Supplementary Material

A Regret Analysis

In this section we provide proofs for the theorems in Sec. 3.3.

Since those assume non-negative temporal losses, let us first modify the losses in Eq. 3 to be non-negative by adding the same constant $\log(\tilde{\alpha})$ to all possible values:

$$\tilde{\ell}_{t,i} = \begin{cases} 0 & \text{if } i \in C_t \text{ and } y_t > M_{t-1} \\ \log(\tilde{\alpha}\tilde{\beta}) & \text{if } i \in C_t \text{ and } y_t \leq M_{t-1} \\ \log(\tilde{\alpha}) & \text{if } i \notin C_t \end{cases} \tag{7}$$

This modification does not change the resulted distribution π_t induced over the coordinates as it is invariant to shifts of the losses:

$$\begin{aligned}\pi_{t,i} = \frac{w_{t,i}}{W_t} &= \frac{e^{-\eta \sum_{\tau=1}^{t} \tilde{\ell}_{\tau,i}}}{\sum_{j=1}^{D} e^{-\eta \sum_{\tau=1}^{t} \tilde{\ell}_{\tau,j}}} = \frac{e^{-\eta \sum_{\tau=1}^{t} (\ell_{\tau,i} + \log(\tilde{\alpha}))}}{\sum_{j=1}^{D} e^{-\eta \sum_{\tau=1}^{t} (\ell_{\tau,j} + \log(\tilde{\alpha}))}} = \frac{e^{-\eta t \log(\tilde{\alpha})} e^{-\eta \sum_{\tau=1}^{t} \ell_{\tau,i}}}{e^{-\eta t \log(\tilde{\alpha})} \sum_{j=1}^{D} e^{-\eta \sum_{\tau=1}^{t} \ell_{\tau,j}}} \\ &= \frac{e^{-\eta \sum_{\tau=1}^{t} \ell_{\tau,i}}}{\sum_{j=1}^{D} e^{-\eta \sum_{\tau=1}^{t} \ell_{\tau,j}}}\end{aligned} \tag{8}$$

Thus also $\tilde{\pi}_{t,i}$ and $\hat{\pi}_{t,i}$ introduced in sections A.1 and A.2 remain unchanged as well and for simplicity we refer to $\tilde{\ell}$ as ℓ throughout this section.

A.1 Regret analysis for sampling from the combinatorial space of coordinate blocks

The probability $\tilde{\pi}_{t,\mathcal{I}_t}$ of selecting a certain coordinate block $\mathcal{I}_t \subset \mathcal{I} = \{1, \cdots, D\}$ of size $|\mathcal{I}_t| = c \in \mathcal{C}$ follows sampling according to π_t such that:

$$\tilde{w}_{t,\mathcal{I}_t} = \prod_{i \in \mathcal{I}_t} w_{t,i}^{\frac{1}{|\mathcal{I}_t|}} \quad ; \quad \tilde{W}_t = \sum_{c \in \mathcal{C}} \sum_{\mathcal{I}_t \in \mathcal{S}_c} \tilde{w}_{t,\mathcal{I}_t} \quad ; \quad \tilde{\pi}_{t,\mathcal{I}_t} = \frac{\tilde{w}_{t,\mathcal{I}_t}}{\tilde{W}_t} \quad \forall \mathcal{I}_t \in \bigcup_{c \in \mathcal{C}} \mathcal{S}_c \tag{9}$$

such that

$$\sum_{c \in \mathcal{C}} \sum_{\mathcal{I}_t \in \mathcal{S}_c} \tilde{\pi}_{t,\mathcal{I}_t} = 1 \tag{10}$$

A.1.1 Proof of Lemma 1

Lemma 1. *For $\eta > 0$ and non-negative losses $\ell_{t,i} \geq 0$ the update rule in (4) satisfies for any block of coordinates $\mathcal{I}^*$:*

$$\begin{aligned}&\sum_{t=1}^{T} \sum_{c \in \mathcal{C}} \sum_{\mathcal{I}_t \in \mathcal{S}_c} \tilde{\pi}_{t,\mathcal{I}_t} \cdot \frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} \ell_{t,i} - \sum_{t=1}^{T} \frac{1}{|\mathcal{I}^*|} \sum_{i \in \mathcal{I}^*} \ell_{t,i} \leq \\ &\qquad \eta \sum_{t=1}^{T} \sum_{c \in \mathcal{C}} \sum_{\mathcal{I}_t \in \mathcal{S}_c} \tilde{\pi}_{t,\mathcal{I}_t} \cdot \left(\frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} \ell_{t,i} \right)^2 + \frac{D \log(D)}{\eta}\end{aligned} \tag{11}$$

Proof. Set

$$\tilde{w}_{0,\mathcal{I}_t} = 1 \quad \forall \mathcal{I}_t \in \bigcup_{c \in \mathcal{C}} \mathcal{S}_c \tag{12}$$

Thus,

$$\begin{aligned}
\tilde{W}_{t+1} &= \sum_{c\in\mathcal{C}}\sum_{\mathcal{I}_t\in\mathcal{S}_c}\tilde{w}_{t+1,\mathcal{I}_t} = \sum_{c\in\mathcal{C}}\sum_{\mathcal{I}_t\in\mathcal{S}_c}\prod_{i\in\mathcal{I}_t} w_{t+1,i}^{\frac{1}{|\mathcal{I}_t|}} && (13)\\
&= \sum_{c\in\mathcal{C}}\sum_{\mathcal{I}_t\in\mathcal{S}_c}\prod_{i\in\mathcal{I}_t} w_{t,i}^{\frac{1}{|\mathcal{I}_t|}} e^{-\frac{\eta}{|\mathcal{I}_t|}\ell_{t,i}} = \sum_{c\in\mathcal{C}}\sum_{\mathcal{I}_t\in\mathcal{S}_c}\prod_{i\in\mathcal{I}_t} w_{t,i}^{\frac{1}{|\mathcal{I}_t|}} \cdot e^{-\frac{\eta}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i}} && (14)\\
&= \sum_{c\in\mathcal{C}}\sum_{\mathcal{I}_t\in\mathcal{S}_c}\tilde{w}_{t,\mathcal{I}_t}\cdot e^{-\frac{\eta}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i}} && (15)\\
&= \tilde{W}_t\sum_{c\in\mathcal{C}}\sum_{\mathcal{I}_t\in\mathcal{S}_c}\tilde{\pi}_{t,\mathcal{I}_t}\cdot e^{-\frac{\eta}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i}} && (16)\\
&\le \tilde{W}_t\sum_{c\in\mathcal{C}}\sum_{\mathcal{I}_t\in\mathcal{S}_c}\tilde{\pi}_{t,\mathcal{I}_t}\left(1-\frac{\eta}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i}+\eta^2\left(\frac{1}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)^2\right) && (17)\\
&\le \tilde{W}_t\left(1+\sum_{c\in\mathcal{C}}\left(\sum_{\mathcal{I}_t\in\mathcal{S}_c}\eta^2\tilde{\pi}_{t,\mathcal{I}_t}\left(\frac{1}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)^2-\frac{\eta}{|\mathcal{I}_t|}\tilde{\pi}_{t,\mathcal{I}_t}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)\right) && (18)\\
&\le \tilde{W}_t e^{\sum_{c\in\mathcal{C}}\cdot\left(\sum_{\mathcal{I}_t\in\mathcal{S}_c}\eta^2\tilde{\pi}_{t,\mathcal{I}_t}\left(\frac{1}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)^2-\frac{\eta}{|\mathcal{I}_t|}\tilde{\pi}_{t,\mathcal{I}_t}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)} && (19)
\end{aligned}$$

where,

- (16) follows from (9).
- (17) holds since $e^{-x}\le 1-x+x^2$ for $x\ge 0$.
- (18) holds due to Eq. 10.
- (19) holds since $1+x\le e^x$.

Due to Eq. (12), we have,

$$\tilde{w}_{t,\mathcal{I}_t} = \prod_{i\in\mathcal{I}_t} w_{t,i}^{\frac{1}{|\mathcal{I}_t|}} = \prod_{i\in\mathcal{I}_t} w_{0,i}^{\frac{1}{|\mathcal{I}_t|}} e^{-\frac{\eta}{|\mathcal{I}_t|}\sum_{t=1}^{T}\ell_{t,i}} = e^{-\frac{\eta}{\mathcal{I}_t}\sum_{t=1}^{T}\sum_{i\in\mathcal{I}_t}\ell_{t,i}} \tag{20}$$

And,

$$W_0 = \sum_{c\in\mathcal{C}}\sum_{\mathcal{I}_t\in\mathcal{S}_c}\tilde{w}_{0,\mathcal{I}_t} = \sum_{c\in\mathcal{C}}\sum_{\mathcal{I}_t\in\mathcal{S}_c} 1 = \sum_{c\in\mathcal{C}}|\mathcal{S}_c| = \sum_{c\in\mathcal{C}}\binom{D}{c}\le (D!)^{|\mathcal{C}|} \tag{21}$$

Given that the weight of a certain coordinate block $\mathcal{I}^*$ is less than the total sum of all weights, together with Eq. (19), (12) and (21) we have:

$$\begin{aligned}
e^{-\frac{\eta}{|\mathcal{I}^*|}\sum_{t=1}^{T}\sum_{i\in\mathcal{I}^*}\ell_{t,i}} &= \tilde{w}_{t,\mathcal{I}^*}\le\tilde{W}_T\\
&\le (D!)^{|\mathcal{C}|} e^{\sum_{t=1}^{T}\sum_{c\in\mathcal{C}}\cdot\left(\sum_{\mathcal{I}_t\in\mathcal{S}_c}\eta^2\tilde{\pi}_{t,\mathcal{I}_t}\left(\frac{1}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)^2-\frac{\eta}{|\mathcal{I}_t|}\tilde{\pi}_{t,\mathcal{I}_t}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)} && (22)
\end{aligned}$$

Taking the log of both sides, we have:

$$\begin{aligned}
-\eta\sum_{t=1}^{T}\frac{1}{|\mathcal{I}^*|}\sum_{i\in\mathcal{I}^*}\ell_{t,i} \le &\sum_{t=1}^{T}\sum_{c\in\mathcal{C}}\cdot\left(\sum_{\mathcal{I}_t\in\mathcal{S}_c}\eta^2\tilde{\pi}_{t,\mathcal{I}_t}\left(\frac{1}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)^2-\frac{\eta}{|\mathcal{I}_t|}\tilde{\pi}_{t,\mathcal{I}_t}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)\\
&+|\mathcal{C}|\log(D!) && (23)
\end{aligned}$$

and since $D!\le D^D$ the result follows. □

A.1.2 Proof of Theorem 1

Proof. Since $\ell_{t,i} \leq \log(\tilde{\alpha}\tilde{\beta})$ then:

$$\left(\frac{1}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)^2 \leq \left(\frac{1}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\log(\tilde{\alpha}\tilde{\beta})\right)^2 \leq \log(\tilde{\alpha}\tilde{\beta})^2 \tag{24}$$

Thus due to Eq. (10):

$$\sum_{c\in\mathcal{C}}\sum_{\mathcal{I}_t\in\mathcal{S}_c}\tilde{\pi}_{t,\mathcal{I}_t}\cdot\left(\frac{1}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)^2 \leq \sum_{c\in\mathcal{C}}\sum_{\mathcal{I}_t\in\mathcal{S}_c}\tilde{\pi}_{t,\mathcal{I}_t}\log(\tilde{\alpha}\tilde{\beta})^2 = \log(\tilde{\alpha}\tilde{\beta})^2 \tag{25}$$

And setting $\eta = \frac{1}{\log(\tilde{\alpha}\tilde{\beta})}\sqrt{\frac{|\mathcal{C}|D\log(D)}{T}}$ in Eq. (11) yields:

$$Regret_t \leq \eta T\log(\tilde{\alpha}\tilde{\beta})^2 + \frac{|\mathcal{C}|D\log(D)}{\eta} = 2\log(\tilde{\alpha}\tilde{\beta})\sqrt{T|\mathcal{C}|D\log(D)} \tag{26}$$

$\square$

A.2 Regret analysis for sampling coordinates without replacement

Denote by p_c the probability of choosing a certain block size $c \in \mathcal{C}$, such that $p_c > 0$ and $\sum_{c\in\mathcal{C}} p_c = 1$, e.g., for a uniform sampling of the block size $p_c = 1/|\mathcal{C}|$ for all $c \in \mathcal{C}$.

The probability $\hat{\pi}_{t,\mathcal{I}_t}$ of selecting a certain coordinate block $\mathcal{I}_t \subset \mathcal{I} = \{1, \cdots, D\}$ of size $|\mathcal{I}_t| = c \in \mathcal{C}$ follows sampling according to π_t (Eq. (2)) without replacement, such that,

$$\hat{\pi}_{t,\mathcal{I}_t} = \sum_{p\in perm(\mathcal{I}_t)}\prod_{k\in p}\frac{\pi_{t,k}}{1-\sum_{j\in p_{1:k}}\pi_{t,j}} \tag{27}$$

$$= \left(\prod_{i\in\mathcal{I}_t}\pi_{t,i}\right)\cdot\left(\sum_{p\in perm(\mathcal{I}_t)}\prod_{k\in p}\left(1-\sum_{j\in p_{1:k}}\pi_{t,j}\right)^{-1}\right) = \mathcal{P}(\mathcal{I}_t)\cdot\mathcal{R}(\mathcal{I}_t) \tag{28}$$

where $perm(\mathcal{I}_t)$ are all the permutations of the set $\mathcal{I}_t$ and $p_{1:k}$ are the first k coordinates in the permutation p. Eq. (28) holds due to the common numerator of all permutations where the left term $\mathcal{P}(\mathcal{I}_t)$ corresponds to the probability of sampling a subset of coordinates with replacement, and the right term $\mathcal{R}(\mathcal{I}_t)$ is associated with sampling without replacement. Of course, summing over all the possible blocks of size c results $\sum_{\mathcal{I}_t\in\mathcal{S}_c}\hat{\pi}_{t,\mathcal{I}_t} = 1$ for all $c \in \mathcal{C}$.

Thus $\tilde{\pi}_{t,\mathcal{I}_t} = p_c \cdot \hat{\pi}_{t,\mathcal{I}_t}$ and the probability of sampling every block of coordinates of any size sum up to 1 as well:

$$\sum_{c\in\mathcal{C}}\sum_{\mathcal{I}_t\in\mathcal{S}_c}\tilde{\pi}_{t,\mathcal{I}_t} = \sum_{c\in\mathcal{C}}p_c\sum_{\mathcal{I}_t\in\mathcal{S}_c}\hat{\pi}_{t,\mathcal{I}_t} = \sum_{c\in\mathcal{C}}p_c = 1 \tag{29}$$

A.2.1 Proof of Lemma 2

Lemma 2. *Sample a block size $c \in \mathcal{C}$ with probability $p_c > 0$ and c coordinates without replacement according to π_t. Assume $\mathcal{C} \supset \{1\}$, $\eta > 0$ and non-negative losses $\ell_{t,i} \geq 0$. Then the update rule in (4) satisfies for any block of coordinates $\mathcal{I}^*$:*

$$\sum_{t=1}^{T}\sum_{c\in\mathcal{C}}p_c\sum_{\mathcal{I}_t\in\mathcal{S}_c}\hat{\pi}_{t,\mathcal{I}_t}\cdot\frac{1}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i} - \sum_{t=1}^{T}\frac{1}{|\mathcal{I}^*|}\sum_{i\in\mathcal{I}^*}\ell_{t,i}$$
$$\leq \eta\sum_{t=1}^{T}\sum_{c\in\mathcal{C}}p_c\sum_{\mathcal{I}_t\in\mathcal{S}_c}\hat{\pi}_{t,\mathcal{I}_t}\cdot\left(\frac{1}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)^2 + \frac{\log(D)}{\eta} - \frac{T\log(p_1)}{\eta} \tag{30}$$

Proof. Starting with a uniform distribution over the coordinates $w_{0,i} \equiv \frac{1}{D}$ such that $W_0 = 1$ and we have:

$$
\begin{aligned}
p_1 \cdot W_{t+1} &= p_1 \cdot \sum_{i\in\mathcal{I}} w_{t+1,i} && (31)\\
&\le \sum_{c\in\mathcal{C}} p_c \sum_{\mathcal{I}_t\in\mathcal{S}_c} \prod_{i\in\mathcal{I}_t} w_{t+1,i} && (32)\\
&= W_t \sum_{c\in\mathcal{C}} p_c \sum_{\mathcal{I}_t\in\mathcal{S}_c} W_t^{-1} \prod_{i\in\mathcal{I}_t} w_{t,i} e^{-\eta\ell_{t,i}} && (33)\\
&\le W_t \sum_{c\in\mathcal{C}} p_c \sum_{\mathcal{I}_t\in\mathcal{S}_c} W_t^{-|\mathcal{I}_t|} \prod_{i\in\mathcal{I}_t} w_{t,i} e^{-\eta\ell_{t,i}} \cdot |perm(\mathcal{I}_t)| && (34)\\
&= W_t \sum_{c\in\mathcal{C}} p_c \sum_{\mathcal{I}_t\in\mathcal{S}_c} \prod_{i\in\mathcal{I}_t} \frac{w_{t,i}}{W_t} e^{-\eta\ell_{t,i}} \cdot \sum_{p\in perm(\mathcal{I}_t)} 1 && (35)\\
&= W_t \sum_{c\in\mathcal{C}} p_c \sum_{\mathcal{I}_t\in\mathcal{S}_c} \prod_{i\in\mathcal{I}_t} \pi_{t,i} e^{-\eta\ell_{t,i}} \cdot \sum_{p\in perm(\mathcal{I}_t)} \prod_{k\in p} 1 && (36)\\
&\le W_t \sum_{c\in\mathcal{C}} p_c \sum_{\mathcal{I}_t\in\mathcal{S}_c} e^{-\eta\sum_{i\in\mathcal{I}_t}\ell_{t,i}} \prod_{i\in\mathcal{I}_t} \pi_{t,i} \cdot \sum_{p\in perm(\mathcal{I}_t)} \prod_{k\in p} \left(1 - \sum_{j\in p_{1:k}} \pi_{t,j}\right)^{-1} && (37)\\
&= W_t \sum_{c\in\mathcal{C}} p_c \sum_{\mathcal{I}_t\in\mathcal{S}_c} \hat{\pi}_{t,\mathcal{I}_t} e^{-\eta\sum_{i\in\mathcal{I}_t}\ell_{t,i}} && (38)\\
&\le W_t \sum_{c\in\mathcal{C}} p_c \sum_{\mathcal{I}_t\in\mathcal{S}_c} \hat{\pi}_{t,\mathcal{I}_t} e^{-\frac{\eta}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i}} && (39)\\
&\le W_t \sum_{c\in\mathcal{C}} p_c \sum_{\mathcal{I}_t\in\mathcal{S}_c} \hat{\pi}_{t,\mathcal{I}_t} \left(1 - \frac{\eta}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i} + \eta^2\left(\frac{1}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)^2\right) && (40)\\
&\le W_t \left(1 + \sum_{c\in\mathcal{C}} p_c \cdot \left(\sum_{\mathcal{I}_t\in\mathcal{S}_c} \eta^2\hat{\pi}_{t,\mathcal{I}_t}\left(\frac{1}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)^2 - \frac{\eta}{|\mathcal{I}_t|}\hat{\pi}_{t,\mathcal{I}_t}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)\right) && (41)\\
&\le W_t e^{\sum_{c\in\mathcal{C}} p_c \cdot \left(\sum_{\mathcal{I}_t\in\mathcal{S}_c} \eta^2\hat{\pi}_{t,\mathcal{I}_t}\left(\frac{1}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)^2 - \frac{\eta}{|\mathcal{I}_t|}\hat{\pi}_{t,\mathcal{I}_t}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)} && (42)
\end{aligned}
$$

where

- (32) is since $\mathcal{C} \supset \{1\}$ always contains a block size of 1 and thus:

$$
\begin{aligned}
\sum_{c\in\mathcal{C}} p_c \sum_{\mathcal{I}_t\in\mathcal{S}_c} \prod_{i\in\mathcal{I}_t} w_{t+1,i} &= p_1 \sum_{\mathcal{I}_t\in\mathcal{S}_1} \prod_{i\in\mathcal{I}_t} w_{t+1,i} + \sum_{c\in\mathcal{C}\setminus\{1\}} p_c \sum_{\mathcal{I}_t\in\mathcal{S}_c} \prod_{i\in\mathcal{I}_t} w_{t+1,i}\\
&= p_1 \sum_{i\in\mathcal{I}} w_{t+1,i} + \sum_{c\in\mathcal{C}\setminus\{1\}} p_c \sum_{\mathcal{I}_t\in\mathcal{S}_c} \prod_{i\in\mathcal{I}_t} w_{t+1,i}\\
&\ge p_1 \sum_{i\in\mathcal{I}} w_{t+1,i}
\end{aligned}
$$

- (34) holds since $W_0 = 1$ and W_t is monotonically non-increasing following the update rule (4) with non-negative losses, thus $w_t \le 1$ for all t.
- (38) follows from (28).
- (40) holds since $e^{-x} \le 1 - x + x^2$ for $x \ge 0$.
- (41) holds due to Eq. 29.
- (42) holds since $1 + x \le e^x$.

Given that the sum of weights of a certain coordinate block $\mathcal{I}^*$ is less than the total sum of weights, together with Eq. 42, $w_{0,i} \equiv \frac{1}{D}$ and $W_0 = 1$ we have:

$$\frac{1}{D}\sum_{i\in\mathcal{I}^*} e^{-\eta\sum_{t=1}^{T}\ell_{t,i}} = \sum_{i\in\mathcal{I}^*} w_{t,i} \leq W_T$$
$$\leq p_1^{-T} e^{\sum_{t=1}^{T}\sum_{c\in\mathcal{C}} p_c\cdot\left(\sum_{\mathcal{I}_t\in\mathcal{S}_c}\eta^2\hat{\pi}_{t,\mathcal{I}_t}\left(\frac{1}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)^2 - \frac{\eta}{|\mathcal{I}_t|}\hat{\pi}_{t,\mathcal{I}_t}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)} \tag{43}$$

Taking the log of both sides, we have:

$$\log\left(\sum_{i\in\mathcal{I}^*} e^{-\eta\sum_{t=1}^{T}\ell_{t,i}}\right) - \log(D)$$
$$\leq \sum_{t=1}^{T}\sum_{c\in\mathcal{C}} p_c\cdot\left(\sum_{\mathcal{I}_t\in\mathcal{S}_c}\eta^2\hat{\pi}_{t,\mathcal{I}_t}\left(\frac{1}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)^2 - \frac{\eta}{|\mathcal{I}_t|}\hat{\pi}_{t,\mathcal{I}_t}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right) - T\log(p_1) \tag{44}$$

Following the same certain block, all the participating coordinates suffer the same loss ℓ_t^* at every time step as follows from Eq. 3, hence:

$$\begin{aligned}\log\left(\sum_{i\in\mathcal{I}^*} e^{-\eta\sum_{t=1}^{T}\ell_{t,i}}\right) &= \log\left(\sum_{i\in\mathcal{I}^*} e^{-\eta\sum_{t=1}^{T}\ell_t^*}\right)\\ &= \log\left(|\mathcal{I}^*|e^{-\eta\sum_{t=1}^{T}\ell_t^*}\right)\\ &= \log(|\mathcal{I}^*|) - \eta\sum_{t=1}^{T}\ell_t^*\\ &\geq -\eta\sum_{t=1}^{T}\ell_t^*\end{aligned} \tag{45}$$

Thus Eq. 44 and 45 yield:

$$-\eta\sum_{t=1}^{T}\ell_t^* - \log(D) \leq \sum_{t=1}^{T}\sum_{c\in\mathcal{C}} p_c\cdot\left(\sum_{\mathcal{I}_t\in\mathcal{S}_c}\eta^2\hat{\pi}_{t,\mathcal{I}_t}\left(\frac{1}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)^2 - \frac{\eta}{|\mathcal{I}_t|}\hat{\pi}_{t,\mathcal{I}_t}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right) - T\log(p_1) \tag{46}$$

And the result follows. □

A.2.2 Proof of Theorem 2

Proof. Since $\ell_{t,i} \leq \log(\tilde{\alpha}\tilde{\beta})$ then:

$$\left(\frac{1}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)^2 \leq \left(\frac{1}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\log(\tilde{\alpha}\tilde{\beta})\right)^2 \leq \log(\tilde{\alpha}\tilde{\beta})^2 \tag{47}$$

Thus due to Eq. 29:

$$\sum_{c\in\mathcal{C}} p_c \sum_{\mathcal{I}_t\in\mathcal{S}_c}\hat{\pi}_{t,\mathcal{I}_t}\cdot\left(\frac{1}{|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t}\ell_{t,i}\right)^2 \leq \sum_{c\in\mathcal{C}} p_c \sum_{\mathcal{I}_t\in\mathcal{S}_c}\hat{\pi}_{t,\mathcal{I}_t}\log(\tilde{\alpha}\tilde{\beta})^2 = \log(\tilde{\alpha}\tilde{\beta})^2 \tag{48}$$

And Eq. 30 reads,

$$Regret_t \leq \eta T\log(\tilde{\alpha}\tilde{\beta})^2 + \frac{\log(D)}{\eta} - \frac{T\log(p_1)}{\eta} \tag{49}$$

Choosing $\eta \geq 1$, we have:

$$Regret_t \leq \eta T\log(\tilde{\alpha}\tilde{\beta})^2 + \frac{\log(D)}{\eta} - \eta T\log(p_1) = \eta T(\log(\tilde{\alpha}\tilde{\beta})^2 - \log(p_1)) + \frac{\log(D)}{\eta} \tag{50}$$

Thus setting $\eta = \sqrt{\frac{\log(D)}{T(\log(\tilde{\alpha}\tilde{\beta})^2 - \log(p_1))}} \geq 1$ finally we have:

$$Regret_t \leq \mathcal{O}\left(\sqrt{(\log(\tilde{\alpha}\tilde{\beta})^2 - \log(p_1)) \cdot T \log(D)}\right) \tag{51}$$

$\square$

Remark: Note that the condition $\eta \geq 1$ can be replaced by setting an appropriate $p_1 = \sqrt[T]{\epsilon}$ for $0 < \epsilon \leq 1$. Thus Eq. 49 reads:

$$Regret_t \leq \eta T \log(\tilde{\alpha}\tilde{\beta})^2 + \frac{\log(D) - \log(\epsilon)}{\eta} \tag{52}$$

The setting $\eta = \frac{1}{\log(\tilde{\alpha}\tilde{\beta})}\sqrt{\frac{\log(D)-\log(\epsilon)}{T}}$ yields:

$$Regret_t \leq \mathcal{O}\left(\log(\tilde{\alpha}\tilde{\beta})^{-1}\sqrt{T(\log(D) - \log(\epsilon))}\right) \tag{53}$$

A.3 Regret analysis for consistent queries

The regret analyses presented in sections A.1 and A.2 hold when incorporating the consistent queries mentioned in section 3.2 for an adapted settings.

Consider the update rule of Eq. 4 at each time step $t = 1, \ldots, T$ where the sampling of next coordinate blocks happens for $K \leq T$ time steps at $0 = t_0 < t_1 < \cdots < t_k < \cdots < t_{K-1} < t_k = T$. Both K and $\{t_k\}_{k=0}^{K-1}$ are unknown in advance and are revealed to the decision maker along the process. At each time t_k a coordinate block is selected and fixed for the next $t_{k+1} - t_k$ steps. The effective losses incurred to the coordinates are the aggregation of all the temporal losses in this time interval $t \in [t_k, t_{k+1} - 1]$:

$$\bar{\ell}_{k,i} = \sum_{t=t_k}^{t_{k+1}-1} \ell_{t,i} \tag{54}$$

where those are all non-negative $\bar{\ell}_{k,i} \geq 0$ since $\ell_{t,i} \geq 0$.

Since the update rule in Eq. 2 is applied in every time step $t = 1, \ldots, T$, we effectively have:

$$w_{k+1,i} = w_{k,i} \prod_{t=t_k}^{t_{k+1}-1} e^{-\eta \ell_{t,i}} = w_{k,i} e^{-\eta \sum_{t=t_k}^{t_{k+1}-1} \ell_{t,i}} = w_{k,i} e^{-\eta \bar{\ell}_{k,i}} \tag{55}$$

Define the stopping rule mentioned in section 3.2 such that the number of consistent queries in a subspace does not cross $\tau \in [1, 2, \ldots, T]$, such that:

$$t_{k+1} - t_k \leq \tau \quad \forall k = 0, \ldots, K-1 \tag{56}$$

and thus $\bar{\ell}_{k,i} \leq \tau \log(\tilde{\alpha}\tilde{\beta})$ since $\ell_{t,i} \leq \log(\tilde{\alpha}\tilde{\beta})$.

Hence, all the results hold by replacing T with K and $\log(\tilde{\alpha}\tilde{\beta})$ with $\tau \log(\tilde{\alpha}\tilde{\beta})$.

B Implementation

The proposed CobBO algorithm is implemented in Python 3. The source code is attached for review and is publicly released online. The original log files of all the experiments are attached for the review. The specifications of the testbed are as follows: CPU: Intel(R) Xeon(R) CPU E5-2682 v4 2.50GHz, Memory: 32GB, GPU: NVIDIA Tesla P100 PCIe 16GB.

The code has been utilized for various complex real-world applications and handles many corner cases (hence the error fallbacks). For example, a parameter "smooth" of Scipy RBF (kernel=multiquadric, default=0.0) is increased by 0.02 upon "try catch" numerical issues of ill conditioning.

C Auxiliary components and corner cases

Besides the key components of CobBO, several auxiliary components are utilized for dealing with a larger variety of problems and corner cases.

C.1 ϵ-greedy block selection

In order to balance between exploitation and exploration, we alternate between two different approaches in selecting C_t. For the first approach that emphasizes exploitation, we estimate the top performing coordinate directions. A similar method is used in [39]. We select C_t to be the coordinates with the largest absolute gradient values of the RBF regression on the whole space Ω at point V_t.

The second selection policy is as described in Sec. 3.1 works well for low dimensions where $|C_t|/D$ is relatively large, as shown in Section 4.2.1. However, in high dimensions, $|C_t|/D$ could be small. In this case, additionally we also encourage cyclic order for exploration. With a certain probability ϵ (e.g., $\epsilon = 0.3$), we select $|C_t|$ coordinates whose π_t values are the largest, and with probability $1 - \epsilon$, we randomly sample a coordinate subset according to the distribution π_t without replacement. Picking the coordinates with the largest values approximately implements a cyclic order, due to the selected weights update (Eq. 2) incurring probability oscillations. Since improvements tend to be less common than failures, the weights of the selected coordinates tend to decrease as the probability for choosing unselected coordinates increase in turn.

C.2 Designing a stopping rule

Section 3.2 describes the considerations for designing a stopping rule that determines when to sample a new coordinate block and perform Bayesian optimization in the corresponding subspace. Below are the details of CobBO that designs a rule based heuristic stopping time for a large variety of problems and corner cases.

For each iteration t, denote the relative improvement at iteration t by $\Delta_t = \frac{y_t - M_{t-1}}{\max(|M_{t-1}|, 0.1)}$. When looking backward in time from iteration t, denote by P_t the number of consecutive improvements ($\Delta_s > 0, s \le t$) and by N_t the total number of consecutive queries in the same subspace as in Ω_t, respectively. We set

$$C_{t+1} = \begin{cases} \text{sample a new coordinate block}, & N_t \ge \tau \text{ and } \Delta_t \le 0.1 \text{ and } P_t \le \xi \\ C_t, & N_t < \tau \text{ or } \Delta_t > 0.1 \text{ or } P_t > \xi \end{cases}$$

where the value τ depends on both T and D, e.g.,

$$\tau = \frac{T}{1000} + \begin{cases} 1 & D < 20 \\ 2 & 20 \le D < 70 \\ 3 & 70 \le D < 100 \\ 4 & 100 \le D < 200 \\ 5 & 200 \le D \end{cases} \quad ; \quad \xi = \begin{cases} 4 & \Delta_t < 0.05 \\ 2 & 0.05 \le \Delta_t \le 0.1 \\ 0 & 0.1 < \Delta_t \end{cases}$$

C.3 Escaping trapped local optima

CobBO can be viewed as a variant of block coordinate ascent. Each subspace Ω_t contains a pivot point V_t. If fixing the coordinates' values incorrectly, one is condemned to move in a suboptimal subspace. Considering that those are determined by V_t, it has to be changed in the face of many consecutive failures to improve over M_t in order to escape this trapped local maxima. We do that by decreasing the observed function value at V_t and setting V_{t+1} as a selected sub-optimal random point in $\mathcal{X}_t$. Specifically, we randomly sample a few points (e.g., 5) in $\mathcal{X}_t$ with their values above the median and pick the one furthest away from V_t.

Figure 6 shows that the way CobBO escapes local optima is beneficial.

We further the experiment with Levy and Ackley functions of 100 dimensions, as described in Section E.3 to compute the fraction of queries that improve the already observed maximal points due to the change of V_t.

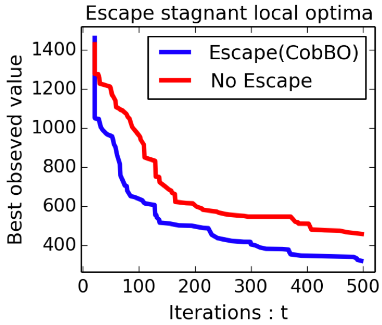

Figure 6: Ablation study for escaping local optima for Rastrigin on $[-5, 10]^{50}$ with 20 initial random samples. The best performing run out of 5 runs for each configuration is presented.

Problem	Average # improved queries	Average # improved queries due to escaping
Ackley	228	15.3
Levy	155	3

Table 1: The number of improved queries due to escaping local maxima

We observe that optimizing the Levy function yields very few queries that improve the maximal points by changing the pivot point, while optimizing the Ackley function can benefit more from that.

C.4 Forming trust regions on two time scales

Trust regions have been shown to be effective in Bayesian optimization [14, 1, 20, 41]. They are formed by shrinking the domain, e.g., by centering at V_t and halving the domain in each coordinate. CobBO forms coarse and fine trust regions on both slow and fast time scales, respectively, and alternates between them. This brings yet another tradeoff between exploration and exploitation. Since sampled points tend to reside near the boundaries in high dimensions [47], inducing trust regions encourages sampling densely in the interior. However, aggressively shrinking those trust regions too fast around V_t can lead to an over-exploitation, getting trapped in a local optimum. Hence, we alternate between two trust regions, following different time scales, as fast ones are formed inside slow ones. When the former allows fast exploitation of local optima, the latter avoids getting trapped in those.

The refinements of trust regions are triggered when a virtual clock K_t, characterizing the Bayesian optimization progress, reaches certain thresholds. Specifically,

$$K_{t+1} = \begin{cases} K_t + 1 & \text{if } \Delta_t \leq 0 \\ \gamma_t(\Delta_t, x_t, x_{t-1}) \cdot K_t & \text{if } 0 < \Delta_t \leq \delta \\ 0 & \text{if } \Delta_t > \delta \end{cases} \tag{57}$$

where $\Delta_t = \frac{y_t - M_{t-1}}{\max(|M_{t-1}|, 0.1}$ is the relative improvement and for example,

$$\gamma_t(\Delta_t, x_t, x_{t-1}) = \left(1 - \frac{\Delta_t}{\delta}\right) \cdot \left(1 - \frac{||x_t - x_{t-1}||}{\sqrt{|C_t|}}\right)$$

Starting from the full domain Ω, on a slow time scale, every time K_t reaches a threshold κ_S (e.g., $\kappa_S = 30$), a coarse trust region Ω_S is formed followed by setting $K_{t+1} = 0$. Within the coarse trust region, on a fast time scale, when the number of consecutive fails exceeds a threshold $\kappa_F < \kappa_S$ (e.g, $\kappa_F = 6$), a fine trust region is formed. In face of improvement, both the trust regions are back to the previous refinement of the coarse one.

In addition, when the amount of queried points exceeds a threshold, e.g., 70% of the query budget, we shrink the total space Ω every time when the fraction of the queried points increases by 10%.

Figure 7 compares CobBO with two other schemes: without any trust regions and forming only coarse trust regions. Two time scales yields better results.

Algorithm 2: FormTrustRegions(K_t,y_t,M_{t-1})

Parameters:
  Slow/fast thresholds $\kappa_{S/F}$ respectively
  Fast duty cycle τ_F
Init: $\Omega_0, \tilde{\Omega}_0 \leftarrow \Omega$
if $y_t > M_{t-1}$ **then**
  $\tilde{\Omega}_t \leftarrow$ Double $\tilde{\Omega}_{t-1}$ around V_t
  $\Omega_t \leftarrow \tilde{\Omega}_t$ [$\tilde{\Omega}_t$ is the trust region formed on the slow time scale]
else if $K_t == \kappa_S$ **then**
  $\tilde{\Omega}_t \leftarrow$ Halve $\tilde{\Omega}_t$ around V_t
  $\Omega_t \leftarrow \tilde{\Omega}_t$
  Reset $K_t = 0$
else
  $\tilde{\Omega}_t \leftarrow \tilde{\Omega}_{t-1}$
  if $mod\,(K_t, \kappa_F + \tau_F) == \kappa_F - 1$ **then**
    $\Omega_t \leftarrow$ Halve Ω_{t-1} around V_t
  else if $mod\,(K_t, \kappa_F + \tau_F) == \kappa_F + \tau_F - 1$ **then**
    $\Omega_t \leftarrow \tilde{\Omega}_t$
  else
    $\Omega_t \leftarrow \Omega_{t-1}$
Output: Trust Region Ω_t

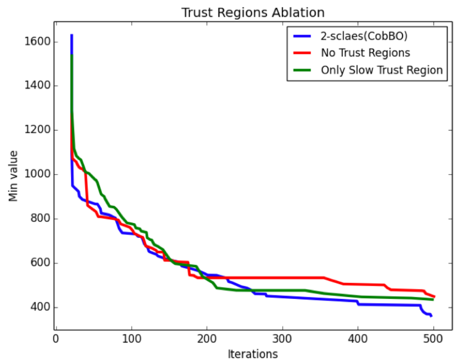

Figure 7: Ablation study for the trust regions of two scales for Rastrigin on $[-5, 10]^{50}$ with 20 initial random samples. The best performing run out of 5 runs for each configuration is presented.

D Default hyper-parameter configuration

Table 2 specifies the default configuration of CobBO used for all the benchmarks in this paper.

E The selected hyperparameters are robust to many problems

We provide more experiments using the very same hyperparameters (Appendix D) for demonstrating thier robustnesss and the good performance of CobBO for a range of dimensions. Confidence intervals (95%) are computed by repeating 30 and 10 independent experiments for the small and medium-sized functions and the 100-dimensional functions, respectively.

E.1 Small-sized synthetic black-box functions (minimization)

Three additional synthetic 10 dimensional functions [57] are experimented with in Fig. 8, including Ackley over $[-5, 10]^{10}$, Levy over $[-5, 10]^{10}$ and Rastrigin over $[-3, 4]^{10}$. TuRBO is configured the same as in [14], with a batch size of 10 and 5 concurrent trust regions where each has 10 initial points.

Hyper-parameter	Description	Default Value
Θ	The threshold for the number of consecutive fails q_t before changing V_t	60 if $T > 2000$ else 30
α	Increase multiplicative ratio for the coordinate distribution update	2.0
β	Decay multiplicative ratio for the coordinate distribution update	1.1
p	Probability for selecting coordinates with the largest π_t values	0.3
κ_S	The threshold for the virtual clock value K_t before shrinking the coarse trust region Ω_S	30
κ_F	The threshold for the number of consecutive fails q_t before shrinking the fine trust region Ω_F on the fast time scale	6
τ_F	The number of consecutive fails q_t in the fine trust region Ω_F	6
δ	The relative improvement threshold governing the virtual clock update rule	0.1
	Gussian process kernel	Matern 5/2

Table 2: CobBO's hyperparameters configuration for all of the experiments

The other algorithms use 20 initial points. The results are shown in Fig. 8. CobBO shows competitive or better performance. It finds the best optima on Ackley and Levy among all the algorithms and outperforms the others for the difficult Rastrigin function. Notably, BADS is more suitable for low dimensions, as commented in [1]. Its performance is close to CobBO except for Rastrigin.

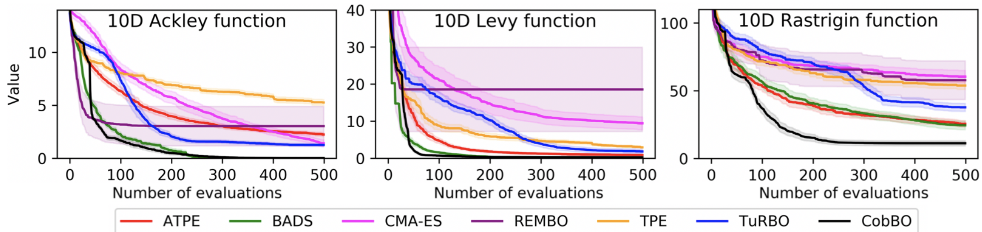

Figure 8: Low dimensional problems: Ackley (left), Levy (middle) and Rastrigin (right)

In Fig. 9 we show that CobBO also optimizes well the Michalewicz function on 10 dimensions, although it has symmetric bumps, where certain subspaces pass through a point in a symmetrical manner and others break it. Other real applications include parameter tuning for recommendation

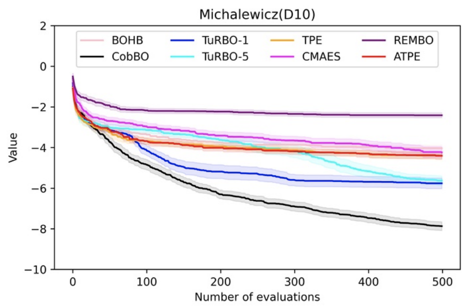

Figure 9: Performance over the low dimensional Michalewicz function with symmetrical and asymmetrical subspaces

systems, database online performance tuning, and simulation based parameter optimization. However, due to deviating from the main study of this paper, we refrain from presenting these results that require elaborated description on the application backgrounds.

E.2 Medium-sized synthetic black-box functions (minimization):

We test three synthetic functions (30 dimensions), including Ackley on $[-5, 10]^{30}$, Levy $[-5, 10]^{30}$, and Rastrigin on $[-3, 4]^{30}$. In addition, we add experiments for an additive function of 36 dimensions,

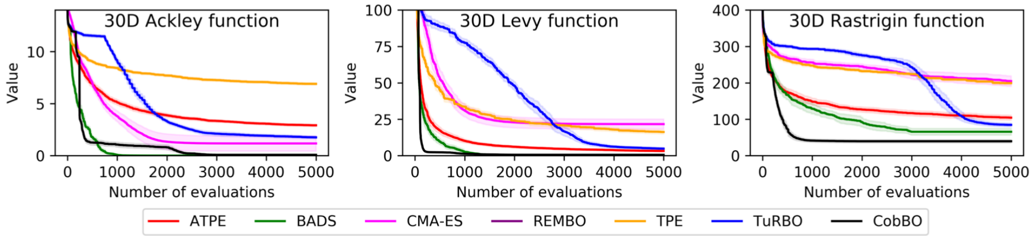

Figure 10: Medium dimensional problems: Ackley (left), Levy (middle) and Rastrigin (right)

defined as $f_{36}(x) = \mathrm{Ackley}(\mathrm{x}_1) + \mathrm{Levy}(\mathrm{x}_2) + \mathrm{Rastrigin}(\mathrm{x}_3) + \mathrm{Hartmann}(\mathrm{x}_4)$, where the first three terms express the same functions over the same domains specified in Section 3.1 of this paper, with the Hartmann function over $[0, 1]^6$. TuRBO is configured identically the same as in Section 3.1, with a batch size of 10 and 5 trust regions with 10 initial points each. The other algorithms use 20 initial points. The results are shown in Fig. 10 and 11, where CobBO shows competitive or better performance compared to all of the methods tested across all of these problems.

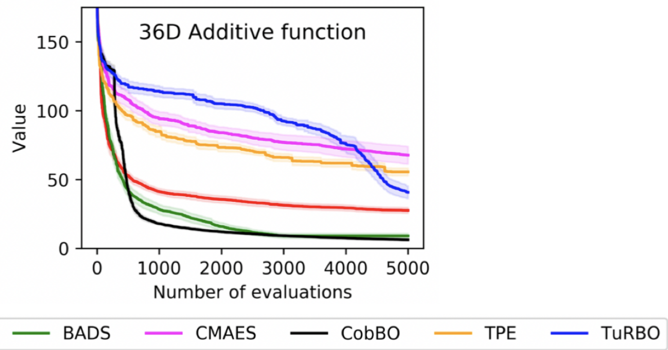

Figure 11: Performance over an additive function of 36 dimensions

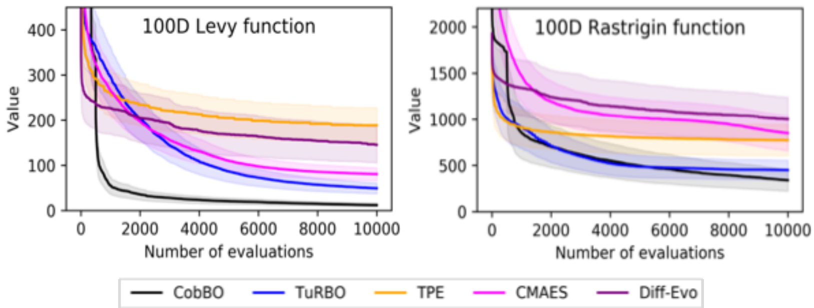

Figure 12: Performance over high dimensional synthetic problems: Levy (left) and Rastrigin (right)

E.3 100 dimensional synthetic black-box functions (minimization):

We minimize the Levy and Rastrigin functions on $[-5, 10]^{100}$ with 500 initial points. TuRBO is configured with 15 trust regions and a batch size of 100. As commented in [14], these two problems are challenging and have no redundant dimensions. Fig. 12 (left) shows that CobBO can greatly reduce the trial complexity. For Levy, it finds solutions close to the final one within $1,000$ trials, and eventually reach the best solution among all the algorithms tested. For Rastrigin, within $2,000$ trials CobBO and TuRBO surpass the final solutions of all the other methods, eventually with a large margin.

F Comparison to LineBO

Although sharing some common basic ideas, LineBO [29] reduces the acquisition maximization cost by restricting on a line but does not reduce the expensive computational costs of the GP regression in the full space. Fig. 13 shows that LineBO is significantly outperformed by CobBO, through a typical example in $D = 10$ (Ackely). In another typical experiment of $D = 30$ and a query budget of 5000,

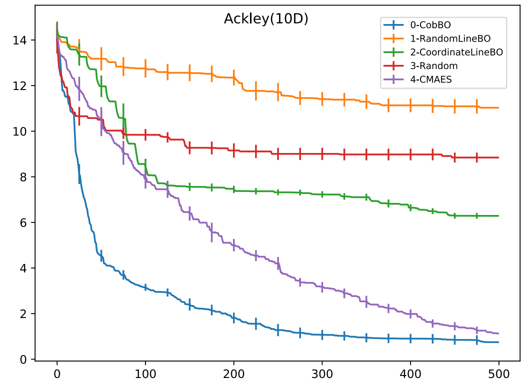

Figure 13: A typical example of CobBO outperforming different variants of LineBO

CobBO reached 0.12 and LineBO reached 7.6.

G Comparison to ALEBO

ALEBO [32] is designed for high-dimensional (large D) problems with low intrinsic dimensions (small d). For comparison, we first test CobBO using exactly the same setting as in [32] for Hartmann6 with $D = 1000$ dimensions and only $d = 6$ intrinsic dimensions, as shown in Fig. 14. Then, for the general problems without the assumption on low intrinsic dimensions, we test ALEBO on Ackley(10D) in three sets of experiments in Fig. 15, where $D = d = 10$. Since ALEBO algorithm requires to provide a low intrinsic dimension $d < D$, we test $d = 2, 4, 8$ dimensions (i.e., ALEBO-2, ALEBO-4, and ALEBO-8), respectively.

Fig. 14 and 15 show the final results by repeating each experiment 30 times. For the first case, ALEBO indeed outperforms CobBO, since CobBO is not suitable for dimensions larger than a few hundreds. For the second case, ALEBO does not show good performance and is outperformed by CobBO, TurBO and CMAES. Regarding the computation times, it takes 6 to 12 hours for ALEBO and only 3 minutes for CobBO to finish 500 queries for each experiment on our testbed for the second case.

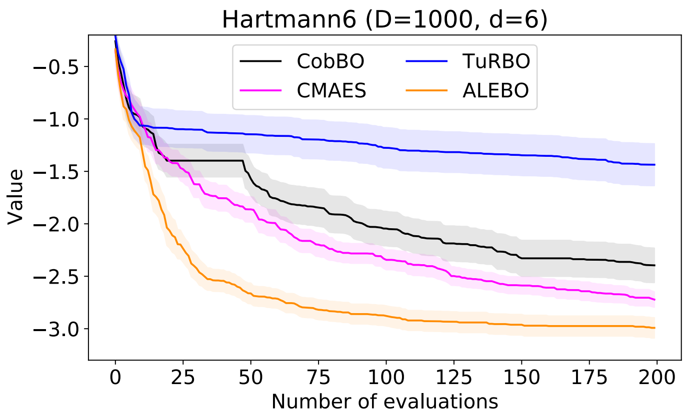

Figure 14: Performance on Hartmann6 ($D = 1000$, $d = 6$)

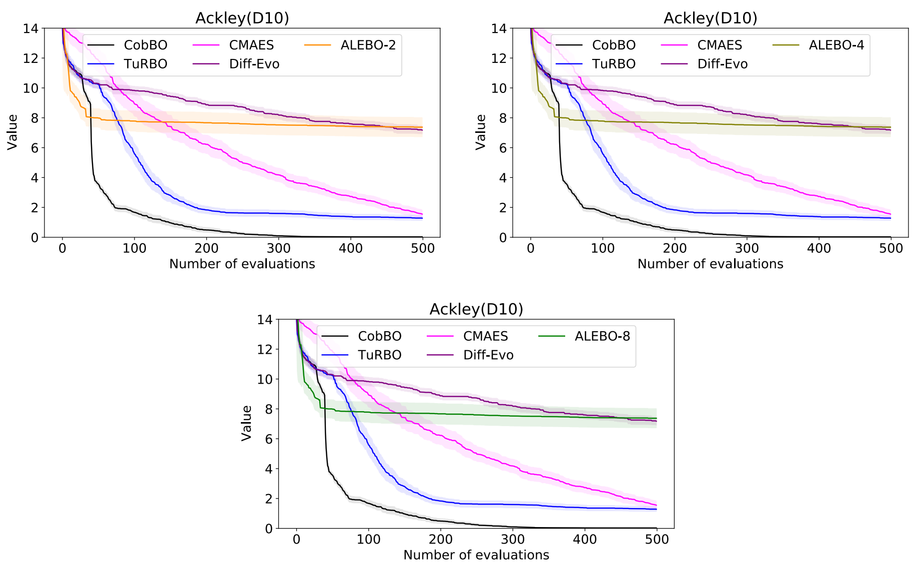

Figure 15: Compare ALEBO and CobBO on Ackley(10D)