Black Box Optimization via a Bayesian-Optimized Genetic Algorithm

John Karro karro@google.com
Greg Kochanski gpk@google.com
Daniel Golovin dgg@google.com
Google Research, Pittsburgh, PA 15206

Abstract

We present a simple and robust optimization algorithm, related to genetic algorithms and with analogies to the popular CMA–ES search algorithm, that serves as a cheap alternative to Bayesian Optimization. The algorithm is robust against both monotonic transforms of the objective function value and affine transformations of the feasible region. It is fast and easy to implement, and has performance comparable to CMA–ES on a suite of benchmarks while spending less CPU in the optimization algorithm, and can exhibit better overall performance than Bayesian Optimization when the objective function is cheap.

1 Introduction

The goal of black box optimization is to locate the optimum (throughout this paper, the minimum) of an objective function $f(x)$ with as little total computation as possible. When evaluations of $f(x)$ are expensive, Bayesian optimization (BO) is typically preferred because it gives good accuracy with few evaluations (c.f., [17, 13, 19, 7, 1]). However the cost of each new suggestion increases as $O(N^2)$ or $O(N^3)$ in the number of trials, and eventually dominates the cost of calculating $f(x)$.

When evaluations of $f(x)$ are cheap relative to the BO suggestion cost (e.g., when N is sufficiently large), it can be more effective to employ a faster suggestion method, even though it may make smaller improvements per evaluation of $f(x)$. One option is to use Bayesian neural networks to model $f(x)$ within BO (e.g., Snoek et al. [14]), however that approach seems to require problem-specific tuning [16] and is thus not ideal for black box optimization. Another set of algorithms use search strategies such as the popular CMA–ES [4, 8, 5], which maintains a estimate of where unusually small values of $f(x)$ are likely to be found, expressed as a multivariate normal distribution. At each step, it generates λ samples from that distribution, and finds the best $K \leq \lambda$ samples, which are then used to update the parameters of the distribution.

Here we describe the Linear Combination Swarm approach that the Vizier black-box optimization tool [3], uses when $N \geq 1000$. It is a variant of a *Contractive Mapping Genetic Algorithm* ([10], [12] and references therein), with analogies to CMA–ES and *Estimation of distribution algorithms* [9], though the probability distribution is maintained implicitly.

This Linear Combination Swarm algorithm relaxes CMA–ES's implicit assumption of a multivariate normal distribution [4], while preserving its robustness against affine transformations of x, and its dependence only on the ranking of $f(x)$. The former matters because it makes the algorithm less sensitive to the condition number of the optimization problem; the latter matters because it makes the algorithm less sensitive to the nonlinearities of the objective function.

2 Algorithm Design and Properties

The algorithm maintains the set S of the K_{best} best trials seen so far, as well as the entire history of trials. At each iteration, it randomly selects among three different sampling strategies — *uniform random search*, *ball sampling*, and *linear combination sampling*.

Uniform random sampling is non-adaptive and thus robust to pathological objective functions. Ball sampling randomly picks uniformly from the surface of a sphere centered on the best point of S whose radius is chosen

OPTML 2017: 10th NIPS Workshop on *Optimization for Machine Learning* (NIPS 2017).

from an power law $1/r$ distribution (e.g., [11]).[1]. Linear combination sampling takes two random points $\{x_a, x_b\}$ from the entire history (with heavier weight given to the K_{best} best points), and samples a linear combination of them, with weight α drawn from a normal distribution.[2]

We motivate these sampling strategies by considering sufficient conditions for rapid convergence to the global optimum under some weak assumptions. Let S_t denote the value of S at the end of the t^{th} iteration of the main loop in Algorithm 1. Let $\mathcal{L}^-(S)$ denote the feasible points where $f(x) \leq \max_{x \in S} f(x)$, and let $\mu(S)$ denote the volume of $\mathcal{L}^-(S)$. Then a sufficient condition for convergence to an ϵ-approximate global optimum (i.e., a point with optimality gap $f(x) - \min_{x'} \{f(x')\} \leq \epsilon$) is that for all t there exists $\delta, \eta > 0$ such that either S_t already contains such an approximate global optimum, or else:

$$\mathbf{Pr}[\mu(S_{t+1}) \leq (1-\eta)\,\mu(S_t)] \geq \delta. \tag{1}$$

If this condition holds, we may apply Azuma's inequality to the potential function $\Phi(t) = \log(\mu(S_t))$ to derive high probability bounds on the shrinkage of $\mu(S_t)$, until it reaches the volume of the sublevel set of an ϵ-approximate global optimum.

Under what conditions will the above convergence criteria hold? Suppose the level sets $\mathcal{L}_v := \{x : f(x) = v\}$ have zero volume for all v.[3] If $f(x)$ is Lipschitz continuous, the union of the regions of attraction around the global optima cannot have arbitrarily small volume. Random sampling guarantees that we will eventually select a trial within that region of attraction. If $\mathcal{L}^-(S_t)$ contains a convex region around the best point in S_t with moderate condition number, then ball–sampling with sufficiently small radius is likely to shrink $t \mapsto \mathcal{L}^-(S_t)$.

If the condition number is very large, linear combination sampling allows the algorithm to meet the convergence condition if the expected overlap between $\mathcal{L}^-(S)$ and lines constructed between pairs of points chosen from S is a positive constant and the sampling distribution for α in Algorithm 1 is chosen appropriately. Note the convergence condition can be almost guaranteed for the linear combination branch alone if $|S'|$ is large and diverse enough that $\{x - x' : x, x' \in S'\}$ spans the feasible space, and σ is chosen to be large enough so that the linear combination branch is an expansive mapping.

It is also worth noting that the linear combination sampling part of the algorithm behaves well if a low-dimensional objective function is lifted into a high-dimensional space, a là [18]; the irrelevant dimensions are ignored and the algorithm converges as if it was operating in the low-dimensional space of important features.

2.1 Additional Properties of the Algorithm

The algorithm is asymptotically robust to affine transformations of the x-coordinates since the linear combination branch is unaffected by affine transformations, and since it operates a fraction $p_l > 0$ of the time. Therefore, one would expect that an optimization with a high condition number would proceed about p_l times as fast as a low-condition number problem (plus any progress made by the ball sampler). Therefore, this algorithm should not slow down dramatically when $\mathcal{L}^-(S)$ is approximately ellipsoidal, even if the condition number is large.

By construction, the algorithm only depends on greater/less than comparisons on $f(x)$. Therefore, it only depends on the ranks of f, and is robust against any monotonic transform of the range of $f(x)$.

3 Empirical Testing

3.1 Methodology

We tested our implementation on six benchmark functions from the Black-Box Optimization Benchmarking (BBOB) competition [6] (see Supplemental Materials for details). The success of each variant of the algorithm on a given function is defined by its *optimality gap*: $g(f, S) = \min_{x \in S}(f(x)) - \min_x(f(x))$, the difference between the best score found by the algorithm and the ideal score. We plot the best optimality gap seen so far relative to the average of random search runs with the same number of trials. Formally, let A_t be the first t

[1] Note that the $1/r$ distribution implies that the ball sampler can pick anywhere in the feasible region, therefore it is technically a global search method in the sense of the "convergence theorem" in [15] and will thus eventually find the global optimum, although that route to convergence may take a long time.

[2] Note the sample may lie outside the line segment joining x_a and x_b if $\alpha \notin [0, 1]$.

[3] This condition can be removed by adding an arbitrary but consistent tie-breaking rule to the comparator on objective values when deciding whether to update S.

Algorithm 1 Linear Combination Swarm Algorithm

p_b, p_l, μ, σ, β_{best}, β_h are hyperparameters; d is the dimension of x.
$H \leftarrow \emptyset$, $S \leftarrow \emptyset$ ▷ H is the set of all selected trials, S is the best $K_{\text{best}} = \lfloor \beta_{\text{best}} d \rfloor$ trials.
for $t = 0, 1, 2, \ldots, \infty$ **do**
  if $\text{rand}() < p_l$ and $|H| \geq 2$ **then** ▷ Genetic Crossover / Linear Combination Sampling
    $S' \leftarrow S \cup \{\text{random sample of } \lfloor \beta_h d \rfloor \text{ trials from } H\}$.
    $\{x_a, x_b\} \leftarrow$ Uniform random sample from S' where $f(x_a) \leq f(x_b)$.
    $\alpha \leftarrow \mathcal{N}(\mu, \sigma)$.
    $c \leftarrow \alpha x_a + (1 - \alpha) x_b$
  else if $\text{rand}() < p_b/(1 - p_l)$ and $|H| \geq 1$ **then** ▷ Genetic Mutation / Ball sampling.
    $r \leftarrow$ Sample random radius R with $\mathbf{Pr}[R = r] \propto 1/r$.
    $c \leftarrow$ Sample uniformly from a sphere centered at $\arg\min_S f(x)$ with random radius r.
  else ▷ Uniform sampling.
    $c \leftarrow$ Choose a point uniformly over the feasible space X.
  end if
  if c is infeasible **then** continue
  $H \leftarrow H \cup \{c\}$ ▷ Accumulate history.
  if $f(c) < \max_{x \in S} f(x)$ **then**
    Insert c into S, and if $|S| > K_{\text{best}}$ then remove the worst trial from S.
  end if
end for

trials selected by the algorithm, and let $U_t^{(r)}$ be a set of t trials selected uniformly at random, independently for each run r. We plot

$$\hat{g}(f, t) = \frac{g(f, A_t)}{\text{Mean}\left(\left\{g(f, U_t^{(r)}) : \text{ runs } r\right\}\right)},$$

versus t, so $\hat{g} < 1$ implies better performance than an average random search.

Figure 1 shows results for three benchmarks functions and a point-wise geometric mean of the six most difficult benchmarks. The x-axis represents the *trial number*, where the trial t is the t^{th} point suggested by the algorithm. (That is, the best point seen after sampling t points.) For the first three columns of the figure, the y-axis represents the normalized optimality gap, averaged over 100 runs of the algorithm.[4] The last column represent the aggregation of all six benchmarks, calculated by taking the geometric mean at each trial number.

Note the trial values returned by any algorithm are monotonically non-increasing with the number of trials (as it always returns the best seen so far). However, we sometimes see positive slopes in these plots because the value is normalized by Random Search– any time random sampling improves more than the algorithm, the normalized value will increase.

Each algorithm has a number of adjustable parameters, which we optimized using the Vizier hyperparameter optimization tool employing BO methods to guide the search. Specifically, for each algorithm we attempted to optimize the objective function

$$F(h) = \text{GeometricMean}\left(\left\{\text{Mean}\left(\{g(f_k, A_{r,k,t}(h)) : \text{ runs } r\}\right) : \text{ benchmarks } k, \text{ budgets } t\right\}\right),$$

where $A_{r,k,t}(h)$ is the set of the first t trials selected by the algorithm during run r of the optimization with hyperparameters h on benchmark function f_k ($k = 1 \ldots 9$)[5]. Here, the budgets t range over $\{1, 2, \ldots, 10^5\}$ except for Gaussian Process Bandits, where they range over $\{1, 2, \ldots, 10^3\}$.

3.2 Results

In Figure 1 we plot the normalized optimality gap (w.r.t. a random search) for our algorithmic variants for three representative benchmarks, and then the aggregate result over the six nontrivial benchmarks (Beale, Branin, Rastrigin, Rosenbrock, Styblinki, SixHumpCamel), showing these in both 4 and 16 dimensions. Each

[4] The runs differ in random seeds and $x-$ and $y-$ shifts of the benchmark function.

[5] Note that because of the average over t, in the common case where the optimality gap does not make it all the way to zero, $F(h)$ rewards the h that become small early, rather than late.

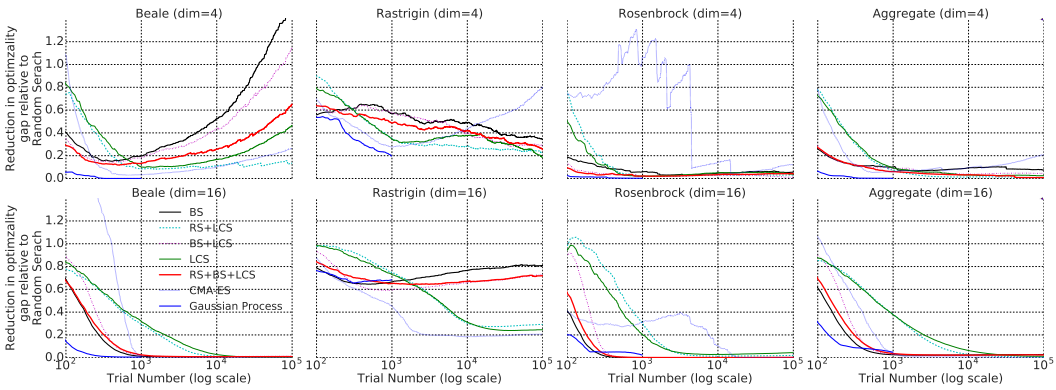

Figure 1: Results for three selected benchmarks and (right) the aggregate over six. LCS= Linear Combination Search, BS= Ball Search, RS= Random Search. The y-axis represent the optimality gap relative to random search; areas in which curves increase indicate that the random sampler is "catching up" on the algorithm. Note that the GP algorithm is run to only 1000 trials due to its computational cost.

curve represents some combination of linear combination search (LCS), ball sampling search (BS), and/or Random Search (RS), with the CMA–ES curve included for reference [4] (see the supplementary material for more details).

The first three columns of Figure 1 represent specific benchmarks, while the fourth column represents the aggregate over the six used benchmarks. Individual plots for all the benchmarks are included in the supplementary materials. A quick examination of those individual benchmark results reveals that there can be considerable variation in algorithm results (both in absolute terms and in terms of their relative ordering): a given algorithm may do quite well in the aggregate while being terrible on certain individual benchmarks. Success on a single function does not mean too much, and even aggregations may not accurately represent black box optimization practice.

If we consider each algorithm in terms of both speed of convergence and final result, we find that the pure linear combination search does consistently well in the second category, finding values as good as almost all competitors with every benchmark, apparently robust to conditions that create pathological behavior for other algorithms. But its rate of convergence can be slow. In contrast, for all its simplicity, the ball sampling technique in isolation does quite well in both aggregates – but it is unreliable on some benchmarks functions. For example we see poor performance on the Rastrigin benchmark, which has many local minima that the algorithm presumably gets stuck in.

In these experiments, combining the techniques provides little gain over Ball Search alone. Our self-tuning suggests that the optimal mixture heavily favors the ball search (using it 60% - 80% of the time), and we have confirmed this with manual experiments. However, if we chose a $F(h)$ that included more functions with multiple local minima, we suspect that the ball sampler would not dominate to the same extent.

CMA–ES has mixed results as well, though it usually does well in the long run. We also compared against a Gaussian Process modeling technique [3], which is based on Srinivas et al. [17]. With its cubic runtime in the number of trials already generated, the GP-based search could not be practically run out to 100,000 trials. It converges rapidly, reaching in 1000 trials values close to those reach by the simpler algorithms in 100,000 trials. However, running those 1000 trials is *considerably* more time consuming. (Running our full set of experiments on the six simple algorithms and six benchmarks, to 100,000 trials in 16 dimensions, required approximately 24 hours; running the GP experiment to 1000 trials takes approximately a week.)

References

[1] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554, 2011.

[2] A. Edelman. *Mathematics of Computation*, 58(197):185–190, January 1993.

[3] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 1487–1495, New York, NY, USA, 2017.

ACM. ISBN 978-1-4503-4887-4. doi: 10.1145/3097983.3098043. URL http://doi.acm.org/10.1145/3097983.3098043.

[4] N. Hansen. The CMA evolution strategy: A tutorial. *CoRR*, abs/1604.00772, 2016. URL http://arxiv.org/abs/1604.00772.

[5] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[6] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. Research Report RR-6829, INRIA, 2009.

[7] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, pages 507–523. Springer, 2011.

[8] C. Igel, T. Suttorp, and N. Hansen. A computational efficent covariance matrix update and a (1+1)-cma for evolution strategies. In *GECCO'06, July 8–12 2006, Seattle, Washington, USA*, pages 453–460, 2006.

[9] P. Larranaga. A review on Estimation of Distribution Algorithms. In *Estimation of distribution algorithms*, pages 57–100. Springer, 2002.

[10] Z. Michaelwicz. *Genetic algorithms + data structures = evolution programs*. Springer-Verlag, Berlin, 1996. ISBN 3-540-60676-9. Third edition. QZ78.618.M53 1996.

[11] I. Pavlyukevich. Lévy flights, non-local search and simulated annealing. *Journal of Computational Physics*, 226(2):1830 – 1844, 2007. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2007.06.008. URL http://www.sciencedirect.com/science/article/pii/S002199910700263X.

[12] L. M. Rios and N. V. Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293, 2013.

[13] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[14] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. M. A. Patwary, Prabhat, and R. P. Adams. Scalable bayesian optimization using deep neural networks. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2171–2180, 2015.

[15] F. J. Solis and R. J.-B. Wets. Minimization by random search techniques. *Mathematics of Operations Research*, 6(1):19–30, 1981. doi: 10.1287/moor.6.1.19. URL https://doi.org/10.1287/moor.6.1.19.

[16] J. T. Springenberg, A. Klein, S. Falkner, and F. Hutter. Bayesian optimization with robust bayesian neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4134–4142. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6117-bayesian-optimization-with-robust-bayesian-neural-networks.pdf.

[17] N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *ICML*, 2010.

[18] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. de Freitas. Bayesian optimization in a billion dimensions via random embeddings. *ArXiv e-prints*, Jan. 2013.

[19] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 370–378, 2016.

4 Supplemental Material

4.1 Benchmark Functions

We tested our implementation on nine benchmark functions, of which eight are from the Black-Box Optimization Benchmarking (BBOB) competition [6], and used six of them for analysis in Figure 1 (we discarded three that didn't challenge any tool).

These are analytic functions with known optimal solutions, several with multiple minima. Specifically, we used the Beale, Branin, Ellipsoidal, Rastrigin, Rosenbrock, Six Hump Camel, Sphere, and Styblinski benchmark functions. We also generated our own "TiltedSphere" function, $y = xAA^Tx^T$, where A is a $d \times d$ matrix whose elements are chosen from a standard normal distribution. This is a variant of the ellipsoidal function except that the axes of the ellipse are tilted; it typically has a modest condition number of $\sim d^{3/2}$ [2].

The Beale, Branin, and Camel functions are defined in $\mathbb{R}^2$, and when we test in $d > 2$, we extend them by adding $\sum_{i=3}^{d} x_i^2$. However, the Rastrigin, Rosenbrock, and Styblinski functions are naturally defined in all $d \geq 2$.

In Figure 1 of the paper we exclude the Sphere, Ellipsoidal, and Tilted Sphere functions as we find them to be generally fairly easy to optimize over, and thus uninformative (but have included them in the supplementary materials plots).

4.2 Algorithm Parameters

In Figure 1 of the paper we present results for several algorithms. The following are details on the algorithms discussed.

- RS: A basic random search algorithm. Each trial consists of picking a random point from the search space and updating the best point seen as necessary.
- BS: The Ball Sampler algorithm, as described in the main paper. At each step the radius of the ball was $r_0 \cdot 2^k$, where $r_0 = 0.016$ and k was a random integer.
- LCS: The linear combination sampler. The sampler kept a pool of $p_s = 5d$ elements (where d is the dimension of the problem), and $p_r = 0.37p_s$ of the pool was reserved for the best seen points, and the remainder was a random sample of the whole history. Points p_1 and p_2 were combined by calculating the point $\alpha p_1 + (1 - \alpha)p_2$, were p_2 was the better-scoring point and α was picked from the normal distribution $N(1.05, \sigma = 1.87)$.
- LCS+ RS: At each pick there was a probability of 0.14 that the algorithm would use a random point. When the linear combination strategy was used, $p_s = 7$, $p_r = 0.18 * p_s$, and α was picked from $N(0.50, \sigma = 1.11)$.
- LCS+ BS: At each pick there was a probability of 0.82 that the ball sampling strategy would be used, and otherwise the linear combination strategy was used. $r = 0.02$, $p_s = 15$, $p_r = 0.40 * p_s$, and α was picked from $N(2.29, \sigma = 0.85)$.
- LCS+ BS+ RS: At each pick there was a probability of 0.16 that a random strategy would be used, and probability of 0.64 that a ball sampling strategy would be used. $r = 0.04$, $p_s = 7$, $p_r = 0.74 * p_s$, and α was picked from $N(2.29, \sigma = 0.84)$.

All numbers were generated by using the Vizier self-tuner, and we were unable to do any better by hand.

5 Refined Plots

In Figures 2 and 3 we present the breakdowns for all examined experimenters discussed in the results section.

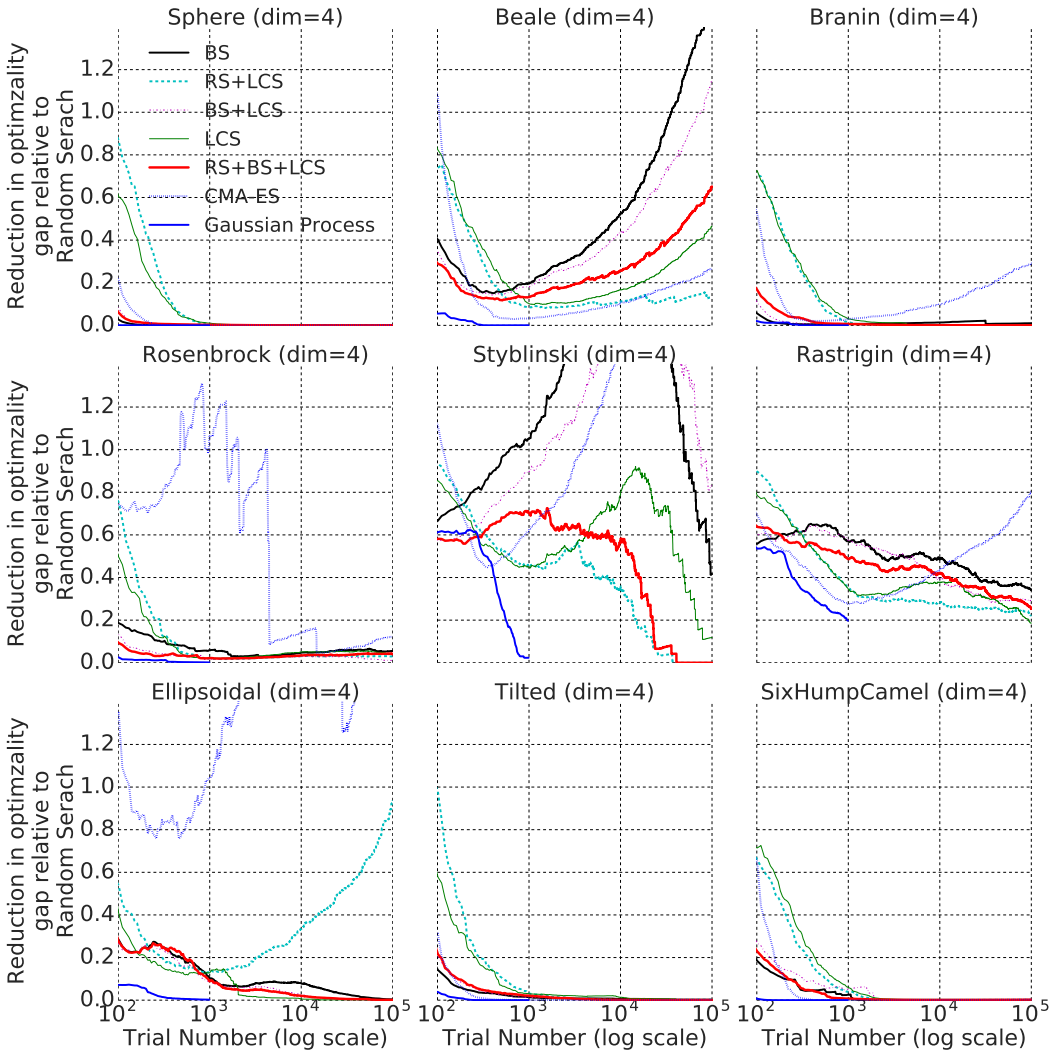

Figure 2: Experimenter-specific plots for 4 dimensions.

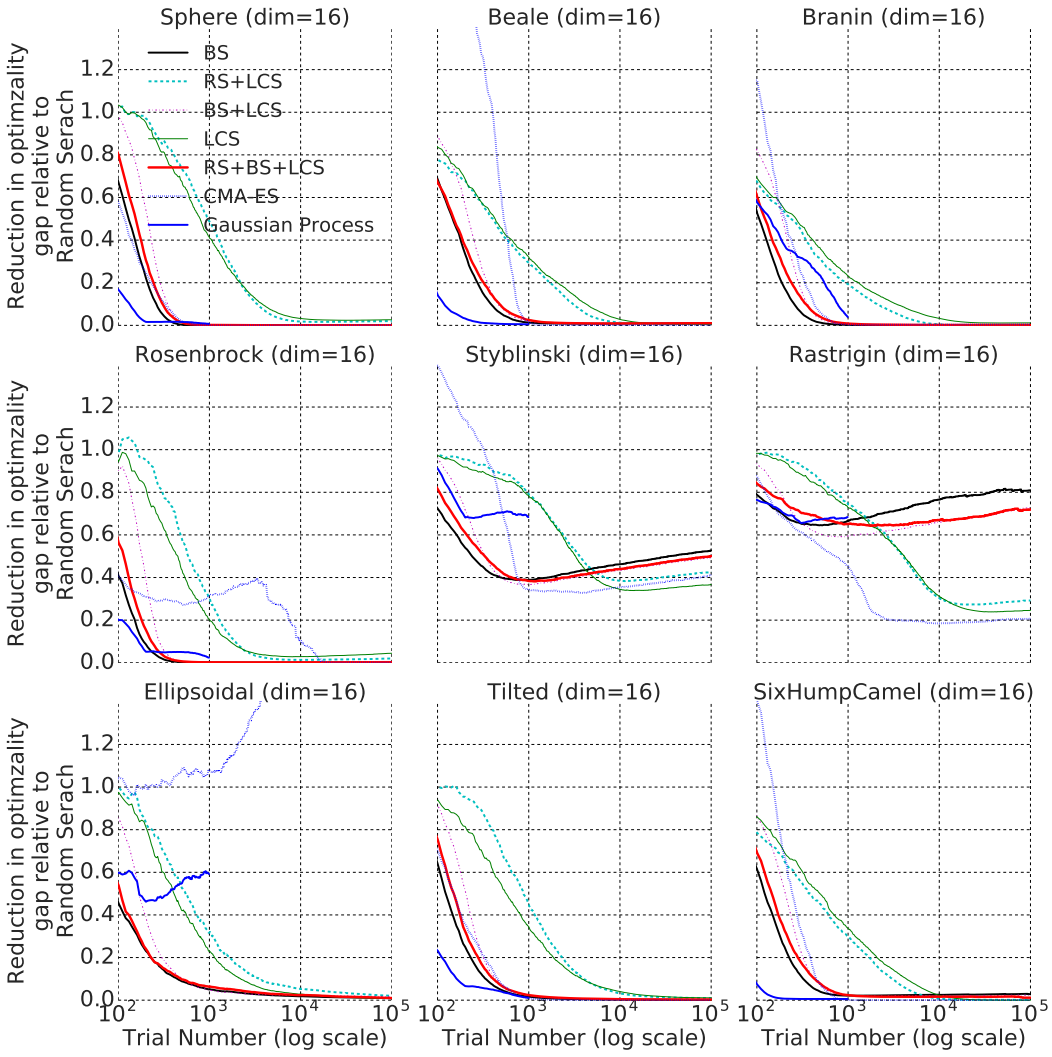

Figure 3: Experimenter-specific plots for 16 dimensions.