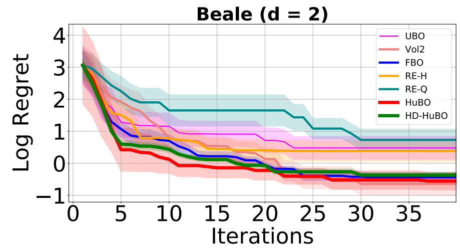
We thank the reviewers for the positive and constructive feedback. Below we respond to their questions.

R1 + R4. "**The regret curves have a very tight confidence bound, starting from the very first iterations. Shouldn't it vary more? Variances are almost same across iterations?**" For the error bars (or variances), we use the standard error: Std. Err $=$ Std. Dev$/\sqrt{n}$, n being the number of runs. In our case, $n = 15$. Division by $\sqrt{n}$ definitely makes the error bars look smaller. We have now included an example case for Beale function using standard deviation for error bars without diving by $\sqrt{n}$. See the plot here. We also confirm that our initial search spaces are randomly placed across different runs and therefore, we see variances even in the beginning.

We would like to emphasise that, different from traditional BO algorithms with fixed search space where error bars (or variances) of regret curves tend to get tighter over time, in the context of unbounded search space where the search space is being expanded over time, error bars do not always have this property, they may even become higher over time till the search spaces have not contained the global optimum. This trend can be seen for many unbounded search space methods such as Ha et al [8] and Vu et al [17] in our references.



R1. "Optimization of the acq. function inside H_t isn't mentioned anywhere?" We optimise acq. function for each hypercube in H_t and then take the overall maximum across all hypercubes. We did mention this in detail in our supp. material (see section "On the computational effectiveness").

R1. "The performance of HD-HuBO against HuBO in high dimensions". In high dimension, HD-HuBO works better than HuBO just because of the acquisition function optimization step *in practice*. Up to a large value of t, the volume of search space H_t for HD-HuBO remains much smaller than the volume of search space X_t for HuBO. For example, assuming $\alpha = -1$, $\lambda = 1$ and the dimension $d = 50$, HD-HuBO at iteration t only uses t small hypercubes with size of 10% of the initial search space $[0, 1]^{50}$. Considering $t = 1000$, the volume of H_t is at most 1000×0.1^{50} which is at least $(1 + \sum_{j=1}^{1000} j^{\alpha})^{50}/(1000 \times 0.1^{50}) \approxeq 5.66 \times 10^{90}$ times smaller compared to the volume of X_t of HuBO. However, despite this, HD-HuBO still attains a sub-linear convergence.

R1 + R2 + R3. "On the comparison with GP-UCB". To see whether our method does better than a BO method using a large, fixed search space, we compared our HuBO against a GP-UCB algorithm with search domain: $[-100, 100]^5$ for the optimization of 5-dim Levy function. After 200 iters, the smallest function value found by the GP-UCB and our HuBO were 23.10 and 2.21 respectively - a clear evidence in favor of our method. We will add these to the paper.

R1 + R3 + R4. "Drawback of setting $\alpha = -1$; X_0 being far from x^*?" $\alpha = -1$ and X_0 being far from x^* make our algorithms reach to x^* slowly. However, in practice, the translation mechanism of our algorithms permits them to jump faster toward the promising regions. We already had results shown in Figure 3 of our Sup. Material where we used $\alpha = -1$ and set X_0 to be only 2% of the initial search space. Our algorithms clearly outperformed all baselines.

R2. "Why we do not start from a large domain?". For this approach, the crucial problem is "how large a compact search space should be set so that x^* belongs almost surely to the search domain"? Without any prior knowledge, we should set the domain as large as possible. We consider two cases. **Case 1**: *Using a fixed search space*. In section "Additional Results" of our Supp. Material, we already showed using GP-UCB and EI algorithms that the larger the fixed search space, the slower is the convergence. Further, we have also compared our HuBO with GP-UCB with a large search space $[-100, 100]^5$ and performed better. See our detail answer above in lines 27-30 in this rebuttal. **Case 2**: *Successively cutting down the search space*. One strategy may be to use confidence bounds UCB and LCB to cut the search space down to a new space S_t as in the algorithm branch and bound of Nando de Freitas et al (ICML 2012): $S_t = \{x | \mu_t(x) + \sqrt{\beta_t}\sigma_t(x) > \sup \mu_t(x) - \sqrt{\beta_t}\sigma_t(x)\}$. However, S_t is usually not compact and expensive to compute in high dimensions. Further, x^* only belongs to S_t with probability $1 - \delta$, not probability 1, and when cutting the search space successively, it is difficult to achieve a significant reduction from the initial large search space while maintaining a high $1 - \delta$ across all t. In contrast, our algorithms do not suffer from such difficulties, easy to implement and achieve a sub-linear rate of convergence.

R3. "The dependence of the regret bound on X_0 and l_h". Theorem 4 is our main result providing the regret bound for HD-HuBO in terms of T ignoring all variables that are not the function of T. However, we can see the regret bound's explicit dependence on X_0 (via A) and l_h (via β_t) through Lemma 10 in Supp. material.

R4. "**On using small** α". We do not see a small α as a limitation. A small α is meant to slow the search space expansion rate and in fact becomes beneficial once the search space contains the global optimum. As seen from Theorem 2, our HuBO algorithm achieves a sub-linear regret $\mathcal{O}^*(T^{((\alpha+1)d+1)/2})$ (e.g. for SE kernel) implying that the smaller the α, the tighter is the regret provided $-1 \leq \alpha < -1 + 1/d$. We note that our algorithm is the only one to guarantee an exact convergence and further with a sub-linear convergence rate despite such small α values.