Batched High-dimensional Bayesian Optimization via Structural Kernel Learning (Appendix)

Zi Wang [*1] Chengtao Li [*1] Stefanie Jegelka [1] Pushmeet Kohli [2]

1. Add-UCB-DPP-BBO Algorithm

We present four variants of Add-UCB-DPP-BBO in Algorithm 1. The algorithm framework is general in that, one can plug in other acquisition and quality functions other than UCB to get different algorithms.

2. Additional experiments

In this section, we provide more details in our experiments.

2.1. Optimization of the Acquisition Functions

We decompose the acquisition function into M subacquisition functions, one for each part, and optimize those separately. We randomly sample 10000 points in the low dimensional space, and then choose the one with the best value to start gradient descent in the search space (i.e. the range of the box on $R^{|A_m|}$). In practice, we observe this approach optimizes low-dimensional (< 5 dimensions) functions very well. As the number of dimensions grows, the known difficulties of high dimensional BO (and global nonconvex optimization) arise.

2.2. Effectiveness of Decomposition Learning

Recovering Decompositions In Table 4, Table 2 and Table 3, we show three quantities which may imply the quality of the learned decompositions. The first quantity , reported in Table 4, is the Rand Index of the decompositions learned by Gibbs sampling, namely, $\frac{\sum_{i<j\le D} \mathbb{1}_{z_i^g \equiv z_j^g \wedge z_i \equiv z_j} + \sum_{i<j\le D} \mathbb{1}_{z_i^g \neq z_j^g \wedge z_i \neq z_j}}{\binom{D}{2}}$. The second quantity, reported in Table 2, is the probability of two dimensions being correctly grouped together by Gibbs sampling in each iteration of Gibbs sampling after the burn-in period, namely, $\frac{\sum_{i<j\le D} \mathbb{1}_{z_i^g \equiv z_j^g \wedge z_i \equiv z_j}}{\sum_{i<j\le D} \mathbb{1}_{z_i \equiv z_j}}$. The third quantity, reported in Table 3, is the probability of two dimensions being correctly separated by Gibbs sampling in each iteration of Gibbs sampling after the burn-in period, namely, $\frac{\sum_{i<j\le D} \mathbb{1}_{z_i^g \neq z_j^g \wedge z_i \neq z_j}}{\sum_{i<j\le D} \mathbb{1}_{z_i \neq z_j}}$.

Sensitivity Analysis for α Empirically, we found that the quality of the learned decompositions is not very sensitive to the scale of α (see Table 4), because the log data likelihood plays a much more important role than $\log(|A_m| + \alpha)$ when α is less than the total number of dimensions. The reported results correspond to alpha = 1 for all the partitions.

BO for Synthetic Functions We show an example of a 2 dimensional function component in the additive synthetic function in Fig. 1. Because of the numerous local maxima, it is very challenging to achieve the global optimum even for 2 dimensions, let alone maximizing an additive sum of them, only by observing their sum. The full results of the simple and cumulative regrets for the synthetic functions comparing Add-GP-UCB with known additive structure (`Known`), no partitions (`NP`), fully partitioned with one dimension for each group (`FP`) and the following methods of learning partition: Gibbs sampling (`Gibbs`), random sampling the same number of partitions sampled by `Gibbs` and select the one with the highest data likelihood (`PL-1`), random sampling 5 partitions and select the one with the highest data likelihood (`PL-2`) are shown in Fig. 3. The learning was done every 50 iterations, starting from the first iteration. For $D = 20, 30$, it is quite obvious that when a new partition is learned from the newly observed data (e.g. at iteration 100 and 150), the simple regret gets a boost.

BO for Real-world functions In addition to be 14 parameter robot pushing task, we tested on the walker function which returns the walking speed of a three-link planar bipedal walker implemented in Matlab (Westervelt et al., 2007). We tune 25 parameters that may influence the walking speed, including 3 sets of 8 parameters for the ODE solver and 1 parameter specifying the initial velocity of the stance leg. To our knowledge, this function does not

*Equal contribution [1]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Massachusetts, USA [2]DeepMind, London, UK. Correspondence to: Zi Wang <ziw@csail.mit.edu>, Chengtao Li <ctli@mit.edu>, Stefanie Jegelka <stefje@csail.mit.edu>, Pushmeet Kohli <pushmeet@google.com>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

Algorithm 1 Add-UCB-DPP-BBO Variants

```
Input: $\mathcal{X}$, $N_{init}$, $N_{cyc}$, $T$, $B$, $M$
Observe function values of $N_{init}$ points chosen randomly from $\mathcal{X}$
Get the initial decomposition of feature space via Gibbs sampling and get corresponding $\mathcal{X}_m$'s
for $t = 1$ to $T$ do
  if $(t \mod N_{cyc} = 0)$ then
    Learn the decomposition via Gibbs sampling and get corresponding $\mathcal{X}_m$'s
  end if
  Choose $x_0$ by maximizing UCB (acquisition function) for each group and combine them
  for $m = 1$ to $M$ do
    Compute $(\mathcal{R}_t^{(m)})^+$ and $\mathbf{K}_{(t-1)B+1}^{(m)}$
    Sample $\{x_i^{(m)}\}_{i\in[B-1]} \subseteq \mathcal{X}_m$ via PE or DPP with kernel $\mathbf{K}_{(t-1)B+1}^{(m)}$
  end for
  Combine $\{x_i^{(m)}\}_{i\in[B-1],m\in[M]}$ either randomly or by maximizing UCB (quality function) without replacement to get $\{x_i\}_{i\in[B-1]}$
  Observe (noisy) function values for $\{x_i\}$ for $i \in \{0, \ldots, B-1\}$.
end for
```

Table 1. Rand Index of the decompositions computed by Gibbs sampling.

D \ N	50	150	250	350	450
5	0.85 ± 0.20	0.83 ± 0.23	0.71 ± 0.18	0.68 ± 0.16	0.66 ± 0.18
10	0.78 ± 0.06	0.85 ± 0.08	0.86 ± 0.10	0.89 ± 0.12	0.95 ± 0.06
20	0.88 ± 0.02	0.88 ± 0.02	0.89 ± 0.02	0.92 ± 0.02	0.95 ± 0.04
50	0.95 ± 0.01	0.95 ± 0.01	0.95 ± 0.01	0.95 ± 0.01	0.95 ± 0.01
100	0.98 ± 0.00	0.97 ± 0.00	0.97 ± 0.00	0.97 ± 0.00	0.97 ± 0.00

Table 2. Empirical posterior of any two dimensions correctly being grouped together by Gibbs sampling.

d_x \ N	50	150	250	350	450
5	0.81 ± 0.28	0.91 ± 0.19	1.00 ± 0.03	0.97 ± 0.08	1.00 ± 0.00
10	0.21 ± 0.13	0.54 ± 0.25	0.68 ± 0.25	0.81 ± 0.27	0.93 ± 0.15
20	0.06 ± 0.06	0.11 ± 0.08	0.20 ± 0.12	0.43 ± 0.17	0.71 ± 0.22
50	0.02 ± 0.03	0.02 ± 0.02	0.03 ± 0.03	0.04 ± 0.03	0.06 ± 0.04
100	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	0.02 ± 0.02

Table 3. Empirical posterior of any two dimensions correctly being separated by Gibbs sampling.

d_x \ N	50	150	250	350	450
2	0.30 ± 0.46	0.30 ± 0.46	0.90 ± 0.30	0.90 ± 0.30	1.00 ± 0.00
5	0.87 ± 0.17	0.80 ± 0.27	0.60 ± 0.32	0.55 ± 0.29	0.50 ± 0.34
10	0.88 ± 0.05	0.89 ± 0.06	0.89 ± 0.07	0.91 ± 0.08	0.94 ± 0.07
20	0.94 ± 0.02	0.94 ± 0.02	0.94 ± 0.02	0.95 ± 0.02	0.97 ± 0.02
50	0.98 ± 0.00	0.98 ± 0.00	0.98 ± 0.01	0.98 ± 0.00	0.98 ± 0.01
100	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00

have an additive structure. The regrets of each decomposition learning methods are shown in Fig. 2. In addition to `Gibbs`, we test learning decomposition via constrained Gibbs sampling (`Gibbs-L`), where the maximum size of each group of dimensions does not exceed 2. Because the function does not have additive structure, `Gibbs` performed poorly since it groups together many dimensions of the input. As a result, its performance is similar to that of no partition (`NP`). However, `Gibbs-L` appears to learn a good decomposition with the group size limit, and manages to achieve a slightly lower regret than other methods. `Gibbs`, `PL-1`, `PL-2` and `FP` all performed relatively well

Table 4. Rand Index of the decompositions learned by Gibbs sampling for different values of α.

α \ N	50	150	250	350	450
0.2	0.87811 ± 0.019002	0.90126 ± 0.022394	0.95284 ± 0.047111	0.98811 ± 0.02602	0.98811 ± 0.026322
0.5	0.88211 ± 0.019893	0.90305 ± 0.024574	0.95295 ± 0.046232	0.98947 ± 0.025872	0.99505 ± 0.013881
1	0.88211 ± 0.016947	0.90326 ± 0.024935	0.95305 ± 0.043878	0.98558 ± 0.034779	0.98053 ± 0.035843
2	0.88084 ± 0.016972	0.9 ± 0.023489	0.95463 ± 0.042968	0.97989 ± 0.038818	0.98832 ± 0.023592
5	0.88337 ± 0.015784	0.90158 ± 0.02203	0.96126 ± 0.037045	0.98716 ± 0.030949	0.99316 ± 0.015491

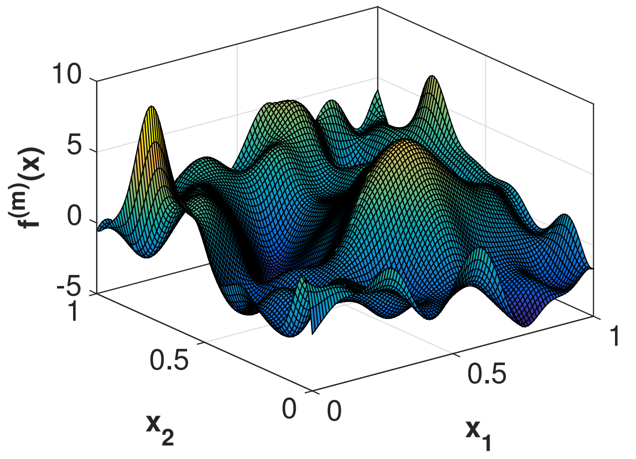

Figure 1. An example of a 2 dimensional function component of the synthetic function.

in for this function, indicating that using the additive structure may benefit the BO procedure even if the function itself is not additive.

2.3. Diverse Batch Sampling

In Fig. 4, we show the full results of the simple and the cumulative regrets on the synthetic functions described in Section 5.2 of the paper.

References

Westervelt, Eric R, Grizzle, Jessy W, Chevallereau, Christine, Choi, Jun Ho, and Morris, Benjamin. *Feedback control of dynamic bipedal robot locomotion*, volume 28. CRC press, 2007.

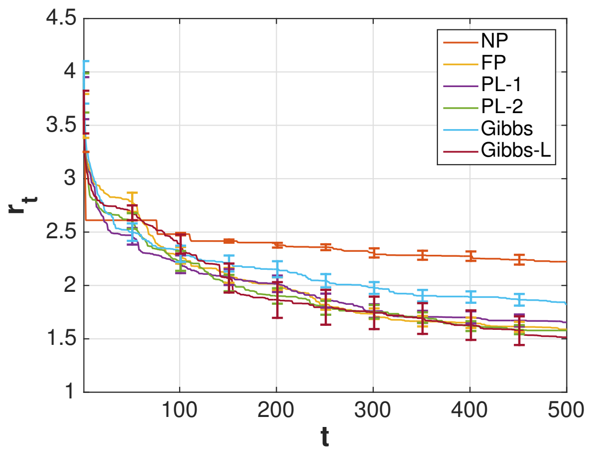

Figure 2. Simple regret of tuning the 25 parameters for optimizing the walking speed of a bipedal robot. We use the vanilla Gibbs sampling algorithm (`Gibbs`) and a Gibbs sampling algorithm with partition size limit set to be 2 (`Gibbs-L`) to compare with partial learning (`PL-1`, `PL-2`), no partitions (`NP`), and fully partitioned (`FP`). `Gibbs-L` performed slightly better than `PL-2` and `FP`. This function does not have an additive structure, and as a result, `Gibbs` does not perform well for this function because the sizes of the groups it learned tend to be large .

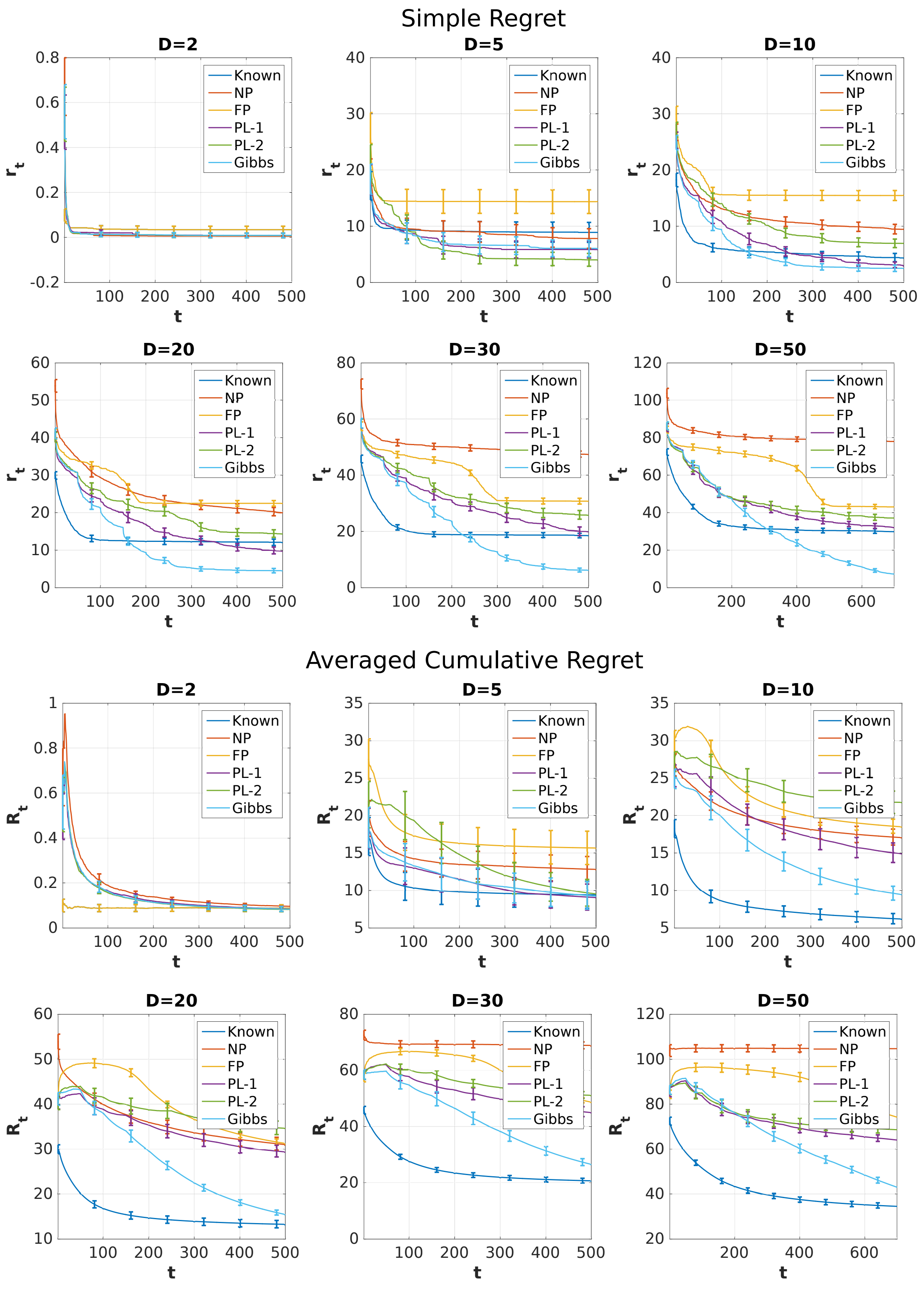

Figure 3. The simple regrets (r_t) and the averaged cumulative regrets (R_t) and for `Known` (ground truth partition is given), `Gibbs` (using Gibbs sampling to learn the partition), `PL−1` (randomly sample the same number of partitions sampled by `Gibbs` and select the one with highest data likelihood), PL-2 (randomly sample 5 partitions and select the one with highest data likelihood), `FP` (fully partitioned, each group with one dimension) and `NP` (no partition) on 10, 20, 50 dimensional functions. `Gibbs` achieved comparable results to `Known`. Comparing `PL−1` and `PL−2` we can see that sampling more partitions did help to find a better partition. But a more principled way of learning partition using `Gibbs` can achieve much better performance than `PL−1` and `PL−2`.

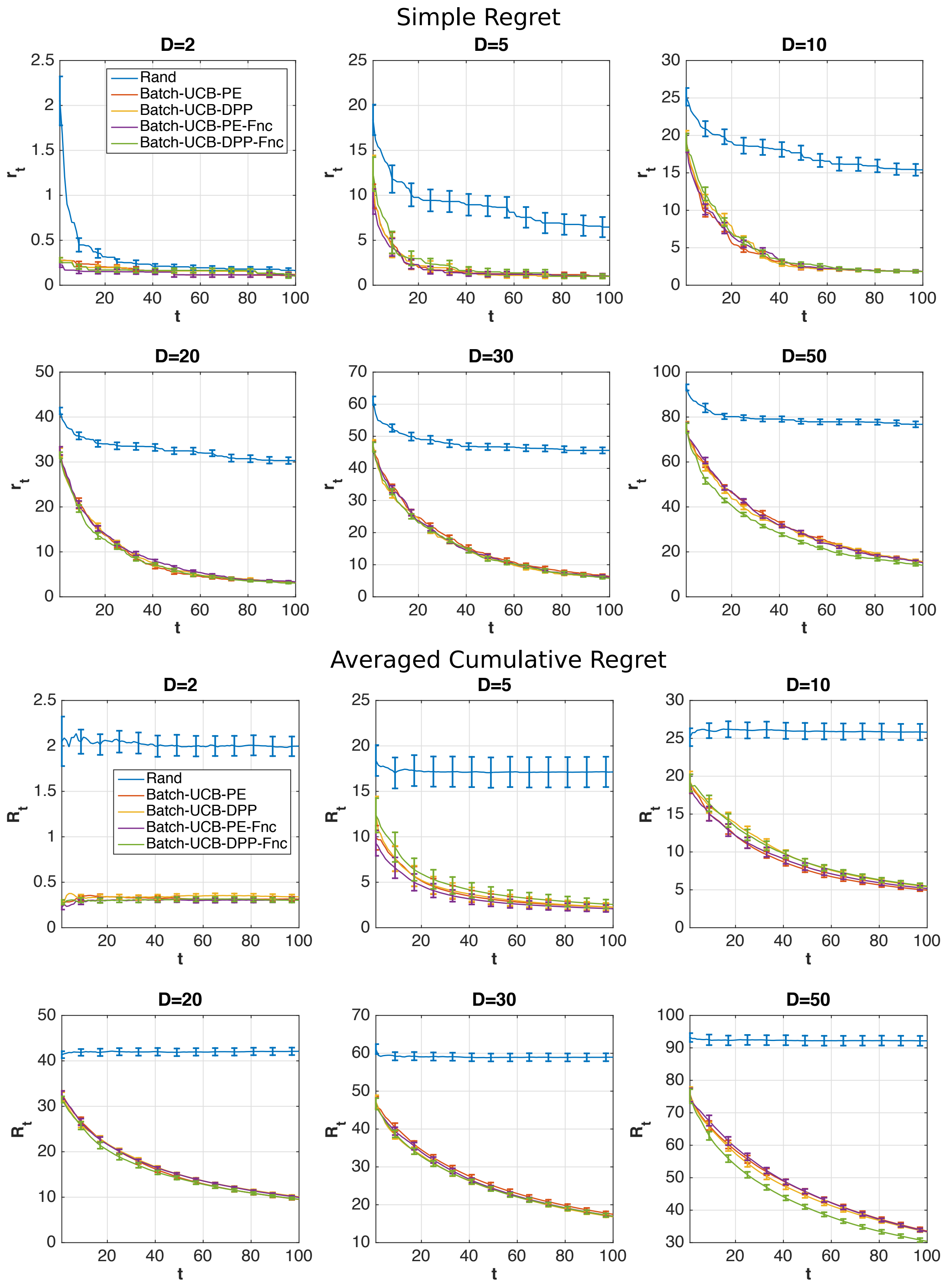

Figure 4. The simple regrets (r_t) and the averaged cumulative regrets (R_t) on synthetic functions with various dimensions when the ground truth partition is known. Four batch sampling methods (`Batch-UCB-PE`, `Batch-UCB-DPP`, `Batch-UCB-PE-Fnc` and `Batch-UCB-DPP-Fnc`) perform comparably well and outperform random sampling (`Rand`) by a large gap.