Multi-armed bandits: When to experiment and when to optimize

Skip Ahead

When to use a multi-armed bandit
Set up a multi-armed bandit optimization
Interpreting MAB results
  MABs do not show statistical significance
  MABs do not use a baseline
  Improvement over equal allocation
MAB optimization vs. A/B testing: a demonstration
FAQs

THIS ARTICLE WILL HELP YOU:

• Understand what Optimizely’s multi-armed bandit optimization is and how it works
• Decide when to choose a multi-armed bandit optimization instead of an A/B experiment

In Optimizely, you can run tests to do one of two things: either experiment or optimize.

• When you experiment, you’re trying to test a hypothesis or validate a claim. The goal is to determine whether a variation is fundamentally different (via statistical significance) with the aim of generalizing learnings from that knowledge into future deployments or experiments.

• When you optimize, on the other hand, you’re using a set-it-and-forget-it algorithm designed to squeeze as much lift from a set of variations as possible, without concern for visibility into whether a variation is fundamentally better or worse. What that means is, you’re ignoring statistical significance in favor of maximizing your goal.

Multi-armed bandit (MAB) optimizations aim to maximize performance of your primary metric across all your variations. They do this by dynamically re-allocating traffic to whichever variation is currently performing best. This will help you extract as much value as possible from the leading variation during the experiment lifecycle, so you avoid the
opportunity cost of showing sub-optimal experiences.

In other words, the better a variation does, the more traffic a multi-armed bandit will send its way. A/B tests don't do this. Instead, they keep traffic allocation constant for the experiment's entire lifetime, no matter how each variation performs:

![Graph comparison between Standard A/B and Multi-Armed Bandit]

Click here for a thorough explanation of what's happening in this graph.

When to use a multi-armed bandit

Here are a couple cases that may be a better fit for a multi-armed bandit optimization than a traditional A/B experiment:

- **Promotions and offers**: users who sell consumer goods on their site often focus on driving higher conversion rates. One effective way to do this is to offer special promotions that run for a limited time. Using a multi-armed bandit optimization (instead of running a standard A/B experiment) will send more traffic to the over-performing variations and less traffic to the underperforming variations.

- **Headline testing**: headlines are short-lived content which lose relevance after a fixed amount of time. If a headline experiment takes just as long to reach statistical significance as the lifespan of a headline, then any learnings gained from the experiment will be irrelevant going forward. Therefore, a multi-armed bandit optimization is a natural choice to allow you to maximize your impact without worrying about balancing experiment runtime and the natural lifespan of a headline.

Set up a multi-armed bandit optimization

To set up multi-armed bandit optimization on your experiment, select *Multi-Armed Bandit* from the *Create New...* dropdown when you first create your optimization.
You can use multi-armed bandit optimizations in Full Stack; however, you can't use them for feature rollouts in Feature Management.

Interpreting MAB results

If you're an Optimizely user, you probably have a good understanding of how to interpret the results of a traditional A/B test. Those interpretations won't work for MABs, for two important reasons:

- Multi-armed bandits don't generate statistical significance, and
- Multi-armed bandits don't use a control or a baseline experience

Because of this, the MAB results page focuses on **improvement over equal allocation** as its primary summary of your experiment's performance.

**MABs do not show statistical significance**

With a traditional A/B test, the goal is **exploration**: collecting data to discover if a variation performs better or worse than the control. This is expressed through the concept of statistical significance.

Statistical significance tells you whether a change had the effect you expected. You can use those lessons to make your variations better each time. Fixed traffic allocation strategies are usually the best ways to reduce the time it takes to reach a statistically significant result.

On the other hand, Optimizely's multi-armed bandit algorithms are designed for **exploitation**: MABs will aggressively push traffic to whichever variations are performing best at any given moment, because the MAB doesn't consider the reason for that superior performance to be all that important.

Since multi-armed bandits essentially ignore statistical significance, Optimizely will do the same. This is why **statistical significance does not appear on the results page for MABs**: It avoids confusion about the purpose and meaning of multi-armed bandit optimizations.

**MABs do not use a baseline**

In a traditional A/B test, statistical significance is calculated relative to the performance of one baseline experience. But MABs don't do this. They're intended to explicitly evaluate the tradeoffs between all variations at once, which means there is no control or baseline experience to compare to.

What's more, MABs are "set-and-forget" optimizations. In an A/B test, you follow up an experiment with a decision: do
you deploy a winning variation, or stick with the control? But since MABs continuously make these decisions throughout the experiment’s lifetime, there’s never any need for a baseline reference point for that decision, because you’ll never need to make it yourself.

**Improvement over equal allocation**

Improvement over equal allocation represents the gain in total conversions in the current MAB test over a hypothetical state, in which an A/B test with fixed, equal traffic allocation had been run instead.

Optimizely estimates the reward of equal allocation by calculating the average reward per visitor for every arm and every time period, then multiplying this number by the number of visitors that would have been assigned had equal allocation been used.

To do this, Optimizely first breaks up the history of your optimization into a series of time spans (or epochs). It then performs the following procedure:

Remember, traffic allocation remains the same for each epoch.

1. Optimizely computes the average reward per visitor for each arm of your Multi-Armed Bandit optimization.
2. The sum total traffic across all arms in a given epoch is then divided equally among the arms.
3. For each arm, Optimizely multiplies the traffic total from step 2 by the average reward per visitor. This generates an estimate of the reward for that arm, in that epoch, under the equal allocation method.
4. These quantities are collected for each arm and each epoch, and then summed to generate an estimate of the total equal allocation reward.
5. Finally, Optimizely subtracts this estimate from the total reward the algorithm actually provided. This is the improvement over equal allocation.

See the math that makes it work

Assume $T$ time epochs (index $t$ in $1, \ldots, T$) and $k$ (index $a$ in $1, \ldots, k$) arms. The reward in time epoch $t$ from arm $a$ is $r_{a,t}$. The number of visitors sent to arm $a$ in time epoch $t$ is $n_{a,t}$.

The total reward from arm $a$ is

$$\sum_{t=1}^{T} r_{a,t}$$

and the total reward from all arms—in other words, the reward generated by the algorithm—is

$$R_T = \sum_{a=1}^{k} \sum_{t=1}^{T} r_{a,t}$$

We can calculate this quantity exactly. The average reward per visitor for arm $a$ at epoch $t$ is

$$r_{a,t} / n_{a,t}$$

If we had used equal allocation during epoch $t$, then the traffic sent to each arm would be

$$\frac{1}{k} \sum_{b=1}^{k} n_{b,t}$$

Hence, the expected reward of equal allocation for arm $a$ at epoch $t$ is

$$\frac{1}{k} \sum_{b=1}^{k} n_{b,t}$$
The total expected reward over all times and all arms is

\[
\frac{r_{a,t}}{n_{a,t}} \sum_{b=1}^{k} n_{b,t}
\]

Therefore, the estimated gain of the reward of the MAB algorithm and the reward of uniform allocation is

\[
R_{T} - R_{U,T} = \sum_{a=1}^{k} \sum_{t=1}^{T} \left( \frac{r_{a,t}}{n_{a,t}} \sum_{b=1}^{k} n_{b,t} \right) - \frac{r_{a,t}}{n_{a,t}} \sum_{b=1}^{k} n_{b,t}
\]

MAB optimization vs. A/B testing: a demonstration

In this head-to-head comparison, simulated data is sent to both an A/B test with fixed traffic distribution and a multi-armed bandit optimization. Traffic distribution over time and the cumulative count of conversions for each mode are both observed. The true conversion rates driving the simulated data are:

- Original: 50%
- Variation 1: 50%
- Variation 2: 45%
- Variation 3: 55%

The multi-armed bandit algorithm senses that Variation 3 is higher-performing from the start. Even without any statistical significance information for this signal (remember, the multi-armed bandit does not show statistical significance), it still begins to push traffic to Variation 3 in order to exploit the perceived advantage and gain more conversions.

For the ordinary A/B experiment, the traffic distribution remains fixed in order to more quickly arrive at a statistically significant result. Because fixed traffic allocations are optimal for reaching statistical significance, MAB-driven experiments generally take longer to find winners and losers than A/B tests.

By the end of the simulation, the multi-armed bandit has optimized the experiment to achieve roughly 700 more conversions than if traffic had been held constant.
FAQs

What algorithms or frameworks does the multi-armed bandit support? For binary metrics, Optimizely uses a procedure inspired by Thompson Sampling ([Russo, Van Roy 2013](#)). Optimizely characterizes each variation as a Beta distribution, where its parameters are the variation's observed number of conversions and visitors. These distributions are sampled several times, and Optimizely allocates traffic to the variations according to their win ratio.

For numeric metrics, Optimizely uses a form of Epsilon Greedy, where a small fraction of traffic is uniformly allocated to all variations and a large amount is allocated to the variation with the highest observable mean.

Does the multi-armed bandit algorithm work with MVT and Personalization? Yes. To use multi-armed bandit in MVT, select Partial Factorial. In the Traffic Mode dropdown, select Multi-Armed Bandit.

In Personalization, multi-armed bandit can be applied on the experience level; this works best if you have two variations aside from the holdback.

How often does the multi-armed bandit make a decision? The multi-armed bandit model is updated hourly. If you need a different frequency for model updates, please let us know.

Why is a baseline variation listed on the Results page for my multi-armed bandit campaign? In MVT and Personalization, your Results page will still designate one variation as a baseline. However, this designation doesn't actually mean anything, since MABs do not measure success relative to a baseline variation. It's just a label that will have no effect on your experiment or campaign.

You should not see a baseline variation when using MAB with a Web or Full Stack experiment.

What happens if I change my primary metric? If you change the primary metric mid-experiment in MVT or Personalization, the multi-armed bandit will begin optimizing for the new primary metric, instead of the one you originally selected. For this reason, we suggest you do not change the primary metric once you begin the experiment or campaign.

It is not possible to change your primary metric in Optimizely X Web or Full Stack once your experiment has begun.

What happens when I stop or pause a variation? If you pause or stop a variation, Optimizely’s multi-armed bandit will ignore data from those variations data when it adjusts traffic distribution among the remaining live variations.

How do multi-armed bandits handle conversion rates that change over time, and Simpson's Paradox? Optimizely uses an exponential decay function that weighs recent visitor behavior more strongly, to better adapt to the effect of time variation more quickly. This approach gives less weight to earlier observations and more weight to recent ones.

On top of that, Optimizely reserves a portion of traffic for pure exploration, so that time variation is easier to detect.