Improving Massive Experiments with Threshold Blocking

Jasjeet S. Sekhon

Drawing Causal Inference from Big Data
Sackler Colloquia, March 26, 2015
Massive Experiments

- Rising interest in fine-grained inference: e.g., subgroups, heterogeneous effects
- Some traditional experimental design methods have become computationally infeasible
- Researcher’s degrees of freedom has increased
- Big rise in false positive rate
Why We Randomize?

- Unbiased estimator by design
- Make probability statements; “reasoned basis for inference” (Fisher, Peirce)
- Separate design from analysis (Cochran, Rubin)
A new blocking method with nice theoretical properties

- Blocking: create strata and then randomize within strata

- Some analytical benefits for blocking, but the main one is transparency and minimizing fishing
A New Blocking Method

The method minimizes the pair-wise Maximum Within-Block Distance: $\lambda$

- Any valid distance metric (must satisfy the triangle inequality)
- Ensures good covariate balance by design
- Works for any number of treatments and any minimum number of observations per block
- It is fast: $O(n \log n)$ expected time
- It is memory efficient: $O(n)$ storage
- Approximately optimal: $\leq 4 \times \lambda$
- Special cases
  1. with one covariate: $\lambda$
  2. with two covariates: $\leq 2 \times \lambda$
PROBLEM: In finite samples, there is a probability of bad covariate balance between treatment groups

Bad imbalance on important covariates:
- Imprecise estimates of treatment effects
- Conditional bias

In large samples problems remain: we want to estimate treatment effects for subgroups
Some theoretical results about blocking

- Blocking cannot hurt the precision of the estimator:
  - if no worse than random matching
  - if sample from an infinite super population

- Blocking may increase the estimated variance. But this is specific to the estimator used (degrees of freedom). e.g., randomization inference solves the problem.
Adjustment and covariate imbalance

- **Post-stratification** [Miratrix, Sekhon, and Yu, 2013]:
  - Group similar units together after *after* randomization
  - SATE/PATE results good; *ex post* problems arise
  - Data mining concerns
- **Re-randomization** [Morgan and Rubin, 2012]:
  - Repeat randomly assigning treatments until covariate balance is “acceptable”

**LESSON**: design the randomization to build in adjustment
Some Current blocking approaches

- Matched-pairs blocking: Pair “most-similar” units together. For each pair, randomly assign one unit to treatment, one to control
- Optimal-greedy blocking [e.g. Moore, 2012]
- Some methods make principled probability statements impossible
Matched-Pairs

- No efficient way to extend approach to more than two treatment categories
- Fixed block sizes (2 units): design may pair units from different clusters
- Cannot estimate conditional variances [Imbens, 2011]
- Difficulty with treatment effect heterogeneity
Blocking by minimizing the Maximum Within-Block Distance (MWBD)

- Experiment with $n$ units and $t$ treatment categories
- Select a threshold $k \geq t$ for a minimum number of units to be contained in a block
- Block units so that each block contains at least $k$ units, and so that the maximum distance between any two units within a block—the MWBD—is minimized
- Threshold $k$: Allows designs with multiple treatment categories, multiple replications of treatments within a block
Threshold blocking: relaxing the block structure

Threshold blocking

Fixed-sized blocking
Theorem

*For all samples, all objective functions and all desired block sizes, the optimal threshold blocking is always weakly better than the optimal fixed-sized blocking.*

- Proof: interpret blocking as an non-linear integer programming problem.
  - The search set of threshold blocking is a superset of fixed-sized blocking.
But there are problems

- Problem 1: the theorem is for the objective function used to construct the blocks.
  - Might not be the quantity of true interest.

- Problem 2: No help to us if we cannot find the optimum. NP-hard problems

<table>
<thead>
<tr>
<th># units</th>
<th>Fixed-sized</th>
<th>Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>105</td>
<td>715</td>
</tr>
<tr>
<td>10</td>
<td>945</td>
<td>17,722</td>
</tr>
<tr>
<td>12</td>
<td>10,395</td>
<td>580,317</td>
</tr>
<tr>
<td>14</td>
<td>135,135</td>
<td>24,011,157</td>
</tr>
<tr>
<td>16</td>
<td>2,027,025</td>
<td>1,216,070,380</td>
</tr>
<tr>
<td>18</td>
<td>34,459,425</td>
<td>73,600,798,037</td>
</tr>
<tr>
<td>20</td>
<td>654,729,075</td>
<td>$5.2 \times 10^{12}$</td>
</tr>
</tbody>
</table>
The AppOpt algorithm

**Input:**
- Units’ covariates
- Distance metric
- Minimum block size: \( k = 2 \)

**Procedure:**
1. A undirected complete graph with distances as edge weights
2. Find \((k - 1)\)-nearest neighbor graph
3. Construct the second power of NNG
4. Find a maximal independent set (seeds)
5. Form blocks with the seeds and their neighbors in NNG
6. Assign remaining units to a block containing any neighbor
The AppOpt algorithm

Input:
- Units’ covariates
- Distance metric
- Minimum block size: $k = 2$

Procedure:
1. A undirected complete graph with distances as edge weights
2. Find $(k - 1)$-nearest neighbor graph
3. Construct the second power of NNG
4. Find a maximal independent set (seeds)
5. Form blocks with the seeds and their neighbors in NNG
6. Assign remaining units to a block containing any neighbor
The AppOpt algorithm

**Input:**
- Units’ covariates
- Distance metric
- Minimum block size: \( k = 2 \)

**Procedure:**
1. A undirected complete graph with distances as edge weights
2. Find \((k - 1)\)-nearest neighbor graph
3. Construct the second power of NNG
4. Find a maximal independent set (seeds)
5. Form blocks with the seeds and their neighbors in NNG
6. Assign remaining units to a block containing any neighbor
The AppOpt algorithm

Input:
- Units’ covariates
- Distance metric
- Minimum block size: \( k = 2 \)

Procedure:
1. A undirected complete graph with distances as edge weights
2. Find \((k - 1)\)-nearest neighbor graph
3. Construct the second power of NNG
4. Find a maximal independent set (seeds)
5. Form blocks with the seeds and their neighbors in NNG
6. Assign remaining units to a block containing any neighbor
The AppOpt algorithm

**Input:**
- Units’ covariates
- Distance metric
- Minimum block size: $k = 2$

**Procedure:**
1. A undirected complete graph with distances as edge weights
2. Find $(k - 1)$-nearest neighbor graph
3. Construct the second power of NNG
4. Find a maximal independent set (seeds)
5. Form blocks with the seeds and their neighbors in NNG
6. Assign remaining units to a block containing any neighbor
The AppOpt algorithm

**Input:**
- Units’ covariates
- Distance metric
- Minimum block size: $k = 2$

**Procedure:**
1. A undirected complete graph with distances as edge weights
2. Find $(k - 1)$-nearest neighbor graph
3. Construct the second power of NNG
4. Find a maximal independent set (seeds)
5. Form blocks with the seeds and their neighbors in NNG
6. Assign remaining units to a block containing any neighbor
The AppOpt algorithm

**Input:**
- Units’ covariates
- Distance metric
- Minimum block size: \( k = 2 \)

**Procedure:**
1. A undirected complete graph with distances as edge weights
2. Find \((k - 1)\)-nearest neighbor graph
3. Construct the second power of NNG
4. Find a maximal independent set (seeds)
5. Form blocks with the seeds and their neighbors in NNG
6. Assign remaining units to a block containing any neighbor
The AppOpt algorithm

**Input:**
- Units’ covariates
- Distance metric
- Minimum block size: $k = 2$

**Procedure:**
1. A undirected complete graph with distances as edge weights
2. Find $(k - 1)$-nearest neighbor graph
3. Construct the second power of NNG
4. Find a maximal independent set (seeds)
5. Form blocks with the seeds and their neighbors in NNG
6. Assign remaining units to a block containing any neighbor
The AppOpt algorithm

Input:
- Units’ covariates
- Distance metric
- Minimum block size: $k = 2$

Procedure:
1. A undirected complete graph with distances as edge weights
2. Find $(k - 1)$-nearest neighbor graph
3. Construct the second power of NNG
4. Find a maximal independent set (seeds)
5. Form blocks with the seeds and their neighbors in NNG
6. Assign remaining units to a block containing any neighbor
Preliminary simulation results: Complexity

<table>
<thead>
<tr>
<th>$n$</th>
<th>Non-bipartite</th>
<th>Opt. Greedy</th>
<th>AppOpt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU</td>
<td>Memory</td>
<td>CPU</td>
</tr>
<tr>
<td>$10^2$</td>
<td>0.0</td>
<td>34.1</td>
<td>0.1</td>
</tr>
<tr>
<td>$10^3$</td>
<td>1.6</td>
<td>84.0</td>
<td>4.3</td>
</tr>
<tr>
<td>$10^4$</td>
<td>352.3</td>
<td>4990.7</td>
<td>1050.5</td>
</tr>
<tr>
<td>$10^5$</td>
<td>?</td>
<td>&gt; 64000</td>
<td>?</td>
</tr>
<tr>
<td>$10^6$</td>
<td></td>
<td></td>
<td>3.4</td>
</tr>
<tr>
<td>$10^7$</td>
<td></td>
<td></td>
<td>44.3</td>
</tr>
<tr>
<td>$10^8$</td>
<td></td>
<td></td>
<td>679.5</td>
</tr>
</tbody>
</table>

- **CPU**: Average running time (seconds).
- **Memory**: Average maximum RAM-use for one run (MB).
Preliminary simulation results: Performance

- Setting: two-dimensional covariate space, uniform distribution.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^2$</td>
<td>2</td>
<td>0.5%</td>
<td>-14.9%</td>
<td>283.5%</td>
<td>6.8%</td>
</tr>
<tr>
<td>$10^2$</td>
<td>4</td>
<td></td>
<td></td>
<td>187.8%</td>
<td>11.0%</td>
</tr>
<tr>
<td>$10^3$</td>
<td>2</td>
<td>-4.2%</td>
<td>-16.4%</td>
<td>881.3%</td>
<td>5.2%</td>
</tr>
<tr>
<td>$10^3$</td>
<td>4</td>
<td>-7.1%</td>
<td>-17.0%</td>
<td>677.2%</td>
<td>11.0%</td>
</tr>
<tr>
<td>$10^4$</td>
<td>2</td>
<td></td>
<td></td>
<td>2565.8%</td>
<td>3.4%</td>
</tr>
<tr>
<td>$10^4$</td>
<td>4</td>
<td></td>
<td></td>
<td>2161.4%</td>
<td>9.9%</td>
</tr>
</tbody>
</table>

- Max: Maximum within-block distance (relative to AppOpt).
- Avg.: Average within-block distance (relative to AppOpt).
Conclusion

- Fast algorithm:
  - NNG plus $O(d^0 kn)$ time and $O(d^0 kn)$ space
  - K-d trees NN: $O(2^d kn \log n)$ expected time, $O(2^d kn^2)$ worst time, and $O(kn)$ storage
  - Compare with bipartite, network flow methods:
    - e.g., Derigs: $O(n^3 \log n + dn^2)$ worst time and $O(d^0 n^2)$ space

- Closer to clustering than traditional blocking methods
- Important for separating design from analysis
- Lots of questions about best way to handle estimation
  - Design based estimators: Difference of means; Horvitz-Thompson estimator; double Hájek estimator
  - Probably do want to run a model on the blocked data. What if there is heterogeneity by blocks? $\frac{p}{n} \neq 0$
Joint Work with Michael J. Higgins and Fredrick Sävje
The Neyman-Rubin potential outcomes framework assumes the following model for response [Splewa-Neyman, Dabrowska, and Speed, 1990, Rubin, 1974]:

$$Y_{kc} = y_{kct_1} T_{kct_1} + y_{kct_2} T_{kct_2} + \ldots + y_{kct_r} T_{kct_r}.$$ 

- $Y_{kc}$: Observed response of $k$th unit in block $c$.
- $y_{kct}$: Potential outcome of the unit under treatment $t$.
- $T_{kct}$: Treatment indicators. $T_{kct} = 1$ if the unit receives treatment $t$, $T_{kct} = 0$ otherwise.
Parameters of interest and estimators

- Parameters of interest: Sample average treatment effect of treatment $s$ relative to treatment $t$ ($\text{SATE}_{st}$):

$$\text{SATE}_{st} = \sum_{c=1}^{b} \sum_{k=1}^{n_c} \frac{y_{kcs} - y_{kct}}{n}$$

- Two unbiased estimators of $\text{SATE}_{st}$ are the difference-in-means estimator and the Horvitz-Thompson estimator.

$$\hat{\delta}_{st,\text{diff}} \equiv \sum_{c=1}^{b} \frac{n_c}{n} \sum_{k=1}^{n_c} \left( \frac{y_{kcs}}{\# T_{cs}} - \frac{y_{kct}}{\# T_{ct}} \right)$$

$$\hat{\delta}_{st,\text{HT}} \equiv \sum_{c=1}^{b} \frac{n_c}{n} \sum_{k=1}^{n_c} \left( \frac{y_{kcs} T_{kcs}}{n_c/r} - \frac{y_{kct} T_{kct}}{n_c/r} \right)$$

- Assume complete randomization of treatment, $r$ divides $n_c$. 
Variance of estimators

\[
\text{Var}(\hat{\delta}_{st, \text{diff}}) = \text{Var}(\hat{\delta}_{st, \text{HT}}) \\
= \sum_{c=1}^{b} \frac{n_c^2}{n^2} \left( \frac{r - 1}{n_c - 1} (\sigma_{cs}^2 + \sigma_{ct}^2) + 2 \frac{\gamma_{cst}}{n_c - 1} \right)
\]

\[
\mu_{cs} = \frac{1}{n_c} \sum_{k=1}^{n_c} y_{kcs}
\]

\[
\sigma_{cs}^2 = \frac{1}{n_c} \sum_{k=1}^{n_c} (y_{kcs} - \mu_{cs})^2
\]

\[
\gamma_{cst} = \frac{1}{n_c} \sum_{k=1}^{n_c} (y_{kcs} - \mu_{cs})(y_{kct} - \mu_{ct})
\]
Variance of estimators

\[ \text{Var}(\hat{\delta}_{st,\text{diff}}) = \text{Var}(\hat{\delta}_{st,HT}) = \sum_{c=1}^{b} \frac{n_c^2}{n^2} \left( \frac{r-1}{n_c - 1} \left( \sigma^2_{cs} + \sigma^2_{ct} \right) + 2 \frac{\gamma_{cst}}{n_c - 1} \right) \]

- Note: \( \sigma^2_{cs} \) and \( \sigma^2_{ct} \) are estimable, \( \gamma_{cst} \) not directly estimable.
- Conservative estimate:

\[ \hat{\text{Var}} = \sum_{c=1}^{b} \frac{n_c^2}{n^2} \left( \frac{2(r-1)}{n_c - 1} \left( \hat{\sigma}^2_{cs} + \hat{\sigma}^2_{ct} \right) \right) \]

- Small differences for more general treatment assignments.
When does blocking help?

- Blocking vs. completely randomized treatment assignment (no blocking): which estimates of $SATE_{st}$ have lower variance?

- Blocking helps if and only if:

\[
\sum_{c=1}^{b} n^2_c \left[ \left( \frac{(r - 1)(\sigma^2_s + \sigma^2_t) + 2\gamma_{st}}{\sum n^2_c (n - 1)} \right) - \left( \frac{(r - 1)(\sigma^2_{cs} + \sigma^2_{ct}) + 2\gamma_{cst}}{n^2 (n_c - 1)} \right) \right] \geq 0
\]

- Intuitive to make $\sigma^2_{cs}, \sigma^2_{ct}$ small w.r.t. $\sigma^2_s, \sigma^2_t$, but other blocking designs may also improve treatment effect estimates.
Can blocking hurt?

- When blocking is completely randomized:

\[
E \left[ \sum_{c=1}^{b} n_c^2 \left( \frac{(r - 1)(\sigma_{cs}^2 + \sigma_{ct}^2) + 2\gamma_{cst}}{n^2(n_c - 1)} \right) \right]
= \sum_{c=1}^{b} n_c^2 \left( \frac{(r - 1)(\sigma_{s}^2 + \sigma_{t}^2) + 2\gamma_{st}}{\sum n_c^2(n - 1)} \right)
\]

Blocked variance = Completely randomized variance

- Any improvement to completely random blocking → Reduced variance in treatment effect estimates.


