# Maximizing the Success Probability of Policy Allocations in Online Systems

Artem Betlei, Mariia Vladimirova, Mehdi Sebbar, Nicolas Urien, Thibaud Rahier, Benjamin Heymann

# Maximizing the Success Probability of Policy Allocations in Online Systems

**Artem Betlei,** [1] **Mariia Vladimirova,** [1] **Mehdi Sebbar,** [2]
**Nicolas Urien,** [2] **Thibaud Rahier,** [1] **Benjamin Heymann** [1]

[1] Criteo AI Lab, France
[2] Criteo Ad Landscape, France
{a.betlei, m.vladimirova, m.sebbar, n.urien, t.rahier, b.heymann}@criteo.com



**Abstract**

The effectiveness of advertising in e-commerce largely depends on the ability of merchants to bid on and win impressions for their targeted users. The bidding procedure is highly complex due to various factors such as market competition, user behavior, and the diverse objectives of advertisers. In this paper we consider the problem at the level of user timelines instead of individual bid requests, manipulating full policies (i.e. pre-defined bidding strategies) and not bid values. In order to optimally allocate policies to users, typical multiple treatments allocation methods solve knapsack-like problems which aim at maximizing an expected value under constraints. In the industrial contexts such as online advertising, we argue that optimizing for the probability of success is a more suited objective than expected value maximization, and we introduce the `SuccessProbaMax` algorithm that aims at finding the policy allocation which is the most likely to outperform a fixed reference policy. Finally, we conduct comprehensive experiments both on synthetic and real-world data to evaluate its performance. The results demonstrate that our proposed algorithm outperforms conventional expected-value maximization algorithms in terms of success rate.

## 1 Introduction

Optimizing marketing effectiveness relies on using individualized bidding policies, exploiting the fact that each user responds differently. A *policy* may include a set of rules or actions over an extended period of time, e.g., cash bonuses, promotion and display ad shown to consumers on online platforms. Without loss of generality, we take the narrow view of bidding for display advertising in order to ground our research into a real life application. In this context, the task at hand is to specify a full bidding strategy (the policy) on the future advertisement opportunities for each given users during a given time period.

In practice, it is typical to have a fixed budget allocated to a campaign. From an advertising perspective, a bidding strategy must maximize the total expected revenue while ensuring that the expected total cost does not exceed a specified budget.

Usually, this problem is modeled as a multiple choice knapsack problem (Demirović et al. 2019; Zhou et al. 2023) with the objective to select at most one item (bid policy) from each user such that the sum of the weights (expected cost) of selected items does not exceed the capacity (budget) while the total *reward* (expected revenue) is maximized. This problem is known to be NP-hard, although it can be tackled with mixed integer linear programming or through Lagrangian relaxation (Sinha and Zoltners 1979).

From a causal perspective, it is classical to consider every individual ad as a treatment, and the optimization problem goal is to maximize the total causal effect of these treatments by correctly assigning treatments to users. There exist various approaches for individual treatment assignment that differ by the objective function they optimize: learning models to predict either outcomes, causal effects or directly the optimal treatment assignment. Fernández-Loría et al. (2022) compare these approaches analytically and show that the assignment learners optimize the bias-variance tradeoff with respect to decision-making errors.

Optimization at the opportunity –or bid –level, which we refer as *bid by bid* optimization, requires to attribute each observed reward to the action that actually caused it, e.g. each conversion must be attributed to a shown ad. This attribution problem is very complex as there usually are several ads displayed in the few hours preceding each conversion (Bompaire, Gilotte, and Heymann 2021; Bompaire, Désir, and Heymann 2021; Ji and Wang 2017; Dalessandro et al. 2012). It causes fundamental problems in the estimation of the causal effects and makes the bid by bid optimization extremely difficult in practice.

Furthermore, display advertising campaigns, like many other online systems, are operated under several business and technical constraints. In particular, it is typical for an advertising campaign to have a budget constraint. Several algorithms allow adapting bid by bid optimization techniques to such constraints (Castiglioni et al. 2022; Conitzer et al. 2022). While these algorithms have their merits and are largely deployed in practice, they are, however, poorly suited for *causal* bid by bid methods. This is because (a) typical causal methods inherently suppose the absence of causal interaction between the treatment units — such assumption is in general violated when mixing causal method for bid by bid optimization and budget pacing; (b) the overall methodology needs to trade off marginal value and marginal future total cost (Bompaire, Gilotte, and Heymann 2021), which is

arguably intractable at the bid level.

Our first idea is to reformulate the problem at the user timeline level (i.e. considering all the bid requests and subsequent events relative to a user along a given time period) which implies to consider entire policies instead of individual bids. With this new formulation, the optimal policy allocation search is framed as a multiple treatment allocation problem, and the causal effects (cost and value) of policies are much easier to estimate than that of individual bids. Our approach is not to be understood as in competition with usual bid by bid design approaches (Moriwaki et al. 2021) but rather complementary. Indeed, any bid by bid design approach could be included as one of the candidate policies we wish to choose from when allocating policies to users with our methodology. If a bid by bid design policy happens to be globally optimal, our method will simply conclude that the optimal policy allocation consists in assigning this policy to every user.

However, we claim that searching for the policy allocation function which maximizes an expected value under an expected cost constraint (which is typically done in treatment allocation problems) is not always the best objective. In a large organization, it is often necessary to have guidelines that allow for consistent decision-making regarding product design and improvements. Without such guidelines, individuals cannot handle trade-offs between different quantities (for example, quality and volume) consistently across the whole organization. One may think about designing medication (which should be efficient but also avoid negative side-effects) or electrical batteries (which should have a big enough capacity while not relying too much on rare materials). This leads to the definition of a *success* across organizations, e.g. in online advertising, it corresponds to increasing generated value *without increasing the cost* with respect to a reference outcome. Taking this as a premise, the (constrained) maximization of a single quantity –such as revenue –is not anymore the right criterion as it does not account for the uncertainty underlying the phenomenon at play, nor does it account for what will be considered a success.

This motivates the focus on finding the policy allocation resulting in the *highest probability of success*. While every metric has its pros and cons, we believe a focus on success probability, with a very flexible notion of *success*, is of particular operational interest, see Fig. 1 for an illustrative example (we refer to Section 3 for a detailed description).

In summary, this work presents the following contributions:

- We formally propose the idea of framing the optimization problem at the policy level instead of focusing on bid by bid design, and mathematically formalize both the expected value maximization and the success probability maximization problems.
- We develop a novel customized solution to address the specificities of the success probability optimization problem.
- Finally, we present a series of numerical experiments which were conducted on both synthetic and real-world data, showing that our approach outperforms traditional value maximization methods in terms of success rate guarantees.

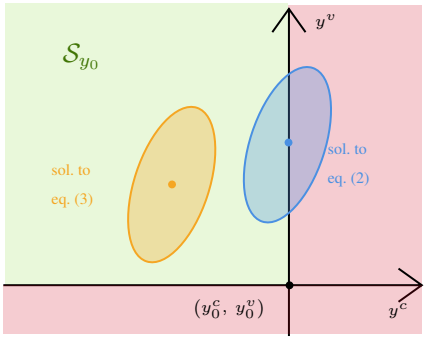


Figure 1: Distributions the (cost,value) outcome vector $\mathbf{Y} = (Y^c, Y^v)$. In blue for the solution of (2) (maximization of $\mathbb{E}[Y^v]$ under the condition $\mathbb{E}[Y^c] \leq y_0^c$) ; and in orange the solution of (3) (maximization of the success probability).

## 2 Problem formulation

**Preliminary considerations** Throughout this section, we will implicitly refer to a given time period $\tau$ of length $\Delta t$, i.e. $\tau = [t_0, t_0 + \Delta t]$. We consider a given advertiser $Adv$ who has a fixed budget $C$ to spend over period $\tau$.

**Set of candidate policies** We assume given a set of $K$ *candidate policies* $\Pi = \{\pi_0, \pi_1, \ldots, \pi_{K-1}\}$ each encapsulating *bidding strategies* that may be applied by $Adv$ to each user **consistently throughout the period** $\tau$. The *reference policy* $\pi_0$ is the default bidding strategy used by $Adv$ (typically corresponding to the strategy which is already rolled out in production for this advertiser). This set of policies $\Pi$ can be thought of as a collection of potential treatments in a *multiple treatment allocation* problem. Note that we do not consider treatments at the level of bidding opportunities here, but at the level of an extended time period, during which we apply policies –or bidding strategies –which each have an integrated way to decide on how to bid on each user for all the opportunities that will arise during period $\tau$.

**Random variables and potential outcomes** Considering the above setup, and with respect to any given user $u$ targetable by $Adv$, we define the following random variables:

- $\mathbf{X} \in \mathcal{X} \subset \mathbb{R}^d$ contains a snapshot of the *features* of $u$ captured at time $t_0$,
- $\mathbf{Y} = (Y^v, Y^c) \in \mathcal{Y} \subset \mathbb{R}_+^2$ contains respectively the *value* generated by $u$ in favor of $Adv$ during period $\tau$ and the *cost* $Adv$ spent to advertize to $u$.

For any $\pi \in \Pi$, we denote $\mathbf{Y}(\pi) = (Y^v(\pi), Y^c(\pi))$ the *potential outcomes* (Rubin 1974) we would have observed **had** $\pi$ **been applied to** $u$ during $\tau$. In what follows, we consider the tuple $(\mathbf{X}, \mathbf{Y}(\pi_0), \ldots, \mathbf{Y}(\pi_{K-1}))$ has an underlying probability distribution $\mathbb{P}$, which we will overload by simplicity to designate its marginals and conditionals. All expectancy notations $\mathbb{E}$ will refer implicitly to $\mathbb{P}$.

**Factuals and counterfactuals** Assuming that we apply policy $\pi_u \in \Pi$ to a given user $u$ during $\tau$, we denote $\mathbf{y}_u = (y_u^v, y_u^c)$ the corresponding realization of the outcome variable $\mathbf{Y}$, and $\{\mathbf{y}_u(\pi)\}_{\pi\in\Pi}$ the corresponding realizations of the potential outcomes variables $\{\mathbf{Y}(\pi)\}_{\pi\in\Pi}$. In that case, $\mathbf{y}_u = \mathbf{y}_u(\pi_u)$ is called the observed *factual outcome* and the $\{\mathbf{y}_u(\pi_u)\}_{\pi\in\Pi\setminus\{\pi_u\}}$ are the un-observed *counterfactual outcomes*.

**Population random variables** Let $\mathcal{U} = \{1, \dots, N\}$ be the set of users who are targetable by $Adv$ during period $\tau$. We have access to a randomized controlled trial (RCT) –in this case also called an online controlled experiment or A/B test –on population $\mathcal{U}$ during this period, **randomly** assigning to each of those $N$ users the $K$ potential policies in $\Pi$.

Formally, let $\{K_u\}_{u\in\mathcal{U}}$ be N i.i.d. uniform categorical variables with values in $\{0, \dots, K-1\}$. Each $u \in \mathcal{U}$ is assigned to the policy $\pi_{K_u}$ during $\tau$, resulting in the definition of the collection $\{(\mathbf{X}_u, \mathbf{Y}_u)\}_{u\in\mathcal{U}}$, where the $(\mathbf{X}_u, \mathbf{Y}_u) = (\mathbf{X}_u, \mathbf{Y}_u(\pi_{K_u}))$. We will denote $\mathbb{P}$[1] the probability distribution of $\{(\mathbf{X}_u, \mathbf{Y}_u)\}_{u\in\mathcal{U}}$. Lastly we assume that, in expectation, $Adv$ exactly spends their advertising cost budget $C$ during $\tau$ had they assigned the default policy $\pi_0$ to every user, i.e. $\mathbb{E}\left[\sum_{u\in\mathcal{U}} Y_u^c(\pi_0)\right] = C$.

## 2.1 Expected value maximization problem

**At the user level** In the setup we introduced, one aim may be to find an *optimal policy allocation*, i.e. a mapping from $\mathcal{X}$ to policies from $\Pi$ so that *expected total value* generated in favor of $Adv$ is maximized, while respecting (in expectation) the total budget constraint.

Formally, we are looking for a solution $\phi^* : \mathcal{X} \to \Pi$ to the problem:

$$\max_{\phi\in\Pi^{\mathcal{X}}} \mathbb{E}\left[\sum_{u\in\mathcal{U}} Y_u^v(\phi(\mathbf{X}_u))\right] \text{ s.t. } \mathbb{E}\left[\sum_{u\in\mathcal{U}} Y_u^c(\phi(\mathbf{X}_u))\right] \leq C. \tag{1}$$

In practice, $\Pi^{\mathcal{X}}$ is very large and hard to explore efficiently, making (1) a difficult problem, especially since it involves estimation of $K$ potential outcomes in parallel. A crucial observation is that the problem may be simplified by reducing it to a partition of the space $\mathcal{X}$, which leads to a reparametrization of (1), as explained in the next subsection.

**Assuming a given partitioning of the user space** We consider given a partition function $\gamma : \mathcal{X} \to \mathcal{G}$ where $\mathcal{G} = \{1, \dots, M\}$ contains the indexes of the partition components (or buckets). Reasoning at a bucket-level instead of the user-level is practical in causal estimation setups since it enables to circumvent the fundamental problem of causal inference (Betlei et al. 2021) and is more compliant with privacy restrictions (Kleber 2019). A reasonable partitioning can be chosen by the domain knowledge or recursively with causal trees through heterogeneous treatment effect estimation (Athey and Imbens 2016; Wager and Athey 2018; Tu et al. 2021; Ai et al. 2022).

Given the partition function $\gamma$, we propose to simplify problem (1): instead of searching through all $\phi$s in $\Pi^{\mathcal{X}}$, we restrict our search to the allocations of the form $\psi \circ \gamma$ where $\psi \in \Pi^{\mathcal{G}}$. In short, we look for allocation functions that assign all users belonging to the same bucket $g \in \mathcal{G}$ to the same policy $\pi \in \Pi$.

Formally, this leads to the reparametrized problem, where we are looking for a solution $\psi^* : \mathcal{G} \to \Pi$ to the problem:

$$\max_{\psi\in\Pi^{\mathcal{G}}} \mathbb{E}\left[\sum_{u\in\mathcal{U}} Y_u^v(\psi(G_u))\right] \text{ s.t. } \mathbb{E}\left[\sum_{u\in\mathcal{U}} Y_u^c(\psi(G_u))\right] \leq C. \tag{2}$$

where $G_u = \gamma(\mathbf{X}_u)$ for all $u \in \mathcal{U}$.

**Solving the expected value maximization problem** The value expectation maximization problem formalized in (2) may be solved using mixed integer linear programming or Lagrangian relaxation approaches, which make the problem tractable in practice despite being NP-Hard (Sinha and Zoltners 1979). Nevertheless, the knapsack formulation remains a proxy to the marketing problem and its solution does not always align with the business goal.

**Remark 1 –mix of A and B rollout** If number of buckets $M = 1$, we assign **all users** to the same policy. This corresponds to a typical rollout decision in an online advertising company: we are A/B testing multiples policies, then depending on the results choosing which one should be rolled out. Our setup allows for a rollout of a **mix** of tested policies, given by function $\psi$.

**Remark 2 –relaxing the allocation space** We can relax problems (1) (2) by allowing for *soft allocations*, i.e. mappings from $\mathcal{X}$ (resp. $\mathcal{G}$) to $\Delta = \Delta(\Pi)$ where $\Delta$ denotes all categorical distributions with values in $\Pi$:

$$\Delta := \left\{ (p(k))_{k\in[\![1,K-1]\!]} \in [0,1]^K \text{ s.t. } \sum_k p(k) = 1 \right\}.$$

For $\psi \in \Delta^{\mathcal{G}}$ and $g \in \mathcal{G}$ and for convenience of notations, we will refer to the $k$th component of $\psi(g)$ –i.e. the probability for $\psi$ to assign a user in bucket $g$ to policy $\pi_k$ –as $\psi(g, k)$.

## 2.2 Success probability maximization problem

In this section, we will focus on the case where we are given a partitioning of the user space and consider more general soft allocation setup presented in *Remark 2* at the end of the previous section.

Instead of searching for the allocation that maximizes the expected value under constraint as in (1) and (2), one can also be interested in maximizing their *success probability*, especially in cases where the variance of the variables at play is high. For instance, a policy $\psi^*$ that satisfies (2) might deliver very bad values occasionally. As explained in the introduction, the risk-aversion of industrial players often motivates them to prefer reliable small-increments to uncertain substantial ones.

Instead, we suppose there is an agreement beforehand on the definition of the *success* of a given policy allocation function $\psi : \mathcal{G} \to \Delta$ through the characterization of a convex region $\mathcal{S} \subset \mathcal{Y}$ such that "$\psi$ is successful on a the set of

[1] We drop the reference to $\mathcal{U}$ in $\mathbb{P}$ for simplicity.

users $\mathcal{U}$" is equivalent to $\sum_{u\in\mathcal{U}} \mathbf{Y}_u(\psi) \in \mathcal{S}$, where for any $\psi \in \Delta^{\mathcal{G}}$ we denote by simplicity $\mathbf{Y}(\psi) := \mathbf{Y}(\psi(\gamma(\mathbf{X}))$.

The success probability maximizing policy $\psi^*$ is therefore a solution to

$$\max_{\psi\in\Delta^{\mathcal{G}}} \mathbb{P}\left(\sum_{u\in\mathcal{U}} \mathbf{Y}_u(\psi) \in \mathcal{S}\right) = \max_{\psi\in\Delta^{\mathcal{G}}} \mathbb{E}\left[\mathbb{I}_{\mathcal{S}}\left(\sum_{u\in\mathcal{U}} \mathbf{Y}_u(\psi)\right)\right], \tag{3}$$

where $\mathbb{I}_{\mathcal{S}}$ is the indicator function of the success set $\mathcal{S}$.

**Example** Our problem is defined with respect to any convex success region $\mathcal{S} \subset \mathcal{Y}$. In practice, we will consider success regions relative to a fixed $\mathbf{y}_0 = (y_0^v, y_0^c)$, of the form

$$\mathcal{S}_{\mathbf{y}_0} = \{(y^v, y^c) \in \mathcal{Y} \text{ s.t. } y^v > y_0^v \text{ and } y^c \leq y_0^c\},$$

where $\mathbf{y}_0$ should be interpreted as a *reference outcome*, for example the outcome we observe if we assign the reference policy to every user $\sum_{u\in\mathcal{U}} \mathbf{Y}_u(\pi_0)$. The success region $\mathcal{S}_{\mathbf{y}_0}$ corresponds to all outcomes with an *increased value and decreased cost* with respect to the reference value and cost $\mathbf{y}_0$.

In Figure 1, $\mathcal{S}_{\mathbf{y}_0}$ is displayed in green and its complementary $\bar{\mathcal{S}}_{\mathbf{y}_0}$ in red. We represent the distributions of the outcome $\mathbf{Y}$ for the respective allocations output by (i) the possible solution to (3) (maximization of the success probability) in orange and (ii) the possible solution to 2 (maximization of $\mathbb{E}[Y^v]$ under the condition $\mathbb{E}[Y^c] \leq y_0^c$) in blue. The orange outcome has a very high probability to be in $\mathcal{S}_{\mathbf{y}_0}$, even if it generates a bit less value on average than the blue one, which presents a high risk of being outside of the success region (for example by breaking the cost constraint).

## 3 The `SuccessProbaMax` algorithm

In this section, we present solutions for the problems (1) and (2). We will focus on the bucket-level versions of these problems, and therefore assume given a fixed partitioning $\gamma : \mathcal{X} \to \mathcal{G} = \{1, \dots, M\}$ of the feature space. This function could have been given by an expert or learned by machine learning algorithm, but it is not the focus of this work.

### 3.1 Gaussian parametrization of the problem

In this subsection, we introduce a novel method to solve the success probability maximization problem. This optimization problem, stated in (3), presents several non-trivial difficulties: (a) the indicator function which expectancy we are maximizing is not continuous on $\Delta^{\mathcal{G}}$ and (b) the criteria we wish to maximize is non-concave. We use

$$\mathbf{Y}_{g,k} = \sum_{u\in\mathcal{U}} \mathbf{Y}_u(\pi_k)\mathbb{I}\left(\gamma(\mathbf{X}_u) = g\right),$$

as a compact notation for the total expected outcome from users in bucket $g$, had they been allocated to policy $\pi_k$. Assuming that the buckets in $\mathcal{G}$ are approximately balanced in size (each containing $\approx N/M$ data points), we observe around $N/(MK)$ i.i.d. realizations to estimate each $\mathbf{Y}_{g,k}$. Let $\boldsymbol{\mu}_{k,g}$ and $\boldsymbol{\Sigma}_{k,g}$ be the mean and covariance matrix of the potential outcomes which contain value and cost.

For any soft allocation $\psi : \mathcal{G} \to \Delta$ –which maps all buckets in $\mathcal{G}$ to a stochastic mix of policies in $\Pi$–the total expected outcome under allocation $\psi$

$$\mathbf{Y}(\psi) = \sum_k \sum_g \psi(g,k)\mathbf{Y}_{g,k}, \quad \sum_k \psi(g,k) = 1.$$

The distributions $\psi(g,k)\mathbf{Y}_{g,k}$ are independent, therefore, we can use the Lyapunov central limit theorem and approximate the total expected outcome by a Gaussian distribution

$$\mathbf{Y}(\psi) \sim \mathcal{N}\left(\boldsymbol{\mu}(\psi), \boldsymbol{\Sigma}(\psi)\right),$$

where $\boldsymbol{\mu}(\psi) = \sum_k \sum_g \psi(g,k)\boldsymbol{\mu}_{g,k}$ and $\boldsymbol{\Sigma}(\psi) = \text{Var}\left[\sum_k \sum_g \psi(g,k)\mathbf{Y}_{g,k}\right]$. Depending on assumptions, $\boldsymbol{\Sigma}(\psi)$ can be a linear or quadratic function of $\psi(g,k)$ due to different sources of randomness which lead to different variances. We assume that $\boldsymbol{\Sigma}(\psi) = \sum_k \sum_g \psi(g,k)\boldsymbol{\Sigma}_{g,k}$ (more details in Supplementary).

### 3.2 Parameters estimation

When we do not have a direct access to parameters $\boldsymbol{\mu}_{g,k}$ and $\boldsymbol{\Sigma}_{g,k}$, we need to estimate them. In practice, the parameters are estimated on a **randomized control trial (RCT) dataset** $\mathcal{D} = \{(\mathbf{x}_u, \mathbf{y}_u)\}_{u\in\mathcal{U}}$ –realization of the collection $\{(\mathbf{X}_u, \mathbf{Y}_u)\}_{u\in\mathcal{U}}$ introduced in the last section. More precisely, $(\mathbf{x}_u, \mathbf{y}_u) = (\mathbf{x}_u, \mathbf{y}_u(\pi_{k_u}))$ are i.i.d. realizations of $(\mathbf{X}, \mathbf{Y}(\pi_{k_u}))$, where $\{k_u\}_{u\in\mathcal{U}}$ are i.i.d. realization of a uniform categorical variable on $\{0, \dots, K-1\}$. For $k \in [\![0, K-1]\!]$ and $g \in \mathcal{G}$, we will refer to the restrictions of $\mathcal{D}$ to points $u \in \mathcal{U}$ for which $\gamma(\mathbf{x}_u) = g$ and $k_u = k$ as $\mathcal{D}_{g,k}$.

To estimate parameters, we choose **mean and variance estimation methods** (e.g. bootstrapping Efron (1979)) which take as input a dataset $\mathcal{D}_{g,k}$ containing realizations of $\mathbf{Y}$ for a given bucket $g$ and policy $k$ and return respectively its mean $\{\hat{\boldsymbol{\mu}}_{g,k}\}$ and variances $\{\hat{\boldsymbol{\Sigma}}_{g,k}\}$.

### 3.3 Gradient computation

In the following, for $\psi \in \Delta^{\mathcal{G}}$, we will denote for clarity purposes $\mathcal{C}(\psi) = \mathbb{P}\left(\sum_{u\in\mathcal{U}} \mathbf{Y}_u(\psi) \in \mathcal{S}\right) = \mathbb{E}\left[\mathbb{I}_{\mathcal{S}}\left(\sum_{u\in\mathcal{U}} \mathbf{Y}_u(\psi)\right)\right]$ the criterion we wish to optimize. The indicator function is discontinuous on the border of $\mathcal{S}$. It prevents us from directly using a stochastic gradient method (Shapiro, Dentcheva, and Ruszczynski 2021). The next lemma [2] provides an explicit expression for the gradient of the criteria.

**Lemma 1.** *The gradient of $\mathcal{C}$ at $\psi$ satisfies*

$$[\nabla\mathcal{C}(\psi)]_{g,k} = \mathbb{E}\Big[\mathbb{I}_{\mathcal{S}}(\mathbf{Y})\Big((\mathbf{Y}-\boldsymbol{\mu}(\psi))^\intercal\boldsymbol{\Sigma}(\psi)^{-1}\cdot\boldsymbol{\mu}_{g,k} - \frac{1}{2}\big(\boldsymbol{\Sigma}(\psi) - (\mathbf{Y}-\boldsymbol{\mu}(\psi))(\mathbf{Y}-\boldsymbol{\mu}(\psi))^\intercal\big)\cdot\boldsymbol{\Sigma}(\psi)^{-1}\boldsymbol{\Sigma}_{g,k}\boldsymbol{\Sigma}(\psi)^{-1}\Big)\Big].$$

[2] The proof of Lemma 1 uses classical arguments from the policy learning literature (Williams 1992; Sutton and Barto 2018) and further relies on the chain rule with a few relations for multivariate Gaussian variables. We defer the proof to the Supplementary.

### 3.4 Optimization

Here, we present our optimization algorithm `SuccessProbaMax` (Algorithm 1) to solve (3) which takes as input

**(a) success region** $\mathcal{S} \subset \mathcal{Y}$ to define a criteria $\mathcal{C}$ introduced in (3). We typically consider success regions relative to a reference outcome $(y_0^v, y_0^c)$: it might be defined as all outcomes corresponding to increased value and decreased cost with respect to the reference value and cost;

**(b) estimated mean and variance values** $\{\hat{\boldsymbol{\mu}}_{g,k}\}$ and $\{\hat{\boldsymbol{\Sigma}}_{g,k}\}$ for all pairwise couples of groups $g$ and candidate policies $k$. There is a particular case when the exact values of mean and variances are known and do not require estimation;

**(c) some hyperparameters** such as an initial policy allocation function $\psi_0 \in \Delta^{\mathcal{G}}$, number of steps $n_{st}$ and learning rate $\eta$.

**Input**: $\mathcal{S}$, $\{\hat{\boldsymbol{\mu}}_{g,k}\}$, $\{\hat{\boldsymbol{\Sigma}}_{g,k}\}$, $\psi_0$, $n_{\text{st}} > 0$, $\eta > 0$
$\psi \leftarrow \psi_0$
**for** $t = 0$ *to* $n_{st}$ **do**
  $\hat{\boldsymbol{\mu}} \leftarrow \sum_{k,g} \psi(g,k)\hat{\boldsymbol{\mu}}_{g,k}$, $\hat{\boldsymbol{\Sigma}} \leftarrow \sum_{k,g} \psi(g,k)\hat{\boldsymbol{\Sigma}}_{g,k}$
  $\nabla \leftarrow \hat{\nabla}\mathcal{C}(\psi)$
  $\psi \leftarrow \psi + \eta\nabla$
  Project $\psi$ onto $\Delta^M$
**end**
**Return** $\psi$

**Algorithm 1:** `SuccessProbaMax`

The algorithm performs a gradient ascent $\nabla \leftarrow \hat{\nabla}\mathcal{C}(\psi)$ which can be computed using the formula from Lemma 1 and a numerical integration method for computing the expectation $\mathbb{E}_\psi$, e.g. a Monte-Carlo approach. The updated gradient is, then, projected onto the space of metapolicies $\Delta^{\mathcal{G}}$ to produce a solution candidate for (3) using a method from (Duchi et al. 2008). We provide several possible improvements of Algorithm 1 in Supplementary.

**Remark** As the computation of the gradient through the closed-form expression requires a matrix inversion, it is not always the best option computationally.

This is the case for the success region proposed in subsection 2.2, for which we observe that the criterion rewrites

$$\mathbb{E}\left[\mathbb{I}_{\mathcal{S}}\left(\sum_{u\in\mathcal{U}} \mathbf{Y}_u(\psi)\right)\right] = \text{cdf}_{Y^c}(y_0^c) - \text{cdf}_{\mathbf{Y}}(\mathbf{y}_0),$$

where $\text{cdf}_{Y_c}(y_0^c)$ is a (univariate) c.d.f. of $Y_c$ in $y_0^c$ and $\text{cdf}_{\mathbf{Y}}(\mathbf{y}_0)$ is a (bivariate) c.d.f. of $\mathbf{Y}$ in $\mathbf{y}_0$. (see appendix for the definition of bivariate c.d.f.). To speed up the algorithm, we rely on an approximation of the bivariate c.d.f. based on the error function (Tsay, Ke et al. 2011) to estimate $\text{cdf}_{\mathbf{Y}}(\mathbf{y}_0)$, and then implement it in JAX – this way we can directly use the automatic differentiation in JAX to numerically approximate the gradient of $\text{cdf}_{Y^c}(y_0^c) - \text{cdf}_{\mathbf{Y}}(\mathbf{y}_0)$.

## 4 Experimental Results

For all experiments below we use JAX framework (Bradbury et al. 2018) for the numerical estimation of the criterion's gradient, utilizing automatic differentiation within JAX instead of explicitly calculating the gradient and integrating it over an outcome. Hyperparameters used for the methods are provided in Supplementary material and source code[3] is published to reproduce all the empirical results.

### 4.1 Datasets

Besides the synthetic setups, which will be described below, we test algorithm on two large-scale, real world datasets.

- **CRITEO-UPLIFT v2** (Diemert et al. 2021) is provided by the AdTech company Criteo. Data contains 13.9 million samples which are collected from several incremental A/B tests. It includes 12 features, 1 binary treatment and 2 binary outcome labels ("visit" and "conversion"). Following (Zhou et al. 2023), we use "visit" label as proxy of the cost and "conversion" as the value. For the buckets, we used quantile bins of the "f0" feature. Finally, we randomly partitioned dataset into two equal parts for train and test. Preprocessing details are in Supplementary.
- **Private dataset** is constructed from a large-scale real-time bidding RCT. One feature was chosen based on an expert knowledge, buckets were created then as quantile-based projections of the feature. Dataset is aggregated over 70 days and consists of 9 buckets, 3 bidding policies (with reference) and 100 bootstraps of values and associated costs for each pair (bucket, policy).

  Remaining details along with aggregated datasets for one- and two-dimensional outcome cases are available in Supplementary material.

### 4.2 One-dimensional outcome

Here we assume an outcome $\mathcal{Y} \in \mathbb{R}$. Problem is parameterized by a difficulty level $r$ so that $\mathcal{S} = \{(r, +\infty)\}$. We present here results for synthetic data. Private data results are in Supplementary.

**Baselines** `SuccessProbaMax` is compared to several baselines searching for the optimal policy allocation:

- `Bruteforce`$(\{\boldsymbol{\mu}_{g,k}\}, \{\boldsymbol{\Sigma}_{g,k}\}, \mathcal{S})$ method that compares all possible *hard* allocations and for a given difficulty level returns allocation that maximises criterion;
- `Greedy1D`$(\{\boldsymbol{\mu}_{g,k}\})$ algorithm that returns the policy with the maximum mean value per bucket.

**Synthetic data generation** We generate Gaussian distributions for two cases: (i) "large variance" and (ii) "small variance", the same setup but the relative difference between the variances is much smaller – we expect the latter problem be harder than the former for the algorithms that take into account the variance. See Table 1 for precise parameters of distributions (data construction details and illustration of policy distributions per bucket are provided in Supplementary).

[3] https://github.com/criteo-research/success-proba-max

Table 1: Gaussian distribution parameters for synthetic data generation with three buckets ($M = 3$), three policies ($K = 3$) and one outcome ($Y \in \mathbb{R}$).

| Example | $[\mu_{g,k}]$ | $[\Sigma_{g,k}]$ | |
|---|---|---|---|
| Large variance | $\begin{bmatrix} 2 & 1.9 & 0 \\ 2 & 1 & 0 \\ 2 & 1 & 0 \end{bmatrix}$ | $\begin{bmatrix} 9 & 1 & 9 \\ 9 & 1 & 9 \\ 1 & 1 & 1 \end{bmatrix}$ | |
| Small variance | $\begin{bmatrix} 2 & 1.9 & 0 \\ 2 & 1 & 0 \\ 2 & 1 & 0 \end{bmatrix}$ | $\begin{bmatrix} 9 & 1 & 9 \\ 9 & 1 & 9 \\ 1 & 1 & 1 \end{bmatrix}$ | $\cdot 0.01$ |

**Results** We firstly fix $\mu_{g,k}$, $\Sigma_{g,k}$ and use them directly in the algorithm to avoid a source of randomness arising from parameters estimation, we provide the results below. Then we generate normal distributions with parameters $\mu_{g,k}$, $\Sigma_{g,k}$ and use *estimations* $\hat{\mu}_{g,k}$, $\hat{\Sigma}_{g,k}$ in the algorithm – corresponding results are presented in Supplementary.

On Fig. 2 (left) we show how our method performs for easier problem with large variance with varying difficulty level $r$. `SuccessProbaMax` starts from uniform allocation $\psi_0^{unif}$ and performs same as `Bruteforce`. The key reasons why our algorithm beats `Greedy1D` is that we i) directly optimize metric of interest and that we ii) effectively incorporate variance into optimization, while `Greedy1D` only operates with means.

Fig. 2 (right) shows results for the small variance case. Firstly, note that the gain over `Greedy1D` (in the region $r \in [5, 5.5]$) is drastically smaller than in the previous case. Then, at difficulty level $r > 6$ the performance of our algorithm drops down. This is because criterion value $\mathcal{C}(\psi_0^{unif})$ becomes 0 and the gradient is not updated. To overcome the problem, we can either "warm-start" from the baseline policy (e.g. from `Greedy1D` one) or to explore, by estimating the criterion for several random initial allocations.

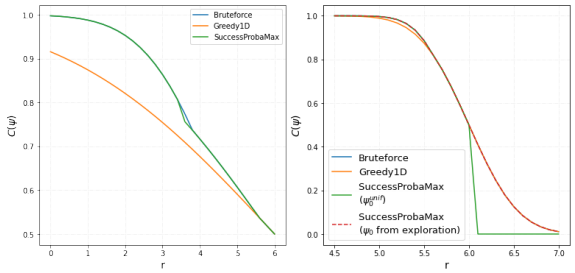


Figure 2: Results for different difficulty level $r$ on synthetic setup with one-dimensional outcome for large (left) and small (right) variance cases.

### 4.3 Two-dimensional outcome

In this case we consider an outcome $\mathcal{Y} \in \mathbb{R}^2$, so $\mathbf{Y} = (Y^v, Y^c)$. Problem is parameterized by two-dimensional difficulty levels $r$ so that $\mathcal{S} = \{(r_v, +\infty), (-\infty, r_c]\}$.

**Baselines** In addition to `Bruteforce`, `SuccessProbaMax` is compared with two other baselines that search for the optimal policy allocation:

- `LinProg`$(\{\boldsymbol{\mu}_{g,k}\}, r_c)$ algorithm (linear programming) that solves the fractional knapsack problem and returns a policy (soft allocation) with the maximum mean value per bucket;
- `MixedInt`$(\{\boldsymbol{\mu}_{g,k}\}, r_c)$ algorithm (mixed-integer linear programming) that solves the 0/1 knapsack problem and returns a policy (hard allocation) with the maximum mean value per bucket.

**Synthetic data generation** We generate bivariate Gaussian distributions for cases (i) and (ii), see Table 2 for precise parameters of distributions (an illustration of policy distributions is provided in Supplementary).

Table 2: Bivariate Gaussian distribution parameters for synthetic data generation with one bucket ($M = 1$), two policies ($K = 2$) and two-dimensional outcome ($\mathbf{Y} = (Y^v, Y^c) \in \mathbb{R}^2$).

| Example | $[\mu^v_{g,k}]$ | $[\Sigma^v_{g,k}]$ | $[\mu^c_{g,k}]$ | $[\Sigma^c_{g,k}]$ | $\rho$ |
|---|---|---|---|---|---|
| $\mu^c_2$ and $\Sigma^c_1$ larger | $[2, 1]$ | $[9, 1]$ | $[1, 1.5]$ | $[4, 1]$ | 0.5 |
| $\mu^c_2$ and $\Sigma^c_1$ smaller | $[2, 1]$ | $[9, 1]$ | $[1, 0.5]$ | $[1, 1]$ | 0.5 |

We firstly fix $\mu^v_{g,k}, \Sigma^v_{g,k}$ and $\mu^c_{g,k}, \Sigma^c_{g,k}$, and use them directly in the algorithm to avoid a source of randomness arising from parameters estimation (results for the case with parameters estimation are presented in Supplementary).

**Two-dimensional outcome: results.** For the experiment, we fix $r_v = 0$ and vary $r_c$ only. Fig. 3 shows that for both cases, `SuccessProbaMax` started from uniform allocation $\psi_0^{unif}$ reaches the same performance as `Bruteforce`.

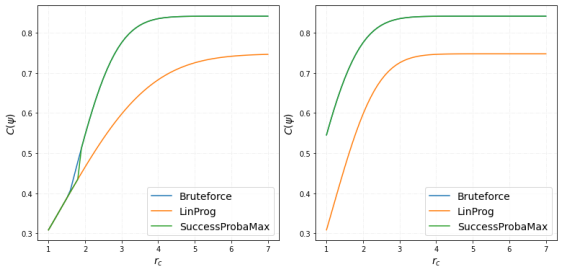


Figure 3: Result for different $r_c$ with fixed $r_v = 0$ on synthetic setup with two-dimensional outcome for cases (i) (left) and (ii) (right).

**Private dataset** In our first experiment, we fix $r_c = 0$ and vary $r_v$ only - so we check if we can increase total value while having same total cost as for reference. We then repeat computations, but now we fix $r_v = 0$ and vary $r_c$ only - in this case we wonder how often we can reach at least total value of the reference policy while changing total cost (this case is described in Supplementary).

**Results** Fig. 4 describes results on the private dataset with two-dimensional outcome for the range of Gain $r_v$ while $r_c = 0$ for train (left) and test (right) splits. Our algorithm, initialized with $\psi_0$ from exploration, reach the Gain of 0.01

in value (1% over the reference) with probability 0.7 for train and 0.4 for test, while for `MixedInt` respective probabilities are 0.35 and 0.1. Note that `Bruteforce` might not be the best here as some soft allocation may outperform hard ones for particular $(r_v, r_c)$.

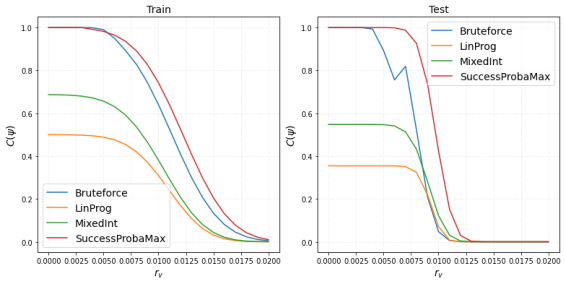


Figure 4: Results for different Gain $r_v$ while $r_c = 0$ on private dataset with one-dimensional outcome for train (left) and test (right) splits.

**CRITEO-UPLIFT v2** Data contains 2 policies including reference ("control"), so value can be increased only by increasing the cost. Thus, now we vary both $r_v$ and $r_c$ from 0 to 0.2, and a trade-off between value and cost is expected.

**Results** Fig. 5 depicts differences in $\mathcal{C}(\psi)$ between our algorithm and best baseline `Bruteforce` (absolute values are provided in Supplementary). Firstly, there is indeed a trade-off - for increasing cost by $x\%$ value increases by roughly $2x\%$. In addition, our algorithm reach higher $\mathcal{C}(\psi)$ in several regions (e.g. where $r_c \in [0.03, 0.04]$ and $r_v \in [0.04, 0.08]$ or where $r_c \in [0.08, 0.1]$ and $r_v \in [0.1, 0.16]$).

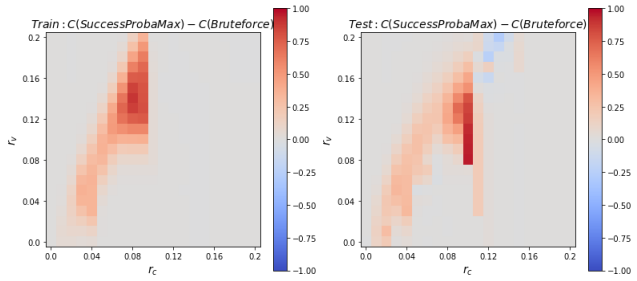


Figure 5: Results for different $r_v$ and $r_c$ on CRITEO-UPLIFT v2 with two-dimensional outcome for train (left) and test (right) splits.

The results correspond to the sketch provided in Fig. 1. It represents the distributions of the outcome $\mathbf{Y}$ for the respective allocations output by (i) `SuccessProbaMax` in orange and (ii) `Greedy` in blue. `SuccessProbaMax` outputs a solution for which the outcome has a very high probability to be in $\mathcal{S}_{\mathbf{y}_0}$ with $\mathbf{y}_0 = (y_0^v, y_0^c) = (r_v, r_c)$, even if generates a bit less value on average than `Greedy`, which presents a high risk of being outside of the success region (for example by breaking the cost constraint).

## 5 Related work

Some recent papers address multiple treatment allocation problem under budget constraints from different perspectives. The standard two-stage method firstly estimates treatment effects to predict value and cost for each user, then solves a knapsack problem (Ai et al. 2022; Albert and Goldenberg 2022; Tu et al. 2021; Zhao et al. 2019). Nevertheless, the goals of two-stage approaches and real-world scenarios do not perfectly align. Yan et al. (2023) proposes a two-stage method with an addition regularizer to the knapsack problem loss to address a business goal. However, the regularizer requires a mathematically well-defined function (such as expected outcome metric) and its gradients estimation.

Applying the decision-focused framework for marketing problems under budget constraints, Du, Lee, and Ghaffarizadeh (2019) propose a rank method by comparing learned ratios between values and costs for the aggregated targeted treatment effect to improve user retention problem. However, Zhou et al. (2023) show that the suggested loss function cannot converge to a stable extreme point in theory and improve the framework. Authors limit the treatments to different levels of one treatment, e.g. different levels of discount of some products. Further, they develop an algorithm equivalent to the Lagrange dual method ('greedy' approach) but based on learning to rank decision factors for multiple choice knapsack problem solutions. In our current context, our focus is solely on the top-ranked action, rather than the complete ranking itself. Moreover, as we discussed earlier, the knapsack formulation remains a proxy to our problem, so finding efficiently the best decision factors is still not equivalent to finding the best solution to the final business goal.

The closest to our work, Tu et al. (2021) suggest to reformulate the treatment allocation problem as a stochastic optimization task assuming normally distributed outcomes of bucket-level objective and constraints, however, the final problem remains in the knapsack form.

## 6 Conclusion and future works

We suggested a new formulation of the policy allocation problem that is better adapted to some downstream tasks when the success region is clearly identified. Compared to greedy approaches, our algorithm directly optimizes metric of interest and effectively utilizes variance in the optimization, while greedy ones only operate with means. Moreover, the proposed method can be efficiently applied to improve the given baseline policy.

Further works include a theoretical analysis of the algorithm. In particular, how it behaves numerically when the dimension of the outcome increases. Also, it is important to understand the relationship between the means and variances of potential outcomes that makes the proposed method outperform the greedy approaches. In addition to several suggested improvements of Algorithm 1, promising direction should be to couple the choice of user partitioning and the policy allocation problem into one master problem. Last, given that outcomes on different user segments may correlate, adapting the framework for Bayesian learning seems a pragmatic avenue for further research.

## 7 Acknowledgments

We would like to thank David Rohde and Eustache Diemert for their feedback and ideas during the project.

## References

Ai, M.; Li, B.; Gong, H.; Yu, Q.; Xue, S.; Zhang, Y.; Zhang, Y.; and Jiang, P. 2022. LBCF: A Large-Scale Budget-Constrained Causal Forest Algorithm. In *ACM Web Conference*. 3, 7

Albert, J.; and Goldenberg, D. 2022. E-Commerce Promotions Personalization via Online Multiple-Choice Knapsack with Uplift Modeling. In *ACM International Conference on Information & Knowledge Management*. 7

Athey, S.; and Imbens, G. 2016. Recursive partitioning for heterogeneous causal effects. *National Academy of Sciences*, 113(27): 7353–7360. 3

Betlei, A.; Gregoir, T.; Rahier, T.; Bissuel, A.; Diemert, E.; and Amini, M.-R. 2021. Differentially Private Individual Treatment Effect Estimation from Aggregated Data. *PPML Workshop*. 3

Bompaire, M.; Désir, A.; and Heymann, B. 2021. Robust label attribution for real-time bidding. *arXiv preprint arXiv:2012.01767*. 1

Bompaire, M.; Gilotte, A.; and Heymann, B. 2021. Causal Models for Real Time Bidding with Repeated User Interactions. In *ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1

Bradbury, J.; Frostig, R.; Hawkins, P.; Johnson, M. J.; Leary, C.; Maclaurin, D.; Necula, G.; Paszke, A.; VanderPlas, J.; Wanderman-Milne, S.; and Zhang, Q. 2018. JAX: composable transformations of Python+NumPy programs. 5

Castiglioni, M.; Celli, A.; Marchesi, A.; Romano, G.; and Gatti, N. 2022. A Unifying Framework for Online Optimization with Long-Term Constraints. *arXiv preprint arXiv:2209.07454*. 1

Conitzer, V.; Kroer, C.; Sodomka, E.; and Stier-Moses, N. E. 2022. Multiplicative Pacing Equilibria in Auction Markets. *Operations Research*, 70(2): 963–989. 1

Dalessandro, B.; Perlich, C.; Stitelman, O.; and Provost, F. 2012. Causally Motivated Attribution for Online Advertising. In *International Workshop on Data Mining for Online Advertising and Internet Economy (AdKDD)*. 1

Demirović, E.; Stuckey, P. J.; Bailey, J.; Chan, J.; Leckie, C.; Ramamohanarao, K.; and Guns, T. 2019. An investigation into prediction+ optimisation for the knapsack problem. *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. 1

Diamond, S.; and Boyd, S. 2016. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83): 1–5. 13

Diemert, E.; Betlei, A.; Renaudin, C.; Amini, M.-R.; Gregoir, T.; and Rahier, T. 2021. A large scale benchmark for individual treatment effect prediction and uplift modeling. *arXiv preprint arXiv:2111.10106*. 5

Du, S.; Lee, J.; and Ghaffarizadeh, F. 2019. Improve User Retention with Causal Learning. In *ACM SIGKDD Workshop on Causal Discovery*. 7

Duchi, J.; Shalev-Shwartz, S.; Singer, Y.; and Chandra, T. 2008. Efficient projections onto the l1-ball for learning in high dimensions. In *International Conference on Machine Learning*. 5

Efron, B. 1979. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1): 1 – 26. 4

Fernández-Loría, C.; Provost, F.; Anderton, J.; Carterette, B.; and Chandar, P. 2022. A comparison of methods for treatment assignment with an application to playlist generation. *Information Systems Research*. 1

Ji, W.; and Wang, X. 2017. Additional Multi-Touch Attribution for Online Advertising. *AAAI Conference on Artificial Intelligence*. 1

Kleber, M. 2019. Turtledove. 3

Levine, S.; Kumar, A.; Tucker, G.; and Fu, J. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*. 10

Moriwaki, D.; Hayakawa, Y.; Matsui, A.; Saito, Y.; Munemasa, I.; and Shibata, M. 2021. A Real-World Implementation of Unbiased Lift-based Bidding System. In *IEEE International Conference on Big Data*. 2

Rubin, D. B. 1974. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, 66(5): 688. 2

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015. Trust region policy optimization. In *International conference on machine learning*, 1889–1897. PMLR. 10

Shapiro, A.; Dentcheva, D.; and Ruszczynski, A. 2021. *Lectures on stochastic programming: modeling and theory*. Society for Industrial and Applied Mathematics. 4

Sinha, P.; and Zoltners, A. A. 1979. The Multiple-Choice Knapsack Problem. *Operations Research*, 27(3): 503–515. 1, 3

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press. 4

Tsay, W.-J.; Ke, P.-H.; et al. 2011. A simple approximation for bivariate normal integral based on error function and its application on probit model with binary endogenous regressor. Technical report, Institute of Economics, Academia Sinica, Taipei, Taiwan. 5

Tu, Y.; Basu, K.; DiCiccio, C.; Bansal, R.; Nandy, P.; Jaikumar, P.; and Chatterjee, S. 2021. Personalized treatment selection using causal heterogeneity. In *ACM Web Conference*. 3, 7

Wager, S.; and Athey, S. 2018. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523): 1228–1242. 3

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement Learning*, 5–32. 4

Yan, Z.; Wang, S.; Zhou, G.; Lin, J.; and Jiang, P. 2023. An End-to-End Framework for Marketing Effectiveness Optimization under Budget Constraint. *arXiv preprint arXiv:2302.04477*. 7

Zhao, K.; Hua, J.; Yan, L.; Zhang, Q.; Xu, H.; and Yang, C. 2019. A unified framework for marketing budget allocation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1820–1830. 7

Zhou, H.; Li, S.; Jiang, G.; Zheng, J.; and Wang, D. 2023. Direct Heterogeneous Causal Learning for Resource Allocation Problems in Marketing. *AAAI Conference on Artificial Intelligence*. 1, 5, 7, 11

# A Discussion on different sources of variance

There are different sources of randomness:

- *estimation variability*: how close is the estimator $\hat{\mathbf{Y}}_{g,k}$ to the true value $\mathbf{Y}_{g,k}$ (when we do not have a direct access to $\mathbf{Y}_{g,k}$);
- *allocation variability*: when $\psi$ is a soft allocation, the allocation of a user $u \in g$ to policy $\in \mathcal{K}$ can be described by the categorical random variable $P^{\psi}(g) \in \mathcal{K}$ such that $\mathbb{P}\left(P^{\psi}(g) = k\right) = \psi(g,k)$. We call $\mathcal{P}$ the filtration containing all the randomness from the variables $P^{\psi}(g)$ for all $g$.
- *system stochasticity*: for a fixed $g$ and $k$, the quantity $\mathbf{Y}_{g,k}$ is itself a random variable which contains the randomness from the behaviour of users in $g$. We call $\mathcal{Y}$ the filtration containing all randomness from the variables $\mathbf{Y}_{g,k}$ for all $g,k$.

In our work, we do not consider the problem of estimation variability, but focus on the derivation of the variance of $\mathbf{Y}$ under a given soft allocation $\psi$. This variance $\mathrm{Var}[\mathbf{Y}(\psi)]$ can be decomposed according to the variance with respect to the two sources of stochasticity respectively contained in $\mathcal{Y}$ and $\mathcal{P}$.

$$\mathrm{Var}[\mathbf{Y}(\psi)] = \mathbb{E}_{\mathcal{P}}\left[\mathrm{Var}_{\mathcal{Y}}\left[\mathbf{Y}|\mathcal{P}\right]\right] + \mathrm{Var}_{\mathcal{P}}\left[\mathbb{E}_{\mathcal{Y}}\left[\mathbf{Y}|\mathcal{P}\right]\right] \quad (4)$$
$$= \mathbb{E}_{\mathcal{Y}}\left[\mathrm{Var}_{\mathcal{P}}\left[\mathbf{Y}|\mathcal{Y}\right]\right] + \mathrm{Var}_{\mathcal{Y}}\left[\mathbb{E}_{\mathcal{P}}\left[\mathbf{Y}|\mathcal{Y}\right]\right] \quad (5)$$

Depending on assumptions, we can have different approximations of $\mathrm{Var}[\mathbf{Y}(\psi)]$ as a linear or quadratic function of $\psi$. For example, consider the decomposition of the variance with respect to allocation randomness in (4):

$$\mathbb{E}_{\mathcal{P}}\left[\mathrm{Var}_{\mathcal{Y}}\left[\mathbf{Y}|\mathcal{P}\right]\right] = \sum_k \sum_g \psi(g,k)\boldsymbol{\Sigma}_{g,k},$$

$$\mathrm{Var}_{\mathcal{P}}\left[\mathbb{E}_{\mathcal{Y}}\left[\mathbf{Y}|\mathcal{P}\right]\right] = \sum_g \psi(g,k)\big(\boldsymbol{\mu}_{g,k} - \sum_k \psi(g,k)\boldsymbol{\mu}_{g,k}\big)^2.$$

If we assume that there is minimal allocation variability, then $\mathrm{Var}_{\mathcal{P}}\left[\mathbb{E}_{\mathcal{Y}}\left[\mathbf{Y}|\mathcal{P}\right]\right] << \mathbb{E}_{\mathcal{P}}\left[\mathrm{Var}_{\mathcal{Y}}\left[\mathbf{Y}|\mathcal{P}\right]\right]$ and the variance of $\mathbf{Y}$ depends linearly on $\psi$ since

$$\mathrm{Var}[\mathbf{Y}(\psi)] \approx \mathbb{E}_{\mathcal{P}}\left[\mathrm{Var}_{\mathcal{Y}}\left[\mathbf{Y}|\mathcal{P}\right]\right] = \sum_k \sum_g \psi(g,k)\boldsymbol{\Sigma}_{g,k}.$$

In practice, we assume that the allocation variability is small enough so that this variance approximation holds, i.e. we are capable of assigning a given ratio of the population to a given policy. Indeed, we observe that for all $g,k$ there are enough users in $g$ such that we have an (approximately) fixed proportion $\psi(g,k)$ of users from $g$ which are allocated to policy $k$.

# B Gradient derivation (proof of Lemma 1)

1. We have $\nabla\mathcal{C}(\psi) = \mathbb{E}\left(\mathbb{I}_{\mathcal{S}}(\mathbf{Y})\nabla_{\psi}\left(\ln \ell(\psi,\mathbf{Y})\right)\right)$.

   *Proof.*

   $$\begin{aligned}\nabla\mathcal{C}(\psi) &= \nabla\int_{\mathcal{S}} \mathbb{I}_{\mathcal{S}}(\mathbf{y})\ell(\psi,\mathbf{y})\mathrm{d}\mathbf{y}\\ &= \int_{\mathcal{S}} \mathbb{I}_{\mathcal{S}}(\mathbf{y})\nabla\ell(\psi,\mathbf{y})\mathrm{d}\mathbf{y}\\ &= \int_{\mathcal{S}} \mathbb{I}_{\mathcal{S}}(\mathbf{y})\frac{\nabla_{\psi}\ell(\psi,\mathbf{y})}{\ell(\psi,\mathbf{y})}\ell(\psi,\mathbf{y})\mathrm{d}\mathbf{y}\\ &= \int_{\mathcal{S}} \mathbb{I}_{\mathcal{S}}(\mathbf{y})\nabla_{\psi}\left(\ln \ell(\psi,\mathbf{y})\right)\ell(\psi,\mathbf{y})\mathrm{d}\mathbf{y}\\ &= \mathbb{E}\left(\mathbb{I}_{\mathcal{S}}(\mathbf{y})\nabla_{\psi}\left(\ln \ell(\psi,\mathbf{y})\right)\right)\end{aligned}$$

   □

   We need the derivatives of a Gaussian log-likelihood of with respect to its parameters.

2. Let $\mathrm{p}[\boldsymbol{\mu},\boldsymbol{\Sigma}]$ be the probability density function of $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$, then

   $$\nabla_{\boldsymbol{\Sigma}^{-1}}\left(\ln \mathrm{p}[\boldsymbol{\mu},\boldsymbol{\Sigma}](\mathbf{y})\right) = \frac{1}{2}\boldsymbol{\Sigma} - \frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})(\mathbf{y}-\mu)^{\intercal},$$
   $$\nabla_{\boldsymbol{\mu}}\left(\ln \mathrm{p}[\boldsymbol{\mu},\boldsymbol{\Sigma}](\mathbf{y})\right) = (\mathbf{y}-\boldsymbol{\mu})^{\intercal}\boldsymbol{\Sigma}^{-1}.$$

   *Proof.* The proof is based on the rules of derivation by a vector and inversed matrix. □

3. If $f : t \to A(t)$ is an application from $\mathbb{R}$ to the set of non singular matrix of dimension $d$, then the derivative of $g : t \to A^{-1}(t)$ is $-gf'g$.
4. Let $\ell(\psi,\mathbf{y}) = \mathrm{p}[\boldsymbol{\mu}(\psi),\boldsymbol{\Sigma}(\psi)](\mathbf{y})$, then

   $$\begin{aligned}\left(\nabla_{\psi}\left(\ln \ell(\psi,\mathbf{y})\right)\right)_{g,k} &= (\mathbf{y}-\boldsymbol{\mu}(\psi))^{\intercal}\boldsymbol{\Sigma}(\psi)^{-1}\cdot\boldsymbol{\mu}_{g,k}\\ &\quad - \frac{1}{2}\left(\boldsymbol{\Sigma}(\psi) - (\mathbf{y}-\boldsymbol{\mu}(\psi))(\mathbf{y}-\boldsymbol{\mu}(\psi))^{T}\right)\\ &\quad \cdot\boldsymbol{\Sigma}(\psi)^{-1}\boldsymbol{\Sigma}_{g,k}\boldsymbol{\Sigma}(\psi)^{-1}.\end{aligned}$$

   *Proof.* Using the fact that $\boldsymbol{\mu}(\psi) = \sum_{g,k}\psi(g,k)\boldsymbol{\mu}_{g,k}$ and $\boldsymbol{\Sigma}(\psi) = \sum_{g,k}\psi(g,k)\boldsymbol{\Sigma}_{g,k}$, it is clear that

   $$\frac{\partial\boldsymbol{\mu}(\psi)}{\partial\psi(g,k)} = \boldsymbol{\mu}_{g,k},$$
   $$\frac{\partial\boldsymbol{\Sigma}(\psi)}{\partial\psi(g,k)} = \boldsymbol{\Sigma}_{g,k}.$$

Let $\ell(\psi, \mathbf{y}) = \mathrm{p}[\boldsymbol{\mu}(\psi), \boldsymbol{\Sigma}(\psi)](\mathbf{y})$ and index $(g,k) \in M \times K$, then, the chain rule and the previous steps lead to

$$\begin{aligned}\frac{\partial\left(\ln \ell(\psi, \mathbf{y})\right)}{\partial \psi(g,k)} &= \frac{\partial\left(\ln \mathrm{p}[\boldsymbol{\mu}(\psi), \boldsymbol{\Sigma}(\psi)](\mathbf{y})\right)}{\partial \psi(g,k)} \\ &= \frac{\partial\left(\ln \mathrm{p}[\boldsymbol{\mu}(\psi), \boldsymbol{\Sigma}(\psi)](\mathbf{y})\right)}{\partial \boldsymbol{\mu}(\psi)} \frac{\partial \boldsymbol{\mu}(\psi)}{\partial \psi(g,k)} \\ &\quad + \frac{\partial\left(\ln \mathrm{p}[\boldsymbol{\mu}(\psi), \boldsymbol{\Sigma}(\psi)](\mathbf{y})\right)}{\partial \boldsymbol{\Sigma}(\psi)^{-1}} \frac{\partial \boldsymbol{\Sigma}(\psi)^{-1}}{\partial \psi(g,k)},\end{aligned}$$

where

$$\begin{aligned}&\frac{\partial\left(\ln \mathrm{p}[\boldsymbol{\mu}(\psi), \boldsymbol{\Sigma}(\psi)](\mathbf{y})\right)}{\partial \boldsymbol{\mu}(\psi)} \frac{\partial \boldsymbol{\mu}(\psi)}{\partial \psi(g,k)} \\ &\quad = (\mathbf{y} - \boldsymbol{\mu}(\psi))^T \boldsymbol{\Sigma}(\psi)^{-1} \boldsymbol{\mu}_{g,k},\end{aligned}$$

$$\begin{aligned}&\frac{\partial\left(\ln \mathrm{p}[\boldsymbol{\mu}(\psi), \boldsymbol{\Sigma}(\psi)](\mathbf{y})\right)}{\partial \boldsymbol{\Sigma}(\psi)^{-1}} \frac{\partial \boldsymbol{\Sigma}(\psi)^{-1}}{\partial \psi(g,k)} = \\ &= \frac{1}{2}\left(\boldsymbol{\Sigma}(\psi) - (\mathbf{y} - \boldsymbol{\mu}(\psi))(\mathbf{y} - \boldsymbol{\mu}(\psi))^T\right) \\ &\quad \cdot \left(\boldsymbol{\Sigma}(\psi)^{-1} \boldsymbol{\Sigma}_{g,k} \boldsymbol{\Sigma}(\psi)^{-1}\right).\end{aligned}$$

□

5. The gradient of index $(g,k) \in M \times K$ is the following

$$\begin{aligned}[\nabla \mathcal{C}(\psi)]_{g,k} &= \mathcal{C}(\psi) \cdot \mathbb{E}_\psi\big((\mathbf{y} - \boldsymbol{\mu}(\psi))^\intercal \boldsymbol{\Sigma}(\psi)^{-1} \cdot \boldsymbol{\mu}_{g,k} \\ &\quad - \frac{1}{2}\left(\boldsymbol{\Sigma}(\psi) - (\mathbf{y} - \boldsymbol{\mu}(\psi))(\mathbf{y} - \boldsymbol{\mu}(\psi))^T\right) \\ &\quad \cdot \left(\boldsymbol{\Sigma}(\psi)^{-1} \boldsymbol{\Sigma}_{g,k} \boldsymbol{\Sigma}(\psi)^{-1}\right) | \mathbf{y} \in \mathcal{S}\big)\end{aligned}$$

Let $M = 1$ and $d = 1$, we get

$$\begin{aligned}\frac{\partial_k \mathcal{C}(\psi)}{\mathcal{C}(\psi)} &= \mathbb{E}_\psi\Bigg[\mathbb{I}_{\mathcal{S}}(y)\Bigg(\mu_k \frac{(y - \mu(\psi))}{\Sigma(\psi)} \\ &\quad - \frac{\Sigma_k}{2} \frac{\left(\Sigma(\psi) - (y - \mu(\psi))^2\right)}{\Sigma(\psi)^2}\Bigg)\Bigg].\end{aligned}$$

# C Improvements of `SuccessProbaMax`

We identify several directions of how `SuccessProbaMax` can be improved. Firstly, gradient step may be accelerated, either by i) using second-order methods like Newton method, ii) by applying line search to adapt step size. Secondly, we observe that algorithm may be stuck in the "flat" regions, e.g. if the criterion value of the initial policy allocation equals 0 – this problem often appears in policy gradient methods in reinforcement learning (Schulman et al. 2015; Levine et al. 2020). Currently, we explore several random allocations to begin optimization from (akin to epsilon-greedy exploration in reinforcement learning) or "warm-start" from baseline policy, but there are more options to avoid this behaviour, e.g. forcing exploration by regularization.

# D Example of alignment

Here, we provide an example of a problem when `SuccessProbaMax` and `Greedy` give the same solution.

Consider two policies $\pi_0$ and $\pi_1$ and users from $\mathcal{U}$ with a population of size $N$. Let potential outcome $Y_u(\pi_k)$ of user $u$ follow Bernoulli distributions $\mathcal{B}(p_k)$, where $k \in \{0,1\}$. Our goal is to maximize *success* $\mathbb{P}\left(\sum_u Y_u(\pi_{k_u}) = r\right)$, i.e. the probability of getting exactly $r$ successes in $N$ independent Bernoulli trials with parameters $p_0$ or $p_1$ depending on which policy $\pi_{k_u} \in \{\pi_0, \pi_1\}$ is assigned to users $u \in \mathcal{U}$.

If parameters $p_0$ and $p_1$ are not known, we need to estimate them from the data. Let $N_0$ users be assigned policy $\pi_0$ and $N_1 = N - N_0$ users be assigned policy $\pi_1$. For each policy, we observe $y(\pi_k) = \sum_{u=1}^{N_k} y_u(\pi_k)$, a realization of $Y(\pi_k) \sim \mathrm{Binom}(N_k, p_k)$, where $y_u(\pi_k)$ are sampled from $\mathcal{B}(p_k)$. We estimate the Bernoulli probability for each policy as $\hat{p}_k = \frac{1}{N_k} y(\pi_k)$. If we assume that there is no variance due to *estimation variability*, i.e. $|\hat{p}_k - p_k| \approx 0$, the total variance of $Y$ after observing $y = y(\pi_0) + y(\pi_1)$ is due to *system stochasticity* that consists of variance coming from the binomial distribution:

$$\begin{aligned}\mathrm{Var}[Y] &= \mathrm{Var}[Y(\pi_0)] + \mathrm{Var}[Y(\pi_1)] \\ &= N_0 \hat{p}_0 (1 - \hat{p}_0) + N_1 \hat{p}_1 (1 - \hat{p}_1)\end{aligned}$$

We notice that the variance of the total outcome $\mathrm{Var}[Y]$ is monotone with respect to $N_0$ and $N_1$.

In this example, `SuccessProbaMax` will search for a trade-off between the policy with the minimum estimated Bernoulli variance $\arg_k \min \hat{p}_k(1 - \hat{p}_k)$ and the maximum Bernoulli mean $\arg_k \max \hat{p}_k$. Since $\arg_k \min \hat{p}_k(1 - \hat{p}_k) = \arg_k \max \hat{p}_k$, we obtain that, in this example, `SuccessProbaMax` returns the same solution as `Greedy`.

# E Datasets

Here we describe two real datasets used in the paper.

## E.1 Private Dataset

Data was created from a large-scale real-time bidding randomized control trial (RCT) over 70 days and consists of 3 labels. The main label is the *value* - originally binary variable of some user action. For each value we collected an associated *cost*. We used *value* and *cost* for *two-dimensional* experiments. Along with this, we approximated a *revenue* as a function of value and cost, which will be used for *one-dimensional* experiments. Data consists of 3 randomly (respecting the RCT procedure) assigned policies, $\{\pi_0, \pi_1, \pi_2\}$, where $\pi_0$ is a reference bidding policy used in production, $\pi_1$ and $\pi_2$ are candidate bidding policies. In order to separate the user-level feature space, one feature was chosen based on an expert knowledge, 9 buckets were created then as quantile-based projections of the feature.

We aggregated labels by summarising them across the triplets (day, bucket, policy). Along with the sums, we computed 100 bootstraps of the aggregated value, cost and revenue, that will be used for the mean and (co-)variance estimations.

In order to make a balanced yet realistic train/test split, we summed labels for odd (train) and even (test) calendar days, hence we got both train and test data aggregated over 35 days.

To maintain data confidentiality, we computed a relative difference of labels with respect to the reference policy – for each pair (bucket, policy) we subtracted a value of the reference policy from the original one and divided it by a total reference value (a sum over buckets), we did the same for the cost and revenue. Finally, we used resulted bootstraps to estimate $\hat{\boldsymbol{\mu}}_{g,k}$ and $\hat{\boldsymbol{\Sigma}}_{g,k}$.

### E.2 CRITEO-UPLIFT v2

Dataset is provided by the AdTech company Criteo. Data contains 13.9 million samples which are collected from several incremental A/B tests. It includes 12 features, 1 binary treatment and 2 binary outcome labels ("visit" and "conversion"). Following (Zhou et al. 2023), we use "visit" label as proxy of the cost and "conversion" as the value. We randomly partitioned the dataset into two equal parts for train and test. For the buckets, we used quantile bins of the "f0" feature resulting in 8 buckets.

## F One-Dimensional outcome

### F.1 Criterion

Here we assume an outcome $\mathcal{Y} \in \mathbb{R}$. The problem is parameterized by a difficulty level $r$ so that $\mathcal{S} = \{(r, +\infty)\}$. Criterion then is defined as maximizing the following probability

$$\mathbb{P}(Y(\psi) > r) = \mathbb{E}\left[\mathbb{I}_{\mathcal{S}}\left(\sum_{u \in \mathcal{U}} Y_u(\psi)\right)\right] = 1 - \mathrm{cdf}_{Y(\psi)}(r),$$

where

$$Y(\psi) = \sum_k \sum_g \psi(g,k) Y_{g,k} \sim \mathcal{N}(\mu(\psi), \Sigma(\psi)),$$
$$\mu(\psi) = \sum_k \sum_g \psi(g,k) \mu_{g,k},$$
$$\Sigma(\psi) = \sum_k \sum_g \psi(g,k) \Sigma_{g,k},$$

and $\mathrm{cdf}_{Y(\psi)}(r)$ is a cumulative distribution function (c.d.f.) of $Y(\psi)$ in $r$.

Instead of computing the gradient as an integration over $Y(\psi)$ to obtain the expected value (see Lemma 1), we use an automatic differentiation in JAX to numerically approximate the gradient of $1 - \mathrm{cdf}_{Y(\psi)}(r)$.

### F.2 Baselines: `Bruteforce`

This method compares all possible *hard* allocations and for a given difficulty level returns allocation that maximises criterion, thus, resulting in the complexity $O(K^M)$, where $K$ is a number of policies and $M$ is a number of buckets – for large $K$ and $M$, Bruteforce is not an option because of its non-polynomial complexity.

### F.3 Synthetic setup

We simulate a toy yet sufficient setup in order to illustrate typical situations in which our algorithm can make an advantage.

Specifically, we generate parameters of Gaussian distributions for the setting of $M = 3$ buckets and $K = 3$ policies. We consider two cases: "large variance" and "small variance". The difference is that for "small variance", we scale variances by $0.01$ factor. See Table 1 for precise parameters of distributions and Fig. 7 and 8 for an illustration of policy distributions per bucket. For better describing the intuition, let us focus on the first bucket in both cases (first plots of the Figure 7 and 8 respectively).

The "large variance" case represents the situation when $\mu_{1,0} = 2, \mu_{1,1} = 1.9, \Sigma_{1,0} = 9, \Sigma_{1,1} = 1$, so the difference in means $\mu_{1,0} - \mu_{1,1}$ is much smaller than the difference between variances $\Sigma_{1,0} - \Sigma_{1,1}$. If we assume now $r = 0$, it becomes clear that policy $\pi_1$ is one that maximizes the success probability, however the "greedy" approach will choose $\pi_0$ because of the highest mean.

In the "small variance" case, the relative difference between the variances is now much smaller, meaning that an effective range of $r$, where our algorithm can outperform "greedy", drastically decreases.

To illustrate this, we plot in Fig. 6 the difference of criterion values

$$\mathcal{C}(\pi_1) - \mathcal{C}(\pi_0) = \mathbb{P}(Y(\pi_1) > r) - \mathbb{P}(Y(\pi_0) > r)$$

for a range of $r$. We define $r_{max}$ as a point where $\mathrm{cdf}_{\pi_0}(\mathrm{r_{max}}) = \mathrm{cdf}_{\pi_1}(\mathrm{r_{max}})$. We can see that

$$\mathcal{C}(\pi_1) - \mathcal{C}(\pi_0) \geq 0, r \in [-\infty; r_{max}].$$

The intuition is that while $r$ grows, the left tail of the $\pi_0$ distribution gets outside of the $\mathcal{S}$, then, we reach a point $r_{max}$, where two criterion values are equal. Finally,

$$\mathcal{C}(\pi_1) - \mathcal{C}(\pi_0) \leq 0, \forall r > r_{max}$$

due to a bigger variance of the $\pi_0$ distribution.

Comparing between the "large" and "small" variance cases, one can clearly see i) a gap in the potential "winning region" size and ii) a difference in the maximum value of $\mathcal{C}(\pi_1) - \mathcal{C}(\pi_0)$.

### F.4 Synthetic setup results

We use $\mu_{0,k} = [2, 1.9, 0], \Sigma_{0,k} = [9, 1, 9], r = 0$ to show the convergence of our algorithm on Fig. 9. Note that `SuccessProbaMax` found an optimal allocation $[0, 1, 0]$, which differs from the one of `Greedy1D`, $[1, 0, 0]$.

To check the algorithm performance where a source of randomness arising from the parameters estimation is presented, we generate normal distributions with parameters $\mu_{g,k}$, $\Sigma_{g,k}$ of sizes $N \in \{1000, 10000\}$ and use *estimations* $\hat{\mu}_{g,k}$, $\hat{\Sigma}_{g,k}$ in the algorithm.

To test the noise coming from the parameters estimation, for both "large" and "small" variance cases we randomly split generated data into train/test parts, estimate $\hat{\mu}_{g,k}$, $\hat{\Sigma}_{g,k}$ on train, and check resulted allocations on both train and

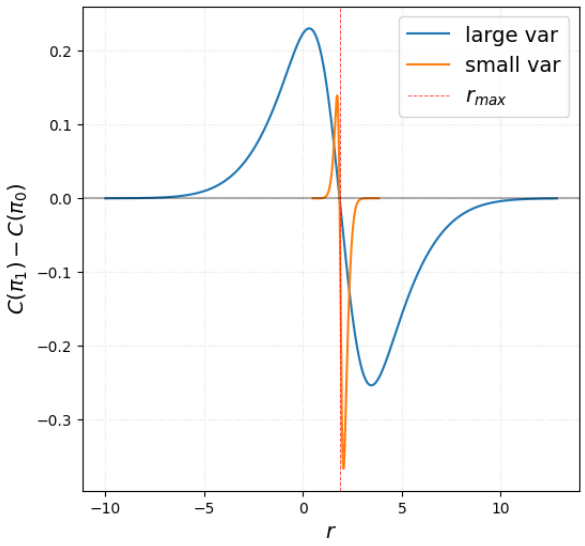


Figure 6: The difference of criterion values for $\pi_1$ and $\pi_0$.

test. We repeat the procedure 100 times to build proper confidence intervals. The results for the "large variance" case are presented in Figures 10 (for $N = 1000$) and 11 (for $N = 10000$).

The performance on train and test splits are very similar – it is reasonable as splits contain data from the same distribution. As one can see, the precision of $\hat{\mu}_{g,k}, \hat{\Sigma}_{g,k}$ estimation is higher for the $N = 10000$ case which is reflected in smaller confidence intervals for each method.

The results for the "small variance" case are provided in Figures 12 (for $N = 1000$) and 13 (for $N = 10000$). Due to the smaller original variance, confidence intervals are even smaller than in the previous case.

### F.5 Private dataset

For the real data cases, it is reasonable to have a direct interpretation of difficulty level $r$ for the RCT success probability. Thus, here we interpret $r$ as a relative gain in the outcome over the reference policy (or simply "Gain" hereafter) that we want to reach.

### F.6 Private dataset results

Fig. 14 describes results on the private dataset with one-dimensional outcome for the range of gains $r$ for train (left) and test (right) splits. Notice that the only possible region to improve $\mathcal{C}(\psi)$ in both cases is $[0.02, 0.03]$. For instance, like `Bruteforce`, our algorithm initialized with $\psi_0$ from exploration reaches the Gain of 0.029 (2.9% over the reference) with the probability almost 1, while for `Greedy1D` it is around 0.7.

## G Two-Dimensional outcome

### G.1 Criterion

In this case we consider an outcome $\mathcal{Y} \in \mathbb{R}^2$, so $\mathbf{Y} = (Y^v, Y^c)$. The problem is parameterized by a two-

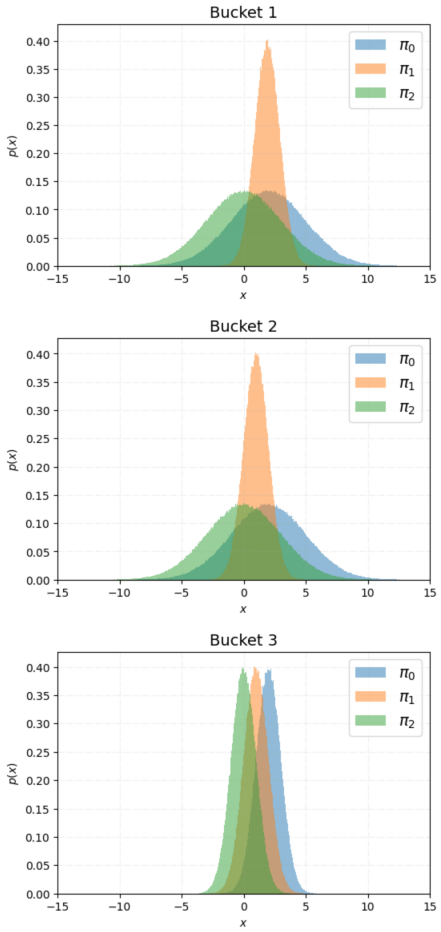


Figure 7: Synthetic data distributions for large variance case, with parameters described in Table 1.

dimensional difficulty level $\mathbf{r} = (r_v, r_c)$ so that $\mathcal{S} = \{(r_v, +\infty), (-\infty, r_c]\}$. The criterion, then, is defined as maximizing the following probability

$$\mathbb{E}\left[\mathbb{I}_{\mathcal{S}}\left(\sum_{u\in\mathcal{U}} \mathbf{Y}_u(\psi)\right)\right] = \mathrm{cdf}_{Y^c}(r_c) - \mathrm{cdf}_{\mathbf{Y}}(\mathbf{r}),$$

where

$$\mathbf{Y}(\psi) \sim \mathcal{N}\left(\boldsymbol{\mu}(\psi), \boldsymbol{\Sigma}(\psi)\right),$$
$$\boldsymbol{\mu}(\psi) = \sum_k \sum_g \psi(g,k)\boldsymbol{\mu}_{g,k},$$
$$\boldsymbol{\Sigma}(\psi) = \mathrm{Var}\left[\sum_k \sum_g \psi(g,k)\mathbf{Y}_{g,k}\right]$$

$\mathrm{cdf}_{Y_c}(r_c)$ is the (univariate) c.d.f. of $Y_c$ at $r_c$ and $\mathrm{cdf}_{\mathbf{Y}}(\mathbf{r})$ is the (bivariate) c.d.f. of $\mathbf{Y}$ at $\mathbf{r}$

$$\mathrm{cdf}_{\mathbf{Y}}(\mathbf{r}) = \int_{-\infty}^{r_v}\int_{-\infty}^{r_c} f(x_v, x_c)\, dx_v\, dx_c \qquad (6)$$

where $f(x_v, x_c)$ is the p.d.f. of bivariate normal distribution.

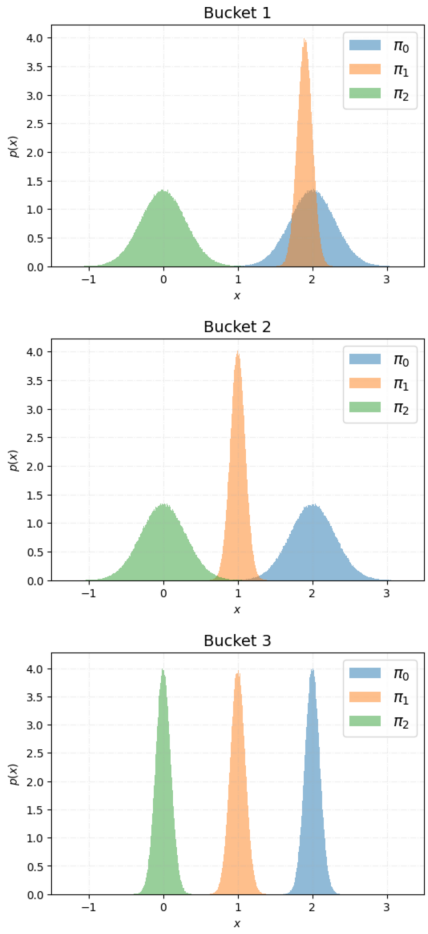


Figure 8: Synthetic data distributions for small variance case, with parameters described in Table 1.

## G.2 Baselines: `LinProg` and `MixedInt`

The `LinProg` algorithm (linear programming) solves the fractional knapsack problem (soft allocation) and returns a policy with the maximum mean value per bucket under the defined cost constraint. Along with the linear programming approach, we also implement `MixedInt` (mixed-integer linear programming) that solves the 0/1 knapsack problem and returns a hard allocation.

The main drawback of both algorithms for the success probability maximization problem is that only means of value and cost are used for optimization, while lacking information about the variance.

For both methods, CVXPY Python library (Diamond and Boyd 2016) was used for the implementation.

## G.3 Synthetic setup

We generate parameters of bivariate Gaussian distributions for the setting of $M = 2$ buckets and $K = 3$ policies. We consider two specific examples, see Table 2 for precise parameters of distributions and Fig. 15 for an illustration of policy distributions.

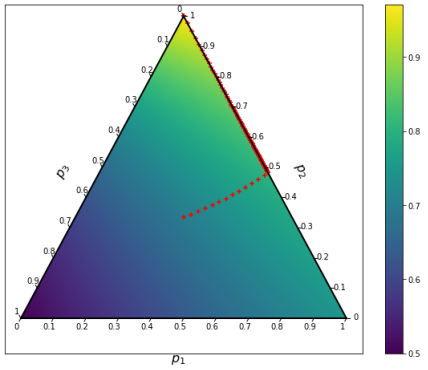


Figure 9: Convergence of `SuccessProbaMax` on the toy example. We observe that the optimal decision is different from the output of the greedy algorithm.

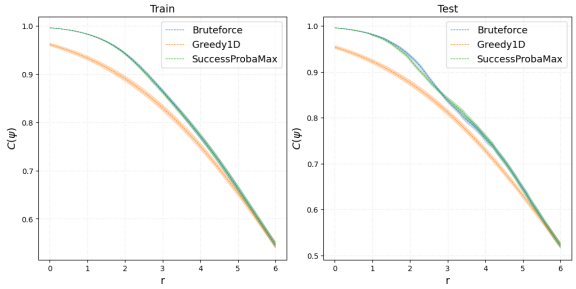


Figure 10: Synthetic setup with one-dimensional outcome, $N = 1000$, "large variance" case: results for a range of difficulty level $r$ on 100 random train (left) and test (right) data splits.

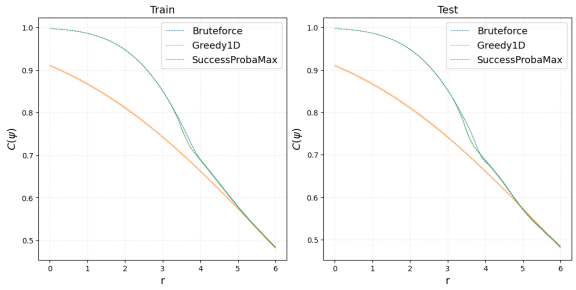


Figure 11: Synthetic setup with one-dimensional outcome, $N = 10000$, "large variance" case: results for a range of difficulty level $r$ on 100 random train (left) and test (right) data splits.

In general, we keep the idea from the one-dimensional setup and create two examples. In example (i), $\pi_1$ has a bigger mean cost and a larger mean value, however, both variances are smaller than for $\pi_0$. For $\mathbf{r} = (r_v, r_c) = (0, 3)$, an optimal policy is $\pi_1$, however, `LinProg` again will choose $\pi_0$ due to the larger $\mu^v_{1,0}$ at the cost constraint $\mu^c_{1,0}$.

In example (ii), there is a positive difference in means

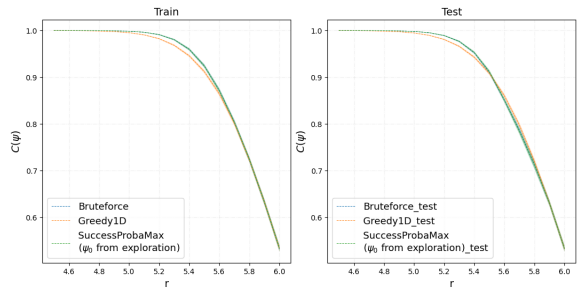


Figure 12: Synthetic setup with one-dimensional outcome, $N = 1000$, "small variance" case: results for a range of difficulty level $r$ on 100 random train (left) and test (right) data splits.

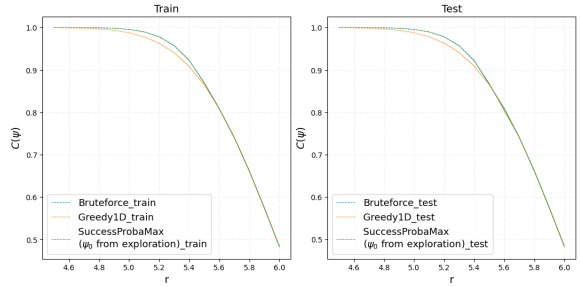


Figure 13: Synthetic setup with one-dimensional outcome, $N = 10000$, "small variance" case: results for a range of difficulty level $r$ on 100 random train (left) and test (right) data splits.

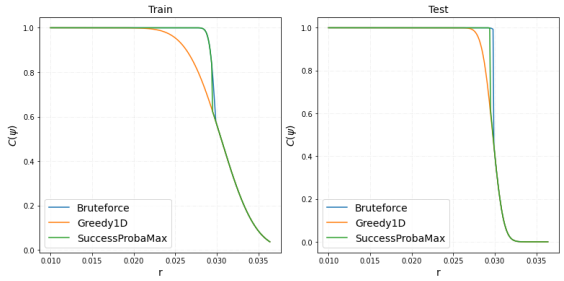


Figure 14: Results for different Gain $r$ on private data with one-dimensional outcome for train (left) and test (right) splits.

$\mu^v_{1,0} - \mu^v_{1,1}$ but it is much smaller than the difference between variances $\Sigma^v_{1,0} - \Sigma^v_{1,1}$. At the same time, $\mu^c_{1,1}$ is smaller than $\mu^c_{1,0}$ and $\Sigma^c_{1,0} = \Sigma^c_{1,1}$. If we fix $\mathbf{r} = (r_v, r_c) = (0, 1)$, an optimal policy is $\pi_1$, however `LinProg` will choose $\pi_0$ due to the larger $\mu^v_{1,0}$ at the cost constraint $\mu^c_{1,0}$.

## G.4 Synthetic setup results

To check the algorithm performance where estimation variability is present, we repeat the same procedure as for the one-dimensional setup, generating bivariate normal distributions of size $N = 1000$ with defined parameters.

The results for cases (i) and (ii) are presented in Figures 16 and 17 respectively.

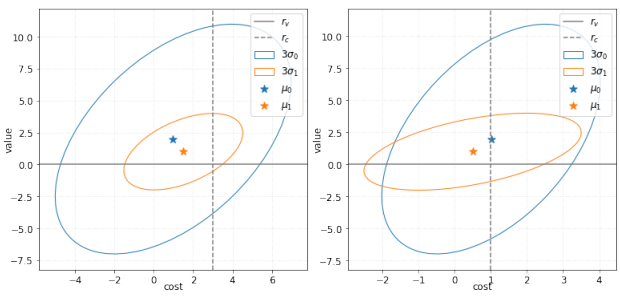


Figure 15: Confidence ellipses of the synthetic data distributions for case (i) (left) and (ii) (right) with parameters described in Table 2.

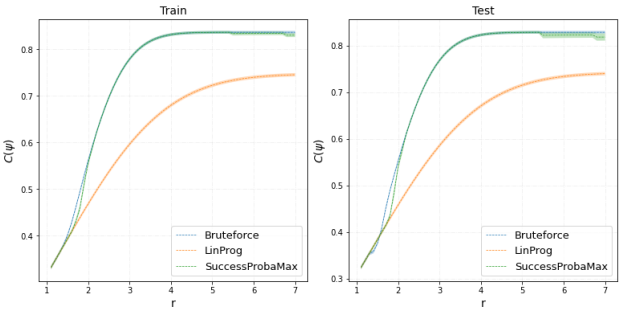


Figure 16: Synthetic setup with two-dimensional outcome, $N = 1000$, case (i): results for a range of $r_c$ on 100 random train (left) and test (right) data splits.

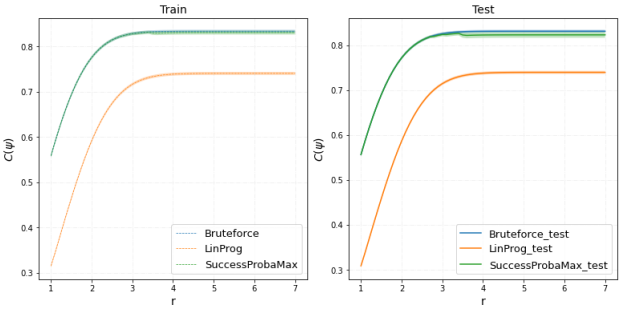


Figure 17: Synthetic setup with two-dimensional outcome, $N = 1000$, case (ii): results for a range of $r_c$ on 100 random train (left) and test (right) data splits.

## G.5 Private data results

Fig. 18 describes results on the private dataset with two-dimensional outcome for the a range of gains $r_c$ while $r_v = 0$ for train (left) and test (right) splits. For instance,

SuccessProbaMax reaches the Gain of -0.02 in cost (-2% over the reference) with probability 0.97 for train and 1 for test, while for MixedInt respective probabilities are 0.86 and 0.89.

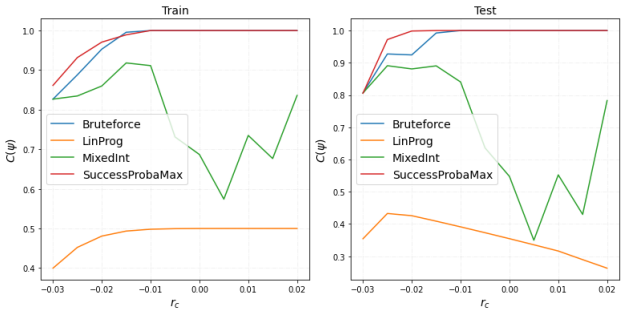


Figure 18: Results for different Gain $r_c$ while $r_v = 0$ on the private dataset with one-dimensional outcome for train (left) and test (right) splits.

### G.6 CRITEO-UPLIFT v2 results

Absolute criterion values for train and test data splits are presented for Bruteforce, LinProg, MixedInt and SuccessProbaMax on Figures 19, 20, 21 and 22 respectively. As we can see, among the other methods our algorithm is the most efficient and stable at the same time.

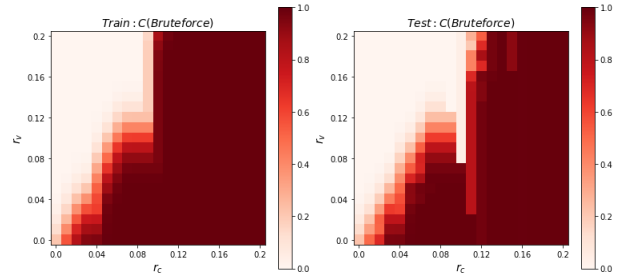


Figure 19: Absolute criterion values of Bruteforce method on CRITEO-UPLIFT v2 data on train (left) and test (right) data splits.

## H Hyperparameters

There are no hyperparameters for the baselines. SuccessProbaMax includes three hyperparameters - initial allocation $\psi_0$, learning rate $\eta$ and number of steps $n_{st}$.

For $\psi_0$, we define three options:

$$\psi_0 \in \{\psi_0^{unif}, \psi_0^{baseline}, \psi_0^{expl}\},$$

where $\psi_0^{unif}$ represents a uniform allocation, $\psi_0^{baseline}$ is a baseline allocation (from Greedy1D in 1D case and LinProg in 2D), and $\psi_0^{expl}$ is an allocation from exploration, when we generate 50000 random allocations and pick one with the maximum criterion value.

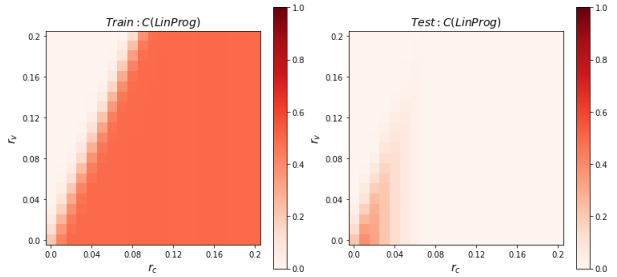


Figure 20: Absolute criterion values of LinProg method on CRITEO-UPLIFT v2 data on train (left) and test (right) data splits.

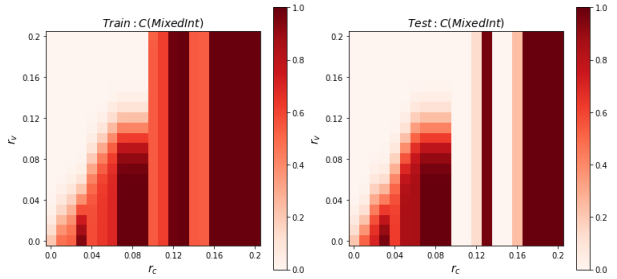


Figure 21: Absolute criterion values of MixedInt method on CRITEO-UPLIFT v2 data on train (left) and test (right) data splits.

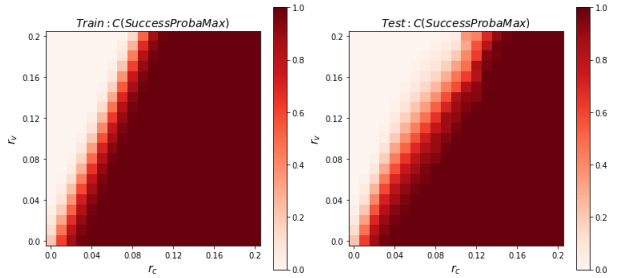


Figure 22: Absolute criterion values of SuccessProbaMax method on CRITEO-UPLIFT v2 data on train (left) and test (right) data splits.

We consider the following possible sets of $\eta$ and $n_{st}$:

$$\eta \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\},$$

$$n_{st} \in \{10^4, 10^5, 10^6, 5 \cdot 10^6\}$$

For each experiment, we did a grid search over the hyperparameters set aiming to maximize $\mathcal{C}(\psi)$. The resulted hyperparameters for each experiment are presented in Table 3.

## I Hardware

Experiments were performed on Linux machine with 8 CPUs (Intel(R) Xeon(R) Silver 4108 CPU @ 1.80GHz) and 16Gb of RAM.

Table 3: Hyperparameters of `SuccessProbaMax`.

| Case | $\psi_0$ | $\eta$ | $n_{st}$ |
|---|---|---|---|
| 1D - synthetic - large var | $\psi_0^{unif}$ | $10^{-1}$ | $10^4$ |
| 1D - synthetic - small var | $\psi_0^{expl}$ | $10^{-1}$ | $10^4$ |
| 1D - private | $\psi_0^{expl}$ | $10^{-2}$ | $10^5$ |
| 2D - synthetic - case (i) | $\psi_0^{unif}$ | $10^{-2}$ | $10^4$ |
| 2D - synthetic - case (ii) | $\psi_0^{unif}$ | $10^{-2}$ | $10^4$ |
| 2D - private | $\psi_0^{expl}$ | $10^{-4}$ | $5 \cdot 10^6$ |
| 2D - Criteo | $\psi_0^{expl}$ | $10^{-3}$ | $10^6$ |