EasyChair Preprint

№ 3110

Orthogonal Traffic Assignment in Online Overlapping A/B Tests

Tao Xiong, Yong Wang and Senlie Zheng

April 3, 2020

Orthogonal Traffic Assignment in Online Overlapping A/B Tests

Tao Xiong[1], Yong Wang[1], and Senlie Zheng[1]

Tencent Inc. China

Abstract. Online controlled experiments, aka A/B tests, are widely used in data-driven decision-making of many companies, including Google, Amazon, Facebook, Microsoft, Yahoo etc. With hundreds of experiments running at the same time in Wechat, we employ an overlapping infrastructure to assign traffic. Traffic assignment in an improper way will cause unreliable conclusions to experiments, which may causes very significant business degradation. Even a little fractions of one percent key metric degradation will brings severe consequences. In this paper we propose a novel approach for traffic assignment in online overlapping A/B tests, this approach aims to reduce interactions between experiments with traffic overlap by a orthogonal traffic assignment design. We illustrated how this technique can reduce interactions between experiments, and evaluated the effects of using this technique compared to regular method. This technique, along with some other mechanisms, formed a reliable traffic assignment infrastructure for trustworthy overlapping A/B tests.

Keywords: A/B testing · traffic assignment · Galois field

1 Introduction

Online controlled experiments (OCEs), also known as A/B tests play an vital role in many data-driven companies, such as Amazon [1], eBay, Etsy [2], Facebook [3], Google [4], Groupon, Intuit [5], LinkedIn [6], Microsoft [7], Netflix [8], Shop Direct [9], Yahoo [10], and Zynga [11]. The underlying statistical analysis of A/B tests is based on correct traffic assignment, which makes traffic assignment crucial to reliable conclusion.

As a simple but robust solution, we can randomly split all randomization units (e.g. users) to different A/B tests without overlap, no overlap means that users in one A/B test will never take part in other A/B tests, makes all parallel A/B tests run without interactions. But with the growth of number of parallel experiments, we will run out of users rapidly, for example, if every A/B test contains 2% of users, we can run 50 A/B tests at the same time with this simple traffic assignment design. But in real world, we have hundreds of A/B tests need to be run at the same time in Wechat.

Traffic reuse is necessary in real world A/B tests, but traffic reuse also brings interactions among overlapping experiments. Google[4] propose an overlapping

A/B tests infrastructure for traffic assignment in a elegant way, and use hash function with seeds to reduce interactions among experiments of different layers. However, it will be illustrated that hash functions such as murmurhash cannot ensure orthogonality property perfectly, which will cause bad traffic assignment results and then unreliable experiment conclusions. This is one of top challenges is practical OCE platform[12].

In this paper we propose a orthogonal traffic assignment design in online overlapping A/B tests, we use a pre-generated table named orthogonal table for traffic assignment, the orthogonal table has good properties for trustworthy traffic assignment, and avoid the bad cases of hash functions. We propose an algorithm for generating orthogonal tables, and provide rigorous proof for this algorithm. This orthogonal design is evaluated by simulations and real world cases, showing that our method is better than traditional methods. The method we describe in this paper was implemented in the Wechat A/B tests platform in 2019, hundreds of A/B tests are running parallelly in the platform every day.

In Section 2 we describe the preliminaries of A/B tests, In Section 3 we describes the principle of orthogonal traffic assignment design and related details. In Section 4 we present simulation illustration of our design. In Section 5 we show some real world cases.

2 Preliminaries

In this section, we describe the typical approach for assigning traffic in overlapping A/B test systems briefly.

2.1 A/B Tests

A/B tests(or experiments) involve comparing variations against a known baseline, web requests or users are assigned into every variations randomly, related data is collected and then statistical inferences are conducted to decide which one is better.

More specifically, for example, we have two variations, A as control and B as treatment, and we observed n_A and n_B i.i.d. samples for A and B respectively, denoted as X_i and Y_i, according to Central Limit Theorem, $\overline{X} = \sum X_i/n_A$ and $\overline{Y} = \sum Y_i/n_B$ are normal distributed when n_A and n_B are large enough, and their mean are μ_A and μ_B. Under null hypothesis $H_0 : \mu_A = \mu_B$, with estimated variance σ_A^2 and σ_B^2 respectively, statistic $t = \frac{\overline{X}-\overline{Y}}{\sqrt{\frac{\sigma_A^2}{n_A}+\frac{\sigma_B^2}{n_B}}}$ is Student's t distributed with zero mean. For a given significant level α, we can construct $1-\alpha$ confidence interval of statistic t, if the observed t do not in the $1-\alpha$ confidence interval, we can conclude that H_0 is rejected.

2.2 Traffic

Traffic are the randomization units assigned to each variations (which is also called experiment groups). Typically, users are usually the randomization units in many real world applications.

2.3 Overlapping A/B Tests Infrastructure

In a simple experiment system, randomization units are assigned to one and only one experiment in order to avoid interactions among different experiments. It is simple and robust, but causes traffic shortage easily when there are hundreds of parallel experiments conducting.

Overlapping A/B tests infrastructure supports traffic reuse by introducing the concepts listed below [4].

- A layer is a reallocation of all traffic of the system. Traffic belongs to one layer are reallocated into buckets in this layer.
- Buckets are the basic units of traffic assignment within a layer.
- A domain is a segmentation of traffic.

Every experiment can only associate to one layer, buckets in the layer are assigned and only assigned to experiments of this layer. Layers and domains can be nested. With layers and domains, traffic can be reused among parallel experiments in different layers.

3 Orthogonal Traffic Assignment

Traffic reuse supports more parallel experiments, but also bring some potential problems. The traffic assigned to one experiment will be reused in experiments of other layers, causing untrustworthy inferences when these traffic are not reallocated to buckets in other layers with equal possibilities. (Which is defined as orthogonality property of traffic assignment)

Generally hash functions with seeds are used in traffic reallocating in different layers, thus choosing good hash functions is the vital part of implementing a trustworthy traffic assignment system. Bad hash functions may causing severe problems, which will be illustrated below.

Murmurhash [13] is a widely used non-cryptographic hash function, and Murmurhash2 is used in our early traffic assignment system, but we found that it is not good enough in orthogonality property in specific seeds combinations. We tested some seeds pairs, for every seeds pair, we assign traffic (2^{20} users with userid from 0 to $2^{20}-1$) into 16 buckets by $Murmurhash2(userid, layer\_seed)\ modulo\ 16$ in every layer. Some seeds pairs shows poor orthogonal results, which is showed in figures below.

Figure 1 shows how traffic is distributed in the first layer whose layer seed is 368, users assigned to different buckets are colored with 16 distinct colors. We can see that users are almost evenly distributed in the first layer. In the second

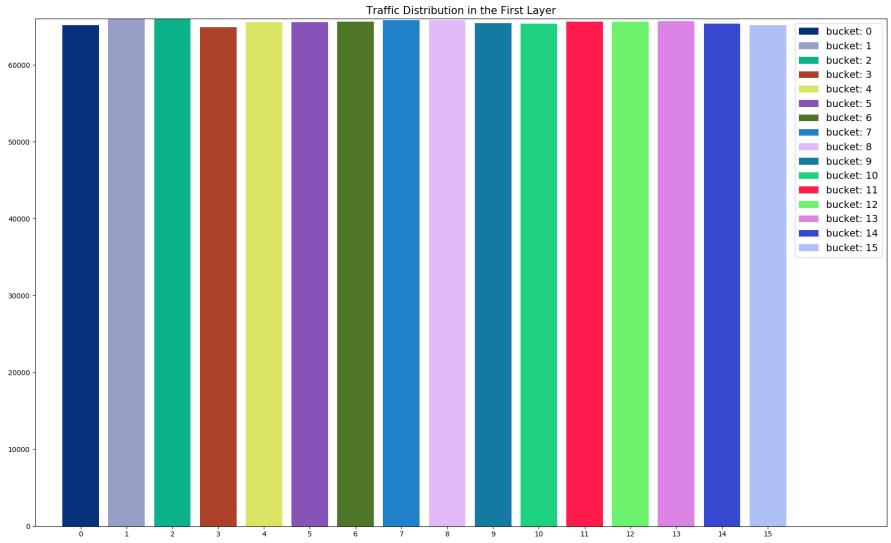

Fig. 1. Traffic Distribution in the First Layer whose seed is 368

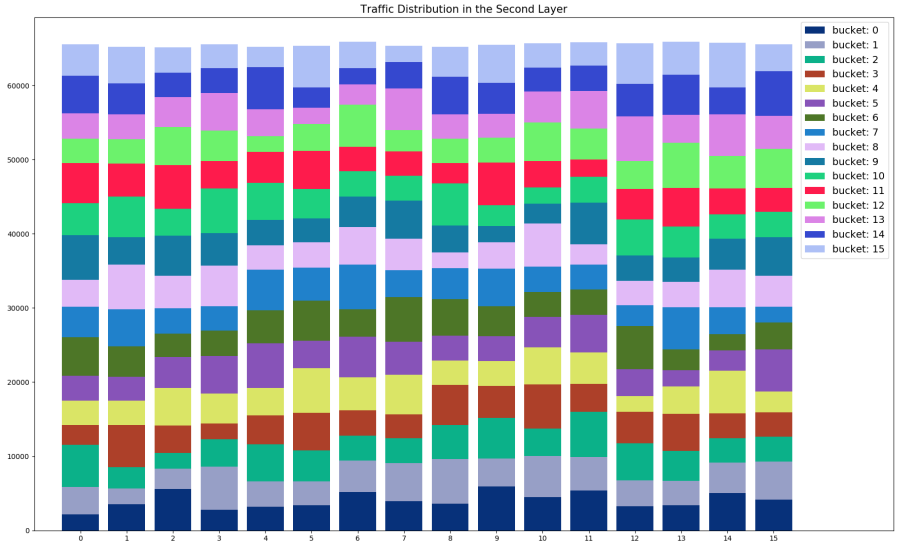

Fig. 2. Reallocated Traffic Distribution in the Second Layer whose seed is 60

layer whose seed is 60, these users are reallocated, but the reallocation is not good enough, as showed in Figure 2.

More formally, we made a chi-square test for every bucket in the first layer, users from every bucket in the first layer are reallocated to 16 buckets in the second layer, we expect these users to be assigned to 16 buckets with equal possibilities. In fact, all the chi-square tests failed with all pvalues nearly equal to zero, we can conclude that this pair of layer seeds are not good enough for generating trustworthy inferences, because the treatment effects to the very bucket of the first layer may cause bias to experiments in the second layer. The details are showed in Table 1.

Such bad seeds pair is not rare in practice, we evaluated all seeds from range [0, 100), it turns out that there are roughly 5% of seeds pairs are not good enough.

Table 1. Chi-square tests for every bucket in first layer

Reallocation distribution	Pvalue
2189,3535,5576,2795,3212,3403,5187,3947,3581,5946,4468,5405,3280,3373,5060,4179	0.0
3655,2144,2760,5798,3393,3209,4228,5120,6067,3730,5551,4460,3443,3311,4061,5112	0.0
5727,2863,2139,3699,4994,4171,3343,3336,4563,5463,3703,6154,5041,4041,3335,3350	0.0
2672,5646,3645,2099,3894,5055,3442,3220,5396,4322,5971,3742,4202,4979,3335,3280	0.0
3274,3315,5092,4045,3685,6056,4462,5387,3292,3410,4981,4253,2115,3680,5749,2781	0.0
3340,3197,4141,5097,6037,3683,5438,4459,3344,3294,4140,5071,3640,2192,2734,5728	0.0
5210,4092,3210,3387,4441,5378,3727,5963,4949,4064,3356,3403,5830,2805,2177,3606	0.0
4077,5023,3362,3315,5523,4470,6036,3693,4195,5054,3403,3340,2795,5722,3643,2141	0.0
3636,6019,4424,5474,3271,3442,5068,4232,2077,3565,5788,2784,3313,3447,5100,4152	0.0
6027,3716,5391,4370,3445,3234,4105,5091,3627,2197,2678,5564,3417,3252,4133,5193	0.0
4332,5444,3616,6052,4991,3930,3395,3388,5728,2831,2204,3548,4890,4206,3325,3477	0.0
5418,4469,5917,3687,4137,5182,3318,3288,2741,5707,3590,2265,4109,5173,3478,3153	0.0
3262,3285,5106,4103,2122,3626,5647,2872,3249,3395,5173,4195,3742,6099,4379,5336	0.0
3412,3377,4038,5069,3632,2176,2777,5632,3328,3177,4172,5103,6001,3746,5628,4427	0.0
5067,4165,3334,3330,5685,2709,2141,3571,5072,4202,3256,3374,4419,5423,3631,6006	0.0
4270,4929,3393,3224,2755,5653,3610,2172,4024,5161,3287,3151,5445,4462,6013,3633	0.0

Choosing another hash function with strong cryptographic guarantees (such as SHA) may solve the bad cases, but it always brings huge performance overhead, which is unacceptable in real-time applications. In this section we propose a novel method of assigning traffic in a orthogonal way.

3.1 Orthogonal Assignment Table

Orthogonal assignment table is a table of N rows and N^2 columns, which satisfy two properties listed below

Property 1. Each row is a permutation of $\{0, 1, 2, ..., N^2 - 1\}$, and is divided into N areas sequentially, every area consists of N elements.

Property 2. Any two areas that not in the same row share one and only one element in common.

Table 2 is a orthogonal assignment table of size $N = 4$.

We can use orthogonal assignment table to ensure orthogonal property of traffic allocation, first, divert all the traffic to N^2 segments randomly, which can be done by regular methods such as hash functions, label all the segments from 0 to $N^2 - 1$, then we associate distinct rows of orthogonal assignment table to different layer of our traffic assignment system, for traffic allocation in each layer, the segments labeled with numbers in one area are assigned to the bucket corresponding to this area. Every layer is associated with one row with N areas, thus all traffic can be assigned to N buckets in every layer.

Property 1 ensures that no segment will be assigned to more than one bucket in one layer, and no segment is missing in this layer. Property 2 ensures that for any two distinct layers, every bucket in one layer will be distributed in the other layer evenly.

Table 2. Orthogonal Assignment Table of size 4

Area 0	Area 1	Area 2	Area 3
00, 04, 08, 12	01, 05, 09, 13	02, 06, 10, 14	03, 07, 11, 15
00, 05, 10, 15	01, 04, 11, 14	02, 07, 08, 13	03, 06, 09, 12
00, 06, 11, 13	01, 07, 10, 12	02, 04, 09, 15	03, 05, 08, 14
00, 07, 09, 14	01, 06, 08, 15	02, 05, 11, 12	03, 04, 10, 13

3.2 Construct Orthogonal Assignment Table

We need to assign traffic to N buckets in one layer in which case N may be large in practice, the orthogonal assignment table becomes complicated with the growing of size N. We propose a algorithm for constructing orthogonal assignment table of any size N that satisfy $N = p^k$ with p is a prime number and k is a integer bigger that 0. See details in Algorithm 1. Correctness of Algorithm 1 is proved in the next section.

3.3 Proof of Algorithm 1

We now prove that the result of Algorithm 1 satisfy Property 1 and Property 2 of orthogonal assignment table.

For N elements of jth row and kth area, we can represent them as $\{e_i = i \times N + (i \bigotimes j \bigoplus k), i \in \{0, 1, .., N-1\}\}$, according to basic property of Galois field, $i \bigotimes j \bigoplus k$ is bounded in $\{0, 1, .., N-1\}$, so e_i is strictly increasing within each area, making these N elements distinct elements.

For Property 1, we consider two different areas k and k' in the same jth row, if the two areas share any elements in common, say one element in area

Algorithm 1 Constructing Orthogonal Assignment Table

Choose Galois field[14] of order $N = p^k$
$\{0, 1, .., p^k - 1\}$ are the elements of the field
$\bigoplus$ and $\bigotimes$ are addition and multiplication of the field.
Initialize $Table$ of N rows and N^2 columns
for $row = 0$ to $N - 1$ **do**
  for $area = 0$ to $N - 1$ **do**
    for $i = 0$ to $N - 1$ **do**
      $column := area \times N + i$
      $Table[row][column] := i \times N + (i \bigotimes row \bigoplus area)$
    end for
  end for
end for
return $Table$

k equals to another element in area k', these two elements must in the same position of two areas, otherwise $i \times N + (i \bigotimes j \bigoplus k) \neq i' \times N + (i' \bigotimes j \bigoplus k')$ for different position i and i', a contradiction. With the same position i, we have $i \times N + (i \bigotimes j \bigoplus k) = i \times N + (i \bigotimes j \bigoplus k')$, it is $i \bigotimes j \bigoplus k = i \bigotimes j \bigoplus k'$, then $k = k'$, also a contradiction, so every two areas in the same row share no element in common, and elements within each area are distinct elements, then Property 1 holds.

For Property 2, we consider any two areas(k and k') selected from different rows j and j'. Two areas share element(s) in common, is equivalent to that there exists i satisfy $i \times N + (i \bigotimes j \bigoplus k) = i \times N + (i \bigotimes j' \bigoplus k')$, which is equivalent to $i \bigotimes j \bigoplus k = i \bigotimes j' \bigoplus k'$, and every element in Galois field has an unique inverse of $\bigoplus$, denoted $-k$, the equation above is equivalent to $i \bigotimes (j \bigoplus -j') = k' \bigoplus -k$, which has one and only one solution to i in any Galois field if $j \bigoplus -j' \neq 0$, which is equivalent to $j \neq j'$. Q.E.D.

3.4 Support Nested Layers and Domains

In the simplest case, we have one big domain which contains many simple layers that only contain experiments, we can use one orthogonal assignment table to do our job. In real world scenes, we need nested layers and domains, in order to ensure orthogonal property in such cases, we followed two rules listed below.

1. Split traffic to buckets as many as possible in the outermost domain, like $N = 2^{30}$.

2. Construct new nested domains that contain buckets having number of power of 2, like 2^{26}, then consider these buckets a set of segments in the new domain, assigning these segments by using another orthogonal assignment table of proper size in the new domain. For 2^{26} segments we need a orthogonal assignment table of size $N = 2^{13}$.

4 Simulation Illustration

To illustrate the effects of using orthogonal assignment table, we compared type I error rates of AA tests in different settings.

4.1 Simulation Settings

We assume that there are $n = 2^{20}$ users with userid from 0 to $2^{20} - 1$, and they will be assigned to $N = 2^{10}$ buckets in each layer. We consider a simple problem of testing the revenue per user that is normally distributed. Revenue of users (denote as $R_1, R_2, .., R_n$) are n *i.i.d.* random variables from a normal distribution $N(10, 1)$ when no treatment is applied.

To evaluate the bias caused by treatment effect of other layers, imaging there is a treatment group with buckets range of $[0, 128)$ in first layer which brings a revenue lift of Δ to every user in it. These users are reallocated in the second layer. We then randomly choose 32 buckets in the second layer for two groups, 16 buckets each. An observation is collected for every user in these buckets according to normal distribution $N(10, 1)$, and an extra lift of Δ is added to the observation if the corresponding user is assigned to treatment group of the first layer. These observations are then used to conduct a two sample t-test(alpha=0.05) for the two groups, and we can finally figure out if type I error happened in this procedure. This procedure will repeat for 10000 times, type I error rate is then calculated based on these results.

4.2 Assign Traffic by Murmurhash2

We choose 368 and 60 as layer seeds of the two layers, and evaluate type I error rates of different Δ.

Table 3. Type I Error Rates of Murmurhash2

Δ	Type I Error Rate
0.1	0.0544
0.5	0.1084
1.0	0.2594

Table 3 shows that the type I error rate goes beyond the expected value of 0.05 with the growing of treatment effects of the first layer. The bigger the treatment effect is, the bigger the bias will be.

4.3 Assign Traffic by Orthogonal Assignment Table

We use orthogonal assignment table of size $N = 2^{10}$, row 368 and 60 are associated with the two layers. 2^{20} users are randomly assigned to 2^{20} segments which will be assigned to buckets by orthogonal assignment table next.

Table 4. Type I Error Rates of Orthogonal Assignment Table

Δ	Type I Error Rate
0.1	0.0513
0.5	0.0481
1.0	0.0497

Table 4 shows that the type I error rate is always in the %95 binomial confidence interval of $(0.0459, 0.0544)$ with the growing of treatment effects of the first layer, which implies that our method eliminated the bias between these two layers.

5 Real Online Experiment Assignment

In addition to the simulations test above, we have also conducted a real online experiment traffic assignment on the our experiment platform using the orthogonal table traffic assignment we have proposed compared to baseline traffic assignment and hash traffic assignment. Specifically, we have done the following:

1. Select an online experiment and one of its metrics that treated group is significantly different from the control group.
2. Apply different methods to assign traffics to the next layer separately.
3. Sample two A/A groups metric data from the assigned layer and use t-test to calculate the p-value between the two sampled A/A groups.

Our goal for the real experiment traffic assignment is two-fold. Firstly, we would like to compare the A/A groups type I error using different traffic assignment methods in a real application setting to complement the observations from simulations. In particular, we want to compare results using different traffic assignment methods with different treated ratios at assignment process step 1 above and different sampling ratios at assignment process step 3 above and see whether our proposed method is strong enough in different situations. For treated ratios, we means the ratio of the number of users in the treated group compared to the total number of users in the whole experiment. For sampling ratio, we means the ratio of the number of users sampled into the new A/A group each compared to the total experimented users. Secondly, as far as we know ,there are not standard methods to compare the effect of different traffic assignment methods, we would like to establish a process for running traffic assignment test in practice and draw conclusions using this process in a real world experiment.For baseline traffic assignment, we simply reshuffle the user in the first layer using MurmurHash2. We then compare orthogonal table traffic assignment and hash traffic assignment to baseline traffic assignment and find out which method works better in different sitiations.

5.1 Traffic Assignment Process

In the traffic assignment process, we would like to select a metric from an online experiment whose's treated group is significantly different from the controlled group. We randomly select an online experiment and one of its metric whose treated group is about 3% significantly different from the control group. The process of our online traffic assignment test works as the Figure 3 describes.

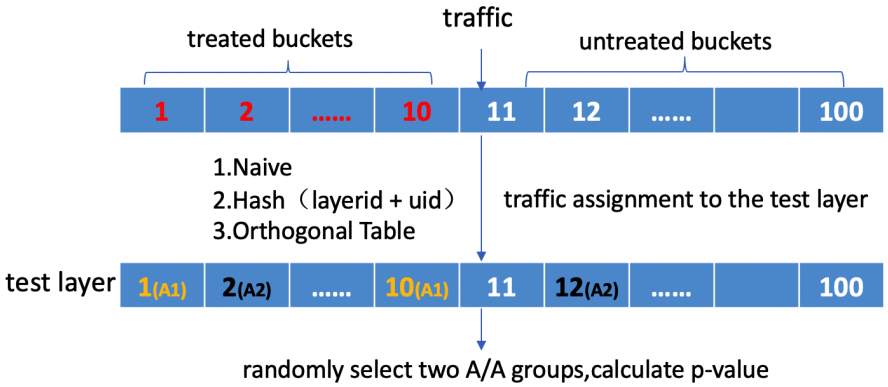

Fig. 3. Traffic Assignment Process

1. At the first step, we will use different ratios of treated buckets compared to controlled buckets by sampling users from the controlled buckets. In our example, we have 450k population users including 64k treated users, if we want a treated ratio of 20%, we can then sample 256k users from untreated users. In the following test, we will use different treated ratios separately.
2. At the second step, we assign the traffic to the test layer using three different methods, naive traffic assignment, hash traffic assignment and orthogonal table traffic assignment. After the traffic assignment, we can get the user behaviour data of the test layer.
3. At the third step, we randomly sample some buckets from the test layer into two A/A groups, and compute the p-value between the two sampled A/A groups. We will test different sampling ratios at this step and get the Type I error between the two A/A groups with different sampling ratios. For every single sampling ratio, we will try this step 10000 times and increase the A/A error counter by one if the p-value is less than 0.05 every round, then we can get the A/A type I error for this sampling ratio.

5.2 Online Experiment Results

We compute the type I error results of different treated ratios and different sampling ratios using different traffic assignment methods as described in section

above. The results are shown in Figure 4 for different treated ratios and in Figure 5 for different sampling ratios. We have the following observations. (i) In all cases, the type I error of the orthogonal table method is always near 0.05 which is the theoretic value when the significant level of A/A test is 0.05. In the different treated ratio case, the type I error of baseline and hash method increase as the treated ratio increase which means if we have larger treated ratio when we do online A/B test, we would probably get worse result using hash traffic assignment. (ii) (ii) The type I error is always near theoretic value using orthogonal table method which means no matter how many traffic ratios we have for treated group, the orthogonal table traffic assignment can works well.

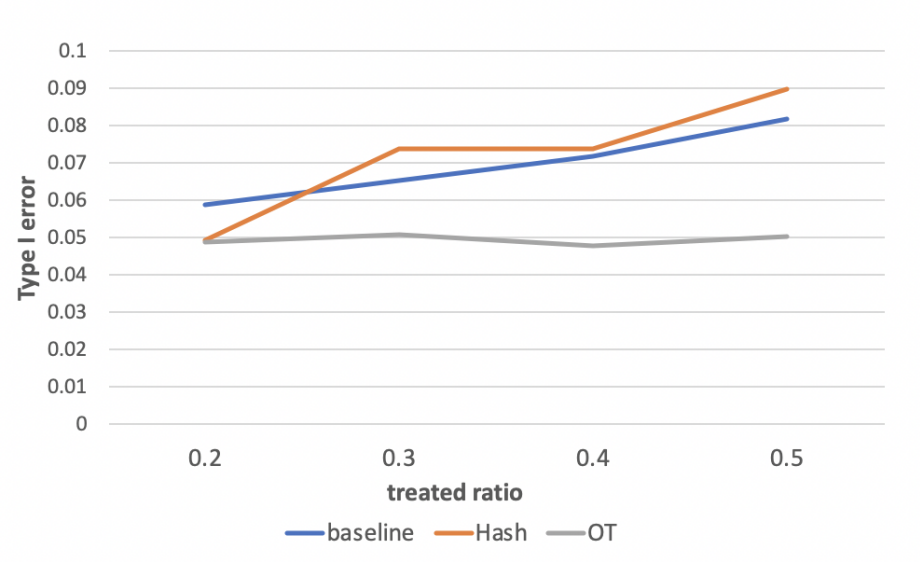

Fig. 4. Type I error with different treated ratios for sampling ratio 50% each A/A group

6 Conclusion

With correctness of Algorithm 1 rigorously proved, we can construct orthogonal table conveniently for theoretically orthogonal traffic assignment. Compared to hash functions with seeds, we solve the orthogonal problem in traffic assignment with theoretical guarantees. Simulations and real world cases also showed that our method have better reliability compared to non-cryptographic hash functions without theoretical guarantees.

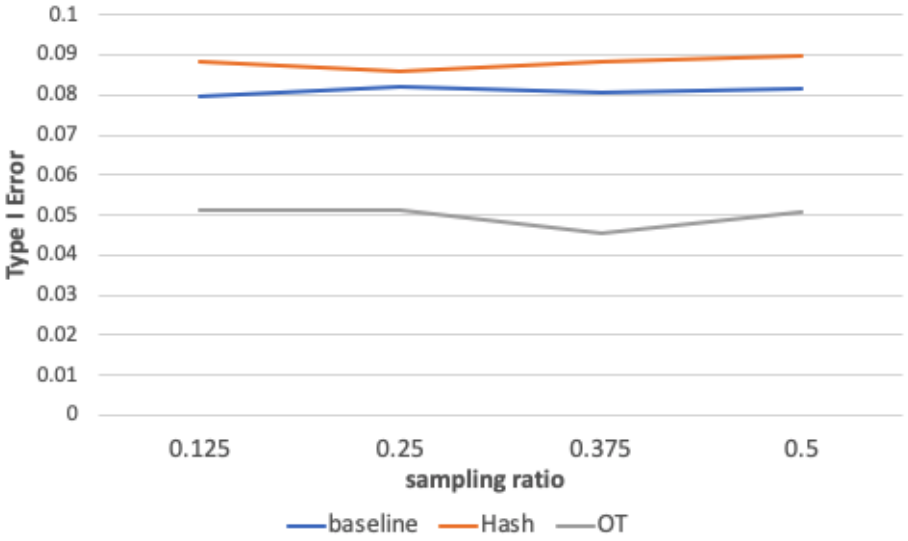

Fig. 5. Type I error with different sampling ratios for treated ratio 50%

References

1. Kohavi, Ronny, and Matt Round. "Front line internet analytics at Amazon. com." Santa Barbara, CA 170 (2004).
2. McKinley, Dan. "Design for Continuous Experimentation." (2012).
3. Bakshy, Eytan, Dean Eckles, and Michael S. Bernstein. "Designing and deploying online field experiments." Proceedings of the 23rd international conference on World wide web. 2014.
4. Tang, Diane, et al. "Overlapping experiment infrastructure: More, better, faster experimentation." Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. 2010.
5. Moran, Mike. "Multivariate testing in action: Quicken loan's regis hadiaris on multivariate testing." Biznology Blog by Mike Moran (2008).
6. Posse, Christian. "Key lessons learned building recommender systems for large-scale social networks." Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. 2012.
7. Kohavi, Ronny, et al. "Online experimentation at Microsoft." Data Mining Case Studies 11.2009 (2009): 39.
8. Amatriain, Xavier, and Justin Basilico. "Netflix recommendations: beyond the 5 stars (part 1)." Netflix Tech Blog 6 (2012).
9. McFarland, Colin. Experiment!: Website conversion rate optimization with A/B and multivariate testing. New Riders, 2012.
10. Katzir, Liran, Edo Liberty, and Oren Somekh. "Framework and algorithms for network bucket testing." Proceedings of the 21st international conference on World Wide Web. 2012.
11. Quora 2010, https://www.quora.com/What-is-Zyngas-core-competency

12. Gupta, Somit, et al. "Top challenges from the first practical online controlled experiments summit." ACM SIGKDD Explorations Newsletter 21.1 (2019): 20-35.
13. Wikipedia: Murmurhash, https://en.wikipedia.org/wiki/MurmurHash
14. Wikipedia: Galois Field, https://en.wikipedia.org/wiki/Finite_field