB-O, the solution of each partition is guaranteed to be no worse than the results yielded by MB-G. Therefore the divide-and-conquer solution is guaranteed to be no worse than MB-G. We refer to MB-DC with greedy partitioning as MB-DC-GP.

Assignment Partitioning MB-DC-GP improves on MB-G only to the extent that items within subpartitions g_i are rearranged. Intuitively, it therefore makes sense to assign blocks to partitions that are close to each other, rather than simply partition in the order provided by MB-G. The idea behind assignment partitioning is to assign blocks to partitions in such a way that minimizes total distance between the centroids of assigned blocks.

As previously noted there are $m = \frac{n}{t}$ blocks in a solution to MB. Consider the m centroids of the items within each block in an MB solution. Let $d \in \mathbb{R}^{m \times m}$ be the centroid-centroid distances. Let $z_{ij} = 1$ iff block i is assigned to partition j. We solve the partitioning assignment problem (PAP):

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{m} \sum_{j=1}^{m} d_{ij} z_{ij} \\
\text{s.t.} \quad & \sum_{j=1}^{m} z_{ij} = 1 \qquad i = 1, \dots, m, \\
& \sum_{i=1}^{m} z_{ij} = b \qquad j = 1, \dots, m, \\
& z_{ij} \geq 0.
\end{aligned}
\tag{PAP}
$$

The resulting LAP formulation of (PAP) is easily solved, as was demonstrated in Section 3.2. The desired partitions are given by $g_j = \{i \mid z_{ij} = 1\}$. We refer to this variant as MB-DC-PAP.

Extensions Finally, note that if MB-DC is recursive, that is, MB-S = MB-DC, the resulting algorithm is analogous to nested dissection approaches for sparse matrix factorization, see [7]. We do not explore this connection further here.

5 Treatment Balance

As noted in Section 2, certain block design techniques attempt to balance covariate values across treatment groups directly. In [32] the authors consider a weighted sum of covariate imbalance measures as an objective function. These measures include balancing means of treatment groups, multivariate moments, and Kolmogorov-Smirnov statistics.

Block distance is invariant with respect to item swaps within blocks. Therefore we can apply swap-based treatment balance techniques as a postprocessing step to any of the block balance approaches previously described. Let us consider treatment balance using a standard measure, standardized mean difference (SMD). First, let us define $y_{ilk} = 1$ if and only if unit i is assigned to block l and treatment k. Letting

$$F_{k\alpha}(y) = \sum_{i=1}^{n} \sum_{l=1}^{m} f_{i\alpha} y_{ilk},$$

the standardized mean difference between treatments k and l for covariate α is

$$SMD(k, l, \alpha; f, y) = \frac{F_{k\alpha}(y) - F_{l\alpha}(y)}{\sqrt{\sum_{i=1}^{n} f_{i\alpha}^2}}.$$

Taking the mean across all pairs, we define the following block design covariate balance metric:

$$b(f, y) = \sum_{\alpha=1}^{v} \frac{1}{\binom{t}{2}} \sum_{k=1}^{t} \sum_{l=k+1}^{t} SMD(k, l, \alpha; f, y).$$

We then consider the nonlinear assignment problem

$$\begin{aligned}
\min \quad & b(f, y) \\
\text{s.t.} \quad & \sum_{i=1}^{n} y_{ilk} = 1 \qquad l = 1, \ldots, m;\ k = 1, \ldots, t, \\
& \sum_{l=1}^{m} y_{ilk} = 1 \qquad i = 1, \ldots, n;\ k = 1, \ldots, t \\
& \sum_{k=1}^{t} y_{ilk} = 1 \qquad k = 1, \ldots, t;\ l = 1, \ldots, m \\
& y_{ilk} \in \{0, 1\}.
\end{aligned} \tag{COV-OPT}$$

A local improvement approach for (COV-OPT) is to sort blocks l by SMD and then repeatedly look for swaps within blocks that reduce $b(f, y)$. This approach is employed in the results that follow. While we do not consider optimal solutions for (COV-OPT) here, we regard this an area for future research.

6 Computational Results

6.1 Test Problems

We have assembled a test suite of matched block design problems, summarized in Table 1. In Table 1, n is the number of items and v is the number of covariates. In certain matched block design scenarios, "grouping" attributes are specified. In such cases the dataset is divided into g independent subproblems; see the `blocktools` documentation [23] for more details.

The sources of the test suite instances are as follows. The Sri Lankan microenterprises, Mexican BNE survey, IFLS, and LEAPS examples from [5] are included. The `x100` test problem is a simple synthetic problem included with `blocktools`. A publicly available dataset provided by Ian Silver is also included, see [29]. Finally, we also include an anonymized dataset from Airbnb called `states`. In this data set, there are three treatment groups and 51 units (the fifty US states and DC).

Name	n	v	g	Description
`x100`	100	2	3	test problem included with `blocktools`
`states`	51	12	1	anonymized Airbnb experiment data
`silver`	492	4	1	a publicly available dataset provided by Ian Silver
`ifls_expend`	30	7	1	a subset of the Indonesian Family Live Survey (IFLS) dataset
`ifls_school`	30	7	1	a subset of the IFLS dataset
`leaps_height`	30	7	1	child and household data from the LEAPS project
`leaps_test`	30	7	1	child and household data from the LEAPS project
`mexico_ene`	30	7	1	a subset of the Mexican employment survey
`sri_lanka`	30	7	1	a subset of the Sri Lankan microenterprise dataset

Table 1. Test Problems

6.2 Algorithm Performance

In Table 2 we compare the match quality of `blocktools`, MB-G, MB-DC-G, MB-DC-PAP, and MB-O. The `blocktools` results are produced by taking the minimum score using the `naiveGreedy`, `optGreedy`, and `optimal` algorithm options. respectively. The newly presented methods MB-G, MB-DC-G, MB-DC-PAP, and MB-O are implemented in Python. Instances of (MB-O) and (PAP) such as those arising in MB-DC-G, MB-DC-PAP, MB-O are solved using the Gurobi 14.1 solver.

We report the relative gap in total distance for our new solution methods, using the best results from `blocktools` as a baseline. We do not report MB-O

results for the `silver` instances because they are too large to be solved optimally within two hours. For MB-DC-G and MB-DC-PAP, we choose maximum subproblem size $s = 8t$. Note that for $t = 2$, `blocktools` provides optimal results in all cases.

		blocktools	MB-G		MB-DC-G		MB-DC-PAP		MB-O	
Name	t	Value	Gap	Value	Value	Gap	Value	Gap	Value	Gap
`ifls_expend`	2	31.5	32.8	4.0%	32.8	4.0%	32.8	4.0%	31.5	0.0%
`ifls_expend`	3	73.9	73.9	0.0%	73.9	0.0%	73.9	0.0%	72.2	-2.3%
`ifls_expend`	4	106.3	106.3	0.0%	101.1	-4.9%	106.3	0.0%	99.6	-6.3%
`ifls_school`	2	29.3	30.8	5.1%	30.8	5.0%	30.8	5.1%	29.3	0.0%
`ifls_school`	3	74.7	74.7	0.0%	74.7	0.0%	73.0	-2.2%	69.4	-7.0%
`ifls_school`	4	115.7	115.7	0.0%	111.5	-3.6%	112.5	-2.8%	104.4	-9.8%
`leaps_height`	2	31.3	33.0	5.5%	33.0	5.5%	33.0	5.5%	31.3	-0.0%
`leaps_height`	3	76.4	76.4	-0.0%	73.4	-4.0%	75.8	-0.9%	68.3	-10.7%
`leaps_height`	4	108.1	108.1	-0.0%	102.2	-5.4%	107.2	-0.9%	101.3	-6.3%
`leaps_test`	2	33.5	36.2	8.2%	36.2	8.2%	35.7	6.6%	33.5	0.0%
`leaps_test`	3	80.3	80.3	0.0%	79.4	-1.2%	76.7	-4.5%	72.3	-10.0%
`leaps_test`	4	116.8	116.8	0.0%	112.7	-3.4%	116.2	-0.5%	106.8	-8.6%
`mexico_ene`	2	27.3	27.8	2.0%	27.8	2.0%	27.6	1.4%	27.3	0.0%
`mexico_ene`	3	69.0	69.0	0.0%	67.1	-2.8%	69.0	0.0%	63.4	-8.1%
`mexico_ene`	4	97.4	97.4	0.0%	91.3	-6.3%	96.5	-0.8%	90.5	-7.1%
`silver`	2	27.7	29.7	7.2%	29.7	7.2%	29.7	7.2%	27.7	0.0%
`silver`	3	73.1	72.8	-0.4%	72.1	-1.4%	72.1	-1.4%	*	*
`silver`	4	132.2	139.5	5.5%	139.5	5.5%	137.6	4.1%	*	*
`sri_lanka`	2	30.6	31.0	1.2%	31.0	1.2%	31.0	1.2%	30.6	0.0%
`sri_lanka`	3	74.1	74.1	0.0%	73.6	-0.7%	74.0	-0.1%	68.6	-7.4%
`sri_lanka`	4	114.6	114.6	0.0%	110.7	-3.4%	112.5	-1.9%	99.0	-13.6%
`states`	2	93.9	99.2	5.6%	99.2	5.6%	99.2	5.6%	93.9	0.0%
`states`	3	241.4	241.4	0.0%	241.4	0.0%	240.4	-0.4%	217.1	-10.0%
`states`	4	348.4	348.4	0.0%	348.4	0.0%	345.5	-0.8%	329.4	-5.5%
`x100`	2	18.9	21.3	12.7%	21.3	12.7%	20.9	10.6%	18.9	0.0%
`x100`	3	55.9	55.9	0.0%	55.9	0.0%	53.2	-4.8%	49.6	-11.3%
`x100`	4	116.7	116.7	0.0%	116.2	-0.4%	111.1	-4.8%	89.8	-23.1%

Table 2. Block algorithm performance relative to greedy approach

Table 2 shows that MB-O matches or exceeds `blocktools` block balance quality in all test instances, except the two `silver` cases where the method is too computationally expensive. For $t \geq 3$ instances we frequently see improvement of 10% or more. The computationally more efficient MB-DC-G and MB-DC-PAP approaches frequently outperform `blocktools`. We noted in Section 4.2 that the block balance of MB-DC-G and MB-DC-PAP is no worse than that provided by MB-G. However, the results for `blocktools` are sometimes better than MB-DC-G and MB-DC-PAP for two reasons. First, for $t = 2$, `blocktools`

produces optimal results when `algorithm=optimal`. Second, the `blocktools` implementation of MB-G randomly breaks minimum-distance ties, see [23].

In Table 3 we compare the runtime of these approaches for selected test instances. We see that for problems of modest size (everything other than `silver`), (MB-O) is solved in well under ten seconds, well within reason for offline use cases.

Name	t	MB-G	MB-DC-G	MB-DC-PAP	MB-O
`silver`	2	1.44	3.00	3.11	1.44
`silver`	3	1.61	4.27	5.7	*
`silver`	4	1.3	6.62	30.29	*
`states`	2	0.01	0.15	0.15	1.91
`states`	3	0.01	0.56	0.33	2.43
`states`	4	0.01	0.45	0.58	7.21
`x100`	2	0.04	0.32	0.31	1.2
`x100`	3	0.03	0.58	0.65	3.22
`x100`	4	0.03	2.85	2.39	4.04

Table 3. Runtime comparison by problem and algorithm

In Table 2 we noted that MB-O cannot solve the full `silver` instance within two hours. In Table 4 we solve increasingly large subsets of `silver`, observing that solution time increases rapidly with problem size and the number of treatments.

n	t=2	t=3	t=4	t=5	t=6
24	0.17	0.18	0.54	0.54	1.64
48	1.51	1.93	2.22	16.91	46.2
72	3.98	161.44	35.45	150.14	2551.25
96	10.55	1096.32	*	*	*

Table 4. MB-O runtime of `silver` by problem and number of treatments

Finally, we consider treatment balance. An example of covariate differences between algorithms is shown in Figure 2, where the mean SMD μ for each covarate is plotted for each algorithm. It is seen that while there are variations across covariates, overall the mean SMD μ for MB-O is 30% smaller than of MB-G, $\mu = 0.09$ and $\mu = 0.129$, respectively.

We summarize covariate balance across algorithms and instances in Table 5, reporting the ratio of the mean SMD of each solution to the best mean SMD each instance. The mean ratio is for MB-O is 0.47, the best of all reported methods. While covariate balance for MB-G, MB-DC-G, and MB-DC-PAP varies by problem, they are similar in the aggregate.

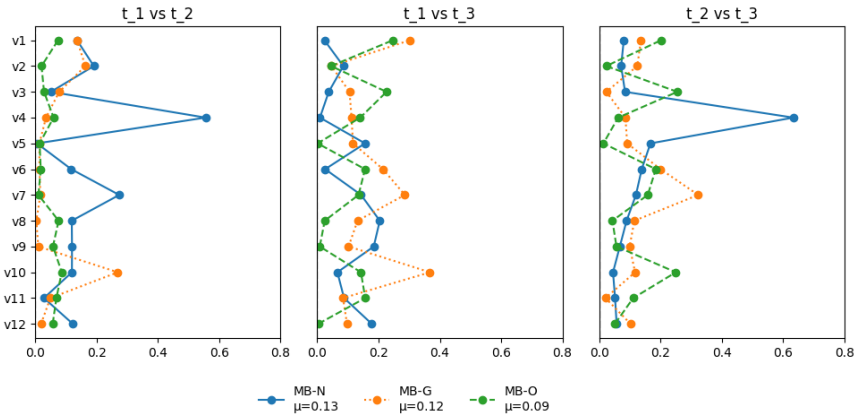

Fig. 2. Mean SMD for `states` by treatment and algorithm

Name	t	MB-N	MB-G	MB-DC-G	MB-DC-PAP	MB-O
`ifls_expend`	2	1.00	0.73	0.73	0.73	0.29
`ifls_expend`	3	1.00	0.77	0.77	0.77	0.55
`ifls_expend`	4	1.00	0.57	0.36	0.47	0.88
`ifls_school`	2	1.00	0.44	0.47	0.44	0.26
`ifls_school`	3	1.00	0.61	0.61	0.63	0.27
`ifls_school`	4	0.89	0.76	0.68	0.63	1.00
`leaps_height`	2	0.51	1.00	1.00	1.00	0.14
`leaps_height`	3	1.00	0.71	0.30	0.74	0.25
`leaps_height`	4	0.81	0.99	0.72	0.51	1.00
`leaps_test`	2	1.00	0.67	0.67	0.62	0.25
`leaps_test`	3	1.00	0.78	0.66	0.56	0.32
`leaps_test`	4	0.85	0.63	0.65	0.55	1.00
`mexico_ene`	2	1.00	0.49	0.49	0.57	0.28
`mexico_ene`	3	0.96	0.96	1.00	0.96	0.23
`mexico_ene`	4	1.00	0.66	0.65	0.70	0.52
`silver`	2	1.00	0.50	0.50	0.50	0.34
`silver`	3	1.00	0.65	0.48	0.68	0.58
`silver`	4	1.00	0.86	0.86	0.60	0.84
`states`	2	1.00	0.78	0.78	0.78	0.23
`states`	3	1.00	0.67	0.67	0.87	0.37
`states`	4	1.00	0.70	0.70	0.65	0.58
`x100`	2	1.00	0.18	0.18	0.10	0.01
`x100`	3	0.78	0.28	0.09	0.19	1.00
`x100`	4	1.00	0.41	0.37	0.26	0.19
μ		0.95	0.66	0.60	0.60	0.47

Table 5. Relative gap for covariate mean SMD by instance and algorithm

7 Conclusions

We have presented new approaches for matched block design that leverage mixed integer programming and divide-and-conquer formulations. We have shown that these approaches can provide improved match quality and covariate balance, and that they can be employed to account for large problem sizes. We have also presented a consolidated set of test instances that can be used to evaluate future approaches.

We believe there are opportunities to further refine the mixed integer formulation given in Section 4.1. We also believe that metaheuristic approaches are a promising direction for matched block design, especially in conjunction with the divide-and-conquer approach presented in Section 4.2. Finally, we believe that there are opportunities to address the covariate balance problem in a more principled manner than that presented in Section 5, as well as to consider block and treatment balance in the same modeling framework.

Acknowledgments

We thank Will Betz and Yan Zhang for their work on a problem that motivated this work. We thank Ian Silver for providing the dataset used in the `silver` examples, and for his encouragement. Finally, the author thanks the reviewers of this paper for their insightful comments and suggestions, which helped to improve the quality of this paper.

References

1. T. Andrabi, J. Das, A. I. Khwaja, and T. Zajonc, *Do value-added estimates add value? accounting for learning dynamics*, American Economic Journal: Applied Economics, 3 (2011), pp. 29–54.
2. G. Birkhoff, *Tres observaciones sobre el álgebra lineal [three observations on linear algebra]*, Univ. Nac. Tucumán. Revista A, 5 (1946), pp. 147–151.
3. H. Bloom, *The core analytics of randomized experiments for social research*, tech. report, MDRC, New York, 2006.
4. M. Bruhn and D. McKenzie, *In pursuit of balance: Randomization in practice in development field experiments*, American Economic Journal: Applied Economics, 1 (2009), pp. 200–232.
5. S. De Mel, D. McKenzie, and C. Woodruff, *Returns to capital in microenterprises: Evidence from a field experiment*, The Quarterly Journal of Economics, 123 (2008), pp. 1329–1372.
6. R. Fisher, *Design of Experiments*, Oliver and Boyd, Edinburgh, 1935.
7. A. George, *Nested dissection of a regular finite element mesh*, SIAM Journal on Numerical Analysis, 10 (1973).
8. R. Greevy, J. Silber, and P. Rosenbaum, *Optimal multivariate matching before randomization*, Biostatistics, 5 (2004), pp. 263–275.
9. M. Grötschel and Y. Wakabayashi, *A cutting plane algorithm for a clustering problem*, Mathematical Programming, 45 (1989), pp. 59–96.

10. M. J. Higgins, P. Hall, and S. Villar, *Improving massive experiments with threshold blocking*, Proceedings of the National Academy of Sciences of the United States of America, 113 (2016), pp. 7369–76.
11. K. Imai, G. King, and C. Nall, *The essential role of pair matching in cluster-randomized experiments, with application to the mexican universal health insurance evaluation*, Statistical Science, 24 (2009), pp. 29–53.
12. K. Imai, G. King, and C. Nall, *Rejoinder: Matched pairs and the future of cluster-randomized experiments*, Statist. Sci, 24 (2009), pp. 65–72.
13. K. Imai, G. King, and E. A. Stuart, *Misunderstandings among experimentalists and observationalists about causal inference*, Journal of the Royal Statistical Society: Series A (Statistics in Society), 171 (2008), pp. 481–502.
14. G. e. i. Instituto Nacional de Estadística, *Encuesta nacional de empleo 2004*. https://en.www.inegi.org.mx/programas/ene/2004/, 2004.
15. X. Ji and J. E. Mitchell, *Finding optimal realignments in sports leagues using a branch-and-cut-and-price approach*, International Journal of Operational Research, 1 (2016).
16. R. Jonker and A. Volgenant, *A shortest augmenting path algorithm for dense and sparse linear assignment problems*, Computing, 38 (1987), pp. 325–340.
17. L. Keele, M. Lenard, and L. Page, *Matching methods for clustered observational studies in education*, Journal of Research on Educational Effectiveness, 14 (2021), pp. 696–725.
18. S. Kesavan, S. J. Lambert, J. C. Williams, and P. K. Pendem, *Doing well by doing good: Improving retail store performance with responsible scheduling practices at the gap, inc.*, Manage. Sci., 68 (2022), p. 7818–7836.
19. H. W. Kuhn, *The Hungarian Method for the Assignment Problem*, Naval Research Logistics Quarterly, 2 (1955), pp. 83–97.
20. Y. P. Li, K. J. Propert, and P. R. Rosenbaum, *Balanced risk set matching*, Journal of the American Statistical Association, 96 (2001), pp. 870–882.
21. B. Lu, R. Greevy, X. Xu, C. Beck, J. Morris, R. Strodel, L. Miao, B. George, A. Linton, T. Speroff, X. Liu, H. Nian, C. Lindsell, and P. Rosenbaum, *Optimal nonbipartite matching and its statistical applications*, American Statistician, 65 (2011), pp. 21–30.
22. B. Lu and P. R. Rosenbaum, *Optimal pair matching with two control groups*, Journal of Computational and Graphical Statistics, 13 (2004), pp. 422–34.
23. R. T. Moore, *blocktools: Blocking, assignment, and diagnosing interference in randomized experiments*, 2013, https://www.ryantmoore.org/html/software.blockTools.html. Accessed 12 February 2024.
24. R. T. Moore, *Multivariate continuous blocking to improve political science experiments*, Political Analysis, 20 (2017), p. 460–479.
25. C. Panagopoulos and D. P. Green, *Field experiments testing the impact of radio advertisements on electoral competition*, American Journal of Political Science, 52 (2008), pp. 156–168.
26. P. R. Rosenbaum, *Impact of multiple matched controls on design sensitivity in observational studies*, Biometrics, 69 (2013), pp. 118–127.
27. P. R. Rosenbaum, *Modern algorithms for matching in observational studies*, Annual Review of Statistics and Its Application, 7 (2020), pp. 143–176.
28. P. R. Rosenbaum and D. B. Rubin, *The central role of the propensity score in observational studies for causal effects*, Biometrika, 70 (1983), pp. 41–55.

29. I. A. Silver, *Blocked randomization with r for randomized controlled trials*. https://ianasilver.com/blocked-randomization-with-r-for-randomized-controlled-trials/, 2025. Accessed: 2025-02-09.
30. J. Strauss, K. Beegle, B. Sikoki, A. Dwiyanto, Y. Herawati, and F. Witoelar, *The third wave of the indonesia family life survey (ifls3): Overview and field report*, (2004). WR-144/1-NIA/NICHD.
31. C. Woodruff, D. McKenzie, and S. de Mel, *Returns To Capital In Microenterprises : Evidence From A Field Experiment*, The World Bank, 2007.
32. J. R. Zubizarreta, *Using mixed integer programming for matching in an observational study of kidney failure after surgery*, Journal of the American Statistical Association, 107 (2012), pp. 1360–1371.
33. J. R. Zubizarreta and L. K. and, *Optimal multilevel matching in clustered observational studies: A case study of the effectiveness of private schools under a large-scale voucher system*, Journal of the American Statistical Association, 112 (2017), pp. 547–560, https://arxiv.org/abs/https://doi.org/10.1080/01621459.2016.1240683.