7 Supplemental Material to Empirical Mechanism Design: Designing Mechanisms from Data

In this document, we present supplemental material to paper *Empirical Mechanism Design: Designing Mechanisms from Data*. The paper is located at `http://auai.org/uai2019/proceedings/papers/406.pdf`. The code is located at `https://github.com/eareyan/emd-adx`.

Game Elements

An advertising **campaign** $C = \langle I, M, R \rangle$ demands $I \in \mathbb{N}$ impressions in total, procured from users belonging to any market segment M' that **matches** the campaign's desired market segment M. (The match function, and market segments, are defined precisely below.) A campaign's budget $R \in \mathbb{R}_+$ is the maximum amount the advertiser is willing to spend on those impressions. The **one-shot AdX game** is played by a set A of agents, where each $j \in A$ is endowed with a campaign $C_j = \langle I_j, M_j, R_j \rangle$.

The AdX game is a game of incomplete information. In general, agents do not know one another's campaigns. Consequently, agent j's strategy set consists of all functions mapping agent j's campaign C_j (i.e., its private information) to tuples of the form $\left\langle \boldsymbol{b}^j, \boldsymbol{l}^j \right\rangle$. The first component of this tuple is a bid vector $\boldsymbol{b}^j = \left\langle b_1^j, b_2^j, \cdots, b_{|\mathcal{M}|}^j \right\rangle$, where $b_M^j \in \mathbb{R}_+$ is agent j's bid for impressions that match market segment M. The second component is a limit vector $\boldsymbol{l}_j = \left\langle l_1^j, l_2^j, \cdots, l_{|\mathcal{M}|}^j \right\rangle$, where $l_M^j \in \mathbb{R}_+$ is agent j's spending limit on impressions that match M.

We now proceed to define utilities. We start by more precisely defining what it means for a user's impression to match an advertiser's campaign. Let $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_k\}$ be an ordered set of k **attributes**: e.g., $\mathcal{A} = \{\text{GENDER}, \text{INCOME}, \text{AGE}\}$. Each attribute is defined as a discrete set of **attribute values**: e.g.,

1. $\text{GENDER} = \{\text{MALE}, \text{FEMALE}, \text{NON-BINARY}\}$
2. $\text{INCOME} = \{\text{HIGH}, \text{LOW}\}$
3. $\text{AGE} = \{\text{OLD}, \text{YOUNG}\}$

A **market segment** M is defined as a vector of attribute values, $M = \langle a_1, \ldots, a_k \rangle$, where $a_i \in \mathcal{A}_i$. Continuing our example, there are $3 \cdot 2 \cdot 3 = 18$ possible market segments, one of them being, $M = \langle \text{NON-BINARY}, \text{LOW}, \text{YOUNG} \rangle$.

We augment each attribute $\mathcal{A}_i$ with a special symbol $*$ to create the **augmented attribute** $\mathcal{A}_i^* = \mathcal{A}_i \cup \{*\}$. An **augmented market segment** M^* is a vector of augmented attributes' values, $M^* = \langle a_1^*, \ldots, a_k^* \rangle$, where $a_i^* \in \mathcal{A}_i^*$. We denote by $\mathcal{M}$ the set of all possible augmented market segments.

Given two augmented market segments M and M', we say that M **matches** M', and write $M \preceq M'$, if and only if for $i = 1, \ldots, k$, if $a_i' \neq *$, then $a_i' = a_i$. In words, $M \preceq M'$ iff

1. In case the i-th attribute value of M' is the special symbol $*$, the i-th attribute value of M is any value.
2. In case the i-th attribute value of M' is not $*$, the i-th attribute value of M' is the same as the i-th attribute value of M.

For example, the following are matches:

1. $\langle \text{FEMALE}, \text{HIGH}, \text{YOUNG} \rangle \preceq \langle *, *, * \rangle$.
2. $\langle \text{FEMALE}, \text{HIGH}, \text{YOUNG} \rangle \preceq \langle *, \text{HIGH}, \text{YOUNG} \rangle$.
3. $\langle \text{FEMALE}, \text{LOW}, \text{YOUNG} \rangle \preceq \langle \text{FEMALE}, *, \text{YOUNG} \rangle$.

The following are not matches:

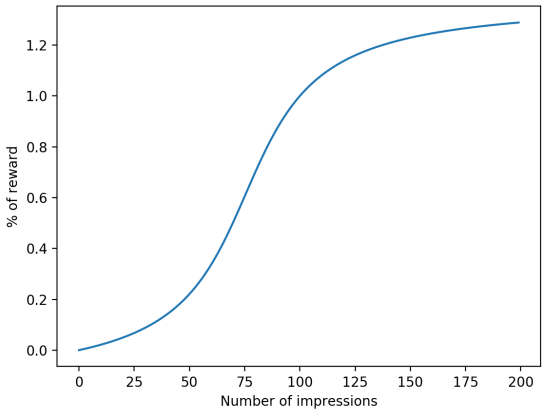

Figure 5: Sigmoidal for a campaign that demands 200 impressions.

1. $\langle$FEMALE, HIGH, YOUNG$\rangle \not\preceq \langle$MALE, *, *$\rangle$, since FEMALE $\neq$ MALE.

2. $\langle$NON-BINARY, HIGH, YOUNG$\rangle \not\preceq \langle$NON-BINARY, *, OLD$\rangle$, since YOUNG $\neq$ OLD.

The above definitions are intended to be used to describe which users' impressions match which advertisers' campaigns. Users belong to market segments (and thus, augmented market segments), and campaigns can be specified in the language of augmented market segments,[4] so these definitions are sufficient for matching impressions with campaigns.

(Hereafter, we drop the qualifier "augmented," as we only ever consider augmented markets segments.)

Denote by $\boldsymbol{y} = \langle y_1, y_2, \ldots, y_{|\mathcal{M}|} \rangle$ a bundle of impressions, where $y_M \in \mathbb{N}$ denotes the number of impressions from market segment M in bundle $\boldsymbol{y}$. The **utility** u_j of agent j, as a function of bundle $\boldsymbol{y}$, is given by:

$$u_j(\boldsymbol{y}, C_j) = \rho(\mu(\boldsymbol{y}, C_j), C_j) - p(\boldsymbol{y}) \tag{2}$$

Here $p(\boldsymbol{y})$ is the total **cost** of bundle $\boldsymbol{y}$, and $\mu(\boldsymbol{y}, C_j) = \sum_{M \in \mathcal{M}} y_M \mathbb{1}\{M \preceq M_j\}$ is a filtering function, which, given a bundle of impressions and a campaign, calculates the number of impressions in the bundle that match the campaign. Finally,

$$\rho(z, C_j) = \left(\frac{2R_j}{b}\right)\left(\arctan\left(\frac{bz}{I_j} - a\right) - \arctan(a)\right) \quad ,$$

where, for any nonzero $k \in \mathbb{R}$, $a = b + k$ and b is the unique solution to the equation $\dfrac{\arctan(k) - \arctan(-b)}{1+b} = \dfrac{1}{1+k^2}$. Following Schain and Mansour [18], we use $k = 1$, which implies $a \approx -3.08577$ and $b \approx 4.08577$. Intuitively, $\rho(z, C_j)$ maps a number of impressions z to a percentage of R_j in a sigmoidal fashion: i.e., small values of z yield a small percentage of R_j, while values close to I_j yield values close to R_j. The non-linearity inherent in this function models **complementarities**,[5] because it incentivizes agents to focus either on completely satisfying a campaign's demand, or not to bother satisfying it at all. Figure 5 depicts a sample sigmoidal, for a campaign that demands 200 impressions. Note that from an agent's point of view, its campaign's budget maps to its potential revenue.

This concludes our description of agents' types (i.e., their private information), their strategies, and their utilities in the one-shot AdX game.

[4] Augmented market segments are also a natural way to describe users, since not all user attributes are revealed, in general.

[5] A good is a complement of another if the value of acquiring both is strictly greater than the value of acquiring either one of them but not the other: e.g., the value of acquiring a pair of shoes (left and right) is usually greater than the value of only the left or the right one.

Game Dynamics

As the one-shot AdX game is a one-shot game, bids and spending limits are submitted to the ad exchange only once. Still, the game unfolds in four stages.

Stage 0: The agents learn their campaigns. *In the current experiments, we assume agents are well versed in their competition, and thus they also learn one another's campaigns.* As such, we model the agents playing a complete-, rather than an information-, information game.

Stage 1: The ad exchange announces a vector of reserve prices $\boldsymbol{r} \in \mathbb{R}_+^{\mathcal{M}}$, where $r_M \in \mathbb{R}_+$ is a price above which agents must bid to participate in auctions for impressions belonging to market segment $M \in \mathcal{M}$.

Stage 2: All agents submit their bids and their spending limits for all auctions: i.e., agent j submits tuple $\left\langle \boldsymbol{b}^j, \boldsymbol{l}^j \right\rangle$. As the name suggests, this limit is the *maximum* the agent is willing to spend on impressions that match M. In case there are multiple matches, the minimum among spending limits is enforced.

Stage 3: Some random number of impressions $K \sim U[K_l, K_u]$ arrive in a random order $\left\langle \vec{M} \right\rangle = \left\langle M^1, M^2, \ldots, M^K \right\rangle$, where $M^i \sim \boldsymbol{\pi} = \left\langle \pi_1, \pi_2, \ldots, \pi_{|\mathcal{M}|} \right\rangle$, and π_M is the probability of M. For each impression that arrives from market segment M, a second-price auction with reserve price $r_M \in \mathbb{R}_+$ is held among all agents whose bids are matched by M, and who have not yet reached their spending limit in M. If an agent submits multiple matching bids, it does not compete against itself; the maximum of its matching bids is entered into the auction.

Stage 4: The output of Stage 3 is, for each agent j, a bundle of impressions $\boldsymbol{y}^j = \left\langle y_1^j, y_2^j, \ldots, y_{|\mathcal{M}|}^j \right\rangle$, which is used to compute agent j's utility according to Equation (2).

The auctioneer's revenue is defined as $\sum_j p(\boldsymbol{y}^j)$.

Strategies

We devised two heuristic strategies, which we call Walrasian Equilibrium (WE) and Waterfall[6] (WF). At a high level, both heuristics work by collapsing the dynamics of the one-day AdX game into a static market model, which is then used to compute an allocation (assignment of impressions to campaigns) and prices (for impressions), based on which bids and limits are determined. In both cases, the static market is an augmented bipartite graph $\mathscr{M} = \langle M, C, E, \vec{N}, \vec{I}, \vec{R} \rangle$, with a set of $n \in \mathbb{N}$ market segments M; a set of $c \in \mathbb{N}$ campaigns C; a set of edges E from market segments to campaigns indicating which segments are of interest to which campaigns; supply vector $\vec{N} = \langle N_1, \ldots, N_n \rangle$, where $N_i \in \mathbb{N}$ is the number of available impressions from market segment $i \in M$; demand vector $\vec{I} = \langle I_1, \ldots, I_c \rangle$, where $I_j \in \mathbb{N}$ is the number of impressions demanded by campaign $j \in C$; and reward vector $\vec{R} = \langle R_1, \ldots, R_c \rangle$, where $R_j \in \mathbb{R}_+$ is campaign j's reward. We denote by $\boldsymbol{X} \in \mathbb{Z}_+^n \times \mathbb{Z}_+^c$ an allocation matrix, where entry x_{ij} is the allocation (i.e., number of impressions) from market segment i assigned to campaign j. We also denote by $\boldsymbol{p} \in \mathbb{R}_+^n$ a price vector, with price p_i per impression from market segment i; and by $\boldsymbol{P} \in \mathbb{R}_+^n \times \mathbb{R}_+^c$ a price matrix, with price p_{ij} per impression from market segment i to campaign j. The two heuristics differ in how the allocation and prices are computed.

Walrasian Equilibrium Strategy

In previous work [26], we noted that Walrasian equilibrium (WE) is not guaranteed to exist in the static market model of the one-day AdX game, assuming all-or-nothing valuations (step functions that approximate sigmoidals). We thus proposed a relaxation of WE, which can be computed in polynomial time using a two-step procedure: first, we solve for an approximate welfare-maximizing allocation (Algorithm 3); then, fixing this allocation, we solve for a winner-restricted pricing, abbreviated WiReP(Algorithm 4).[7] An agent employing the WE heuristic runs Algorithms 3 and 4; then, given $(\boldsymbol{X}_{\boldsymbol{r}}, \boldsymbol{p})$, it bids p_i in all market segments i for which $x_{ij} > 0$, and places limits $x_{ij}p_i$ in all market segments i. We provided some justification for this approach in our previous work [26], where we showed empirically that this two-step procedure results in allocation-price pairs that are very close to WE (in the sense that the constraints

[6]Not to be confused with **waterfalling**, the process by which publishers try to sell remnant inventory; see `https://clearcode.cc/blog/what-is-waterfalling/`

[7]Note that Algorithms 3 and 4 slightly generalize those presented in [26], in that they handle a vector of reserve prices, one per market segment.

Algorithm 3 Greedy allocation algorithm

procedure GREEDYALLOCATION($\mathscr{M}, \boldsymbol{r}$)$\rightarrow \boldsymbol{X_r}$
  input: Market $\mathscr{M}$ and reserve prices $\boldsymbol{r} \in \mathbb{R}_+^n$.
  output: Allocation $\boldsymbol{X_r}$.
  for all i, j, initialize $x_{ij} = 0$
  for $j \in C$ **do** ▷ Loop through j in descending order of bang-per-buck, i.e., ${}^{R_j}/_{I_j}$.
    Let $M_j = \{i \mid (i,j) \in E \text{ and } \sum_{j=1}^{c} x_{ij} < N_i\}$
    if $\sum_{i \in M_j} N_i - \sum_{j=1}^{c} x_{ij} \geq I_j$ **then** ▷ "enough supply remains to satisfy j"
      for $i \in M_j$ **do** ▷ Loop through M_j in descending order of supply.
        $x_{ij} = \min\{I_j - \sum_{i=1}^{n} x_{ij}, N_i - \sum_{j=1}^{c} x_{ij}\}$
      end for
      if $\sum_{i=1}^{n} r_i x_{ij} > R_j$ **then** ▷ "campaign j cannot afford the assigned bundle"
        for all i: $x_{ij} = 0$ ▷ Unallocate j.
      end if
    end if
  end for
  return $\boldsymbol{X_r}$
end procedure

Algorithm 4 Linear program to compute WiReP

procedure WIREP($\mathscr{M}, \boldsymbol{r}, \boldsymbol{X_r}$)$\rightarrow \boldsymbol{p}$
  input: Market $\mathscr{M}$, reserve prices $\boldsymbol{r} \in \mathbb{R}_+^n$, and allocation $\boldsymbol{X}_r$.
  output: A pricing $\boldsymbol{p}$.
  maximize$_{\boldsymbol{p},\boldsymbol{\alpha}}$ $\sum_{j \in C, i \in M} x_{ij} p_i - \sum_{j \in C, (i,k) \in M \times M} \alpha_{jik}$, subject to:
  (1) $\forall j \in C:$ If $\sum_{i=1}^{n} x_{ij} > 0$, then $\sum_{i=1}^{n} x_{ij} p_i \leq R_j$
  (2) $\forall i \in M, \forall j \in C:$ If $x_{ij} > 0$ then $\forall k \in M:$ If $(k,j) \in E$ and $x_{kj} < N_k$ then $p_i \leq p_k + \alpha_{jik}$
  (3) $\forall i \in M : p_i \geq r_i$
  (4) $\forall j \in C, \forall (i,k) \in M \times M : \alpha_{jik} \geq 0$
  return $\boldsymbol{p}$
end procedure

that define WE are violated with small additive errors), in a setup very similar to the static model of the one-day AdX game.

Waterfall Strategy

The Waterfall heuristic is designed to simulate the dynamics of the one-shot AdX game. In the game, the highest bid in a market segment stands for some period of time, namely until the spending limit of the winning campaign is reached, or the campaign is satisfied. Hence, prices in each market segment remain constant for some period of time at the current highest bid. As the game unfolds, more and more impressions are allocated, consuming the budgets of the winning campaigns in each market segment, until the winners drop out, and the second highest bidder is promoted. The bid of the campaign that drops out is by definition higher than the new winning bid, which again stands for some further period of time. The resulting prices thus resemble a waterfall: i.e., decreasing, and constant between decreases.

An agent employing the WF heuristic computes an allocation $\boldsymbol{X}$ and a price matrix $\boldsymbol{P}$ via Algorithm 5, given a market $\mathscr{M}$ and reserve prices $\boldsymbol{r}$. The algorithm works by simulating n second-price auctions assuming agents bid ${}^{R_j}/_{I_j}$, and then allocating impressions to winning campaigns from market segments sorted in ascending order by their second highest bids (breaking ties randomly),[8] and promoting campaigns as previous winners' budgets are exhausted. Given $(\boldsymbol{X}, \boldsymbol{P})$, the agent then bids p_{ij} in all market segments i for which $x_{ij} > 0$, and places limits $x_{ij} p_{ij}$ in all market segments i.

[8] This ordering is arbitrary; another possibility is to sort market segments in *descending* order by their second highest bids.

Algorithm 5 Waterfall Algorithm

procedure WATERFALL($\mathscr{M}, \boldsymbol{r}) \to \boldsymbol{X}, \boldsymbol{P}$
  input: A market $\mathscr{M}$ and reserve prices $\boldsymbol{r} \in \mathbb{R}_+^n$
  output: An allocation matrix $\boldsymbol{X_r}$ and a pricing matrix $\boldsymbol{P_r}$
  $\mathscr{N} = N$ ▷ a copy of the supply vector
  $\mathscr{C} = C$ ▷ a copy of the campaigns
  $\forall i, j : x_{ij} = 0$ ▷ initialize an empty allocation matrix
  $\forall i, j : p_{ij} = \infty$ ▷ initialize an infinite price matrix
  while $\mathscr{C}$ is not empty **do**
    for $j \in \mathscr{C}$ **do**
      if $\sum_{\{i|(i,j)\in E\}} \mathscr{N}_i < I_j$ **then** ▷ "j cannot be fulfilled with remaining supply"
        $\mathscr{C} = \mathscr{C} \setminus \{j\}$ ▷ remove j from the set of candidate campaigns
      end if
    end for
    Choose $j^* \in \arg\max_{j\in\mathscr{C}}\{R_j/I_j\}$
    for $i \in \{i \mid (i, j^*) \in E \text{ and } \mathscr{N}_i > 0\}$ **do**
      Initialize bid vector $\mathbf{b}_i$.
      for $j \in \mathscr{C}$ **do**
        if $(i, j) \in E$ **then**
          Insert bid R_j/I_j in $\mathbf{b}_i$
        end if
      end for
      Insert the 2nd-highest bid in $\mathbf{b}_i$ into vector `2nd_HIGHEST_BIDS`.
      Insert the reserve r_i into vector `2nd_HIGHEST_BIDS`.
    end for
    $i^* \in \arg\min_i\{\texttt{2nd\_HIGHEST\_BIDS}[i]\}$ ▷ select a market segment with the lowest 2nd highest bid
    $p_{i^*j^*} = \texttt{2nd\_HIGHEST\_BIDS}[i^*]$
    $x_{i^*j^*} = \min\{\mathscr{N}_{i^*}, I_{j^*} - \sum_{i=1}^n x_{ij^*}\}$ ▷ to satisfy the campaign, allocate the lessor of ▷ the impressions as needed or the remaining supply
    $\mathscr{N}_{i^*} = \mathscr{N}_{i^*} - x_{i^*j^*}$ ▷ decrement the supply
    if j^* is satisfied (i.e., $\sum_{i=1}^n x_{ij} = I_j$) **then**
      $\mathscr{C} = \mathscr{C} \setminus \{j\}$
    end if
  end while
  return $\boldsymbol{X}, \boldsymbol{P}$
end procedure

Experimental setup. We assumed three user attributes: gender, age, and income level, each with two values. We then simulated a fixed number of impression opportunities, namely $K = 500$, distributed over 8 market segments, corresponding to all the possible combinations of attribute values: $\{\text{MALE}, \text{FEMALE}\} \times \{\text{YOUNG}, \text{OLD}\} \times \{\text{LOW INCOME}, \text{HIGH INCOME}\}$. The distribution π over these impression opportunities was constructed from statistical data at `www.alexa.com` [18].

Each agent's campaign $C_j = \langle I_j, M_j, R_j\rangle$ is determined as follows: A market segment M_j is drawn uniformly over all 20 possible market segments corresponding to combinations of user attributes of size 2 (e.g., $\langle\text{MALE}, \text{YOUNG}\rangle$) and 3 (e.g., $\langle\text{MALE}, \text{YOUNG}, \text{LOW INCOME}\rangle$). Given M_j, the demand $I_j := {}^{K\pi_{M_j}}/_N$, where $K\pi_{M_j}$ is the expected size of market segment M_j, and N is the number of agents in the game. Given I_j, the budget R_j is a noisy signal of the demand modeled by a beta distribution: $R_j \sim I_j(\mathcal{B}(\alpha = 10, \beta = 10) + 0.5)$.

The task is to find an 8-dimensional vector of reserve prices $\boldsymbol{r} \in \mathbb{R}^8$, consisting of one reserve price per market segment, that maximizes the auctioneer's revenue, up to a desired accuracy ϵ and specified failure probability δ. We experiment with $N = 4$ agents, and two possible strategies; in particular, for all p, $S_p = \{\text{WE}, \text{WF}\}$. Since by construction these strategies never bid higher than a campaign's budget-per-impression, ${}^{R_j}/_{I_j} = \mathcal{B}(\alpha = 10, \beta = 10) + 0.5 \in [0.5, 1.5]$, we bound the search for each reserve price to this same range, and hence search the space $[0.5, 1.5]^8$.

Even in this bounded region, the complexity of this task is substantial: for each candidate vector of reserve prices, we

must first learn the corresponding game it induces, and then solve for the equilibria of this game. In this paper, we use Algorithm 2 for this purpose, which finds the set SCC_ϵ using Tarjan's algorithm [28], and then returns the minimum revenue among all SCC_ϵ equilibria. This is a computationally intensive task. Indeed, it took approximately 5 days to obtain these results using 4 machines, each with an E5-2690v4 (2.60GHz, 28Core) Xeon processor, and 1,536 GB of memory.

Experimental Results

Experimental results were obtained by running the procedure described in Algorithm 6, for each of $\mathcal{GP}$, $\mathcal{GP}$-$\mathcal{M}$, and $\mathcal{GP}$-$\mathcal{N}$. This procedure takes as input a simulator for parameterized game Γ_θ, a designer's objective function f (in this case, revenue), samples $\boldsymbol{X}$, a number of trials n_t, a number of initial reserve prices n_i, and a number of search steps n_s. It then outputs a sequence of length n_s, $\{\mu_s\}_{s=1}^{n_s}$, where each μ_s is the running maximum of the revenue obtained by to step s, averaged across n_t independent trials. More specifically, letting t index trials and s index steps, $\mu_s = \frac{1}{n_t}\sum_{t=1}^{n_t} \max_{k \le s} \hat{F}_t(\boldsymbol{r}_k^t)$, where $\hat{F}_t(\boldsymbol{r}_k^t)$ is the revenue assuming reserve prices $\boldsymbol{r}_k^t$, which denotes the reserve prices explored during the kth step of the tth trial.

Algorithm 6 Experimental Procedure

procedure EXPERIMENTALPROCEDURE($\Gamma_{\boldsymbol{r}}, f, \boldsymbol{X}, n_t, n_i, n_s) \to \{\mu_s\}_{s=1}^{n_s}$
  input: Black-box game parameterized by reserve prices $\Gamma_{\boldsymbol{r}}$, designer's objective function f, samples $\boldsymbol{X}$, number of trials n_t, number of initial reserves n_i, and number of search steps n_s.
  output: Running maximum, $\{\mu_s\}_{s=1}^{n_s}$, for n_s search steps, averaged across n_t trials.
  for $t = 1, \ldots, n_t$ **do** ▷ n_t trials.
    $\Theta = \{\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_i}\} \leftarrow$ GETINITIALRESERVEPRICES(t) ▷ Get n_i initial reserve price vectors.
    N.B. $\boldsymbol{r}_j = \langle r_j^1, \ldots, r_j^8 \rangle \in \mathcal{R}$, where r_j^M is the reserve price in market segment M.
    for $\boldsymbol{r} \in \mathcal{R}$ **do**
      $\hat{F}_t(\boldsymbol{r}) \leftarrow$ EMD_MEASURE($\Gamma_{\boldsymbol{r}}$, REVENUE, $\boldsymbol{X}$)
    end for
    $\mathcal{G}_t \leftarrow$ INITIALIZEGAUSSIANPROCESS($\{\boldsymbol{r}, \hat{F}_t(\boldsymbol{r})\}_{\boldsymbol{r} \in \mathcal{R}}$)
    $\nu_t = []$ ▷ Store trial t's running maximum in a list
    for $s = 1, \ldots, n_s$ **do** ▷ n_s search steps.
      $\boldsymbol{r}_s^t \leftarrow$ ACQUISITIONFUNCTION($\mathcal{G}_t$) ▷ We use Expected Improvement
      $\hat{F}_t(\boldsymbol{r}_s^t) \leftarrow$ EMD_MEASURE($\Gamma_{\boldsymbol{r}_s^t}$, REVENUE, $\boldsymbol{X}$)
      $\nu_t[s] = \max_{k \le s} \hat{F}_t(\boldsymbol{r}_k^t)$ ▷ Compute current running maximum
      $\mathcal{G}_t \leftarrow$ UPDATEGAUSSIANPROCESS($\mathcal{G}_t, \{\boldsymbol{r}_s^t, \hat{F}_t(\boldsymbol{r}_s^t)\}$)
    end for
  end for
  for $s = 1, \ldots, n_s$ **do**
    $\mu_s \leftarrow \frac{1}{n_t}\sum_{t=1}^{n_t} \nu_t[s]$ ▷ Averaging across trials
  end for
  return $\{\mu_s\}_{s=1}^{n_s}$
end procedure