An Overview of Stochastic Global Optimization with Multi Level Single Linkage

Mark Olchanyi

Massachusetts Institute of Technology

May 20, 2021

Abstract

This survey begins by providing a general description of the Multi Level Single Linkage (MLSL) algorithm, which is variant of Multistart optimization that focuses on clustering population points in the minimization space based on critical distances between population points where local searches either have or have not been performed. A comparison in terms of objective function call numbers is done between gradient-reliant and gradient-free local search procedures for MLSL. MLSL is then compared in a similar fashion with three competing global optimization methods, CRS, DIRECT, and DIRECT-L. Finally, a note on the lower bound of critical distance parameters is provided in order to assess the utility of MLSL in function spaces where small clusters of population points are required.

1 Introduction

The goal of any *global* optimization problem is to determine the minimum of an objective function $f : \mathbb{R}^n \to \mathbb{R}$. E.g.:

$$x^* = \underset{x \in \mathcal{S}}{\operatorname{argmin}}\ f(x)$$

Where $\mathcal{S}$ is usually a closed and compact set which contains the global minimum of $f(.)$ in its interior. While a plethora of nonlinear optimization methods ranging from gradient-based procedures (such as Newton's method, gradient descent, and conjugate gradient methods), to gradient-free methods (such as Bayesian optimization and Coordinate Descent) exist to robustly solve for local minima (e.g., $x : \nabla f(x) = 0$ and $\nabla^2 f(x)$ is positive definite, where ∇ refers to the gradient and ∇^2 the Hessian of $f(x)$), there are vastly fewer methods to choose from when solving for global minima in both constrained and unconstrained optimization problems. A global minimum, as opposed to a local one, is the true minimum of the objective function. More specifically, global minima arise in non-convex objective functions, and the point of global optimization is to determine these global minima. Global optimization is of interest in many

computational fields where the value in determining the *best* solution to some non-linear problem outweighs the compute time of such a procedure. These problems range from determining the optimal configuration of mirrors in laser equipment design to optimizing the location of urban developments and waste management during environmental risk assessment.

Global methods are generally split into two categories:

Deterministic algorithms refer to optimization schemes which reach a global minimum with certainty, even in the presence of floating-point errors. This however comes with the assumption of indefinite compute times.

Stochastic methods are a generalization of deterministic methods and aim to compute a global minimum via the generation of random variables where randomization can, for example, be over the objective function or over iterates.

Multistart algorithms [1] are a class of stochastic optimization methods introduced in the 1980s in order to solve global optimization problems with only boundary constraints. In essence, Multistart is based on performing local searches on some valid subset of a group of population points which are either randomly or deterministically seeded in a grid-like fashion in the interior of the minimization space and determining the global solution from the set of local optima calculated during the local search phase. Multistart algorithms are an attractive alternative to many deterministic methods when compute time is limited, given that the cardinality of the subset of population points that Multistart performs local searches on can be explicitly tuned. Multi Level Single Linkage (MLSL) is a popular variant of Multistart which aims to cluster the set of population points based on a distance and objective function value metric [2]. MLSL therefore refrains from performing local searches on population points which are within a critical distance measure of a point where a local search has already been applied. This critical distance is iteratively decreased and clustering is repeated until either, in the worst case, all population points have been locally minimized, or some stopping criteria (usually a Bayesian stopping criteria described in Section 3) has been met. The case when no clustering is performed and all population points are naively evaluated in one sweep is referred to as a Pure Random Search (PRS) procedure. PRS therefore serves as an absolute-worst benchmark for MLSL, since PRS has the same, if not better global convergence guarantees, as MLSL (as no population points are discarded in PRS). Therefore, if MLSL requires longer run-times than PRS, PRS should always be chosen.

This survey of MLSL focuses on testing functions of various complexities in terms of dimensionality, numbers of local minima, and relative sizes of basins of attraction. However, only test functions which are **continuous** and **differentiable** are considered in order to assess the performance of MLSL with gradient-based local minimizers compared to gradient-free ones, as MLSL is thought to really shine in terms of compute time when its local solvers have access to gradients. A commonly-used modification of MLSL which uses Low-Discrepancy Sequences (LDS) as population points is used , since it was been stated that the use of LDS improves convergence rates [4]. In this overview, Sobol points are chosen as the LDS sequence [5].

2 Algorithm Overview

The pseudocode for MLSL is outlined below:

Algorithm: MLSL

I1: Generate a set $\mathbf{X}_N$ of N points $\mathbf{x}_i$ uniformly distributed in $\mathcal{B}^d$;
I2: Reduce $\mathbf{X}_N$ to a set $\mathbf{X}_R = \{\mathbf{x}_i : f(\mathbf{x}_i) \leqslant \gamma\}$ where γ is determined such that there are αN points from $\mathbf{X}_N$ with objective function values smaller than γ, for a predetermined α;
I3: Initialize clustering list (can be a vector) $\mathbf{C}$ where the value of element $\mathbf{c}_i \in \mathbf{C}$ indicates the cluster association of $\mathbf{x}_i$, and $\mathcal{L}$ indicates the cluster index;
$k \leftarrow 0$;
while *max iterations not reached* ***or*** *eq. (3) is not satisfied* **do**
  Update $\mathbf{r}_k$;
  for $j \leftarrow 1$ **to** $|\mathbf{X}_R|$ **do**
    for $\mathcal{L} \leftarrow 1$ **to** $\max\{\mathbf{C}\}$ **do**
      if $\exists \mathbf{x}_r \in \{c : \mathbf{c}_r = \mathcal{L}\}$ *s.t.* $\|\mathbf{x}_r - \mathbf{x}_j\| < \boldsymbol{r}_k$ *and* $f(\mathbf{x}_j) \geqslant f(\mathbf{x}_r)$ **then**
        assign $\mathbf{x}_j$ to cluster index $\mathcal{L}$;
      else
        Perform local search with $\mathbf{x}_j$ as seed point and store $f^* \in \mathbf{F}^*$;
        Assign $\mathbf{x}_j$ to cluster index $\max\{\mathbf{C}\} + 1$
      end
    end
  end
  $k \leftarrow k + 1$;
end
return $\min\{\mathbf{F}^*\}$

We denote $|\mathcal{Z}|$ as the cardinality of set $\mathcal{Z}$, $||\mathbf{v}||$ as the 2-norm of a vector $\mathbf{v}$, and $f(.)$ as the objective function of interest. Since MLSL only supports boundary constraints, we implement and denote $\mathcal{B}^d$ as the d-dimensional bounding box which specifies the minimization domain for both the global and local search phases.

For iteration step k and reduced population set size R, the critical distance is defined as:

$$r_k = \left[\frac{m(\mathcal{S})}{\omega_d}\sigma\frac{log(kR)}{kR}\right]^{\frac{1}{d}} \approx \left[\frac{vol(\mathcal{B}^d)}{\omega_d}\sigma\frac{log(kR)}{kR}\right]^{\frac{1}{d}} \tag{1}$$

Where ω_d is the volume of a d-dimensional unit sphere defined as:

$$\omega_d = \frac{\pi^{d/2}}{\Gamma(1 + \frac{d}{2})} \tag{2}$$

$\Gamma(.)$ is the Gamma function. $m(\mathcal{S})$ is the Lebesgue measure of an d-dimensional set $\mathcal{S}$. If our bounding constraint is a hyper-cube $\mathcal{B}^d$, then $m(\mathcal{S}) = 1$. In general, if $\mathcal{B}^d$ is a hyper-rectangle, then $m(\mathcal{S}) = vol(\mathcal{B}^d)$. σ can be thought of as a dimension-normalized scaling parameter for the critical distance. Larger σ values correspond to larger critical distance cutoffs, hence larger cluster sizes in $\mathbf{C}$ and less initiations of local searches during a specific iteration. It has been shown in [9] that if σ is chosen to be greater than 2.0, then the probability of pairing a test point with a cluster index $\in \mathbf{C}$ goes to 1 (e.g. the probability of commencing a local during a iteration of MLSL goes to 0 as iteration numbers approach infinity). Furthermore, if σ is chosen to be greater than 4, then the number of local searches with asymptotically approach a finite value.

This version of MLSL implements a Bayesian stopping criteria described in [6], which is defined as such. If $|\mathbf{F}^*|$ is the number of minima found after S local searches done by iteration k, then MLSL terminates if:

$$F_{exp} > 0.5 + |\mathbf{F}^*| \tag{3}$$

Where:

$$F_{exp} = \frac{|\mathbf{F}^*|(S-1)}{S - |\mathbf{F}^*| - 2}$$

3 Validation of MLSL Implementation

Note: This implementation of MLSL was validated by comparison to the version of MLSL using LDS points in the open-source library Nlopt [8] (specified by *NLOPT_G_MLSL_LDS*) in terms of function calls for the Six-Hump Camel-back function described in Section 4. All critical distance and set reduction parameters were matched ($\sigma = 2.0, \alpha = 0.5$), and the same local minimization procedure was used (VAR2) and a domain tolerance of 10^{-8} for the comparison. This version of MLSL converged and terminated in **307** function calls, and the NLopt version terminated in **292** function calls. This difference was considered acceptable given the heuristic which allows skipping population points in NLopt MLSL which was not included in this implementation.

4 Benchmarking/Comparison of Local Minimizers

This implementation of MLSL is performed on three test functions with varying dimensions, numbers of local minima, and density of minima (e.g. relative sizes of basins of attraction).

The first objective function to be evaluated is the well-known **Six-Hump Camel Function** (6HC) [11], which contains 4 global minima $f^* = -1.03162...$ at $x^* = (\pm 0.08984..., \pm 0.71265...)$, and with chosen bound constraints:

$$\mathcal{B}^2_{6HC} = \begin{bmatrix} -3.0 & -2.0 \\ 3.0 & 2.0 \end{bmatrix}$$

6HC is considered an "easy" problem based on criteria in [7] as its global minimum is relatively far from the local minima, and basins of attraction are large.

The second objective function is the **2D Schubert function** (SHUBERT), which possessed several local minima and many global minima with a value of $f^* = -186.7309....$ The bounds for the Schubert function which are used are:

$$\mathcal{B}^2_{SH} = \begin{bmatrix} -10.0 & -10.0 \\ 10.0 & 10.0 \end{bmatrix}$$

The 2D Shubert function possesses several global minima, which should favor the performance of MLSL as the Bayesian stopping criteria is approached faster when similar minima are encountered. However, the relative density of minima is quite high, and therefore steep gradients in the basins of attraction. Local searches are therefore more likely to fail.

The third objective function is the **7-Dimensional Schwefel function** (SCHWEF), which possessed one global minimum of $f^* = 0.0$ at $x^* = (420.9687, ..., 420.9687)$. The bounds for the Schwefel function which are used are:

$$\mathcal{B}^7_{SHWEF} = \begin{bmatrix} -500.0 & \dots & -500.0 \\ 500.0 & \dots & 500.0 \end{bmatrix}$$

While the topology of the Schwefel function is similar in complexity to SCHUBERT, it only contains 1 global minimum, which increases the likelihood of failure in the global phase.

We compared MLSL with three gradient-based local search algorithms (Method of Moving Averages (MMA), Sequential Least-Squares Quadratic Programming (SLSQP), and rank-2 Shifted limited-memory variable-metric (VAR2) and one gradient-free search algorithm (BOBQYA) to PRS. Comparisons were done is terms of the **number of objective function calls** until relative domain tolerances of 10^{-3} or 10^{-8} were reached in order to avoid overhead and implementation quality costs. All of the local searches were implemented from Nlopt [8]. The results are summarized in Tables 1 2 and 3. All listed runs of both MLSL and PRS converged to the true global minima within their respective tolerances. We did not repeat trials due to the deterministic nature of LDS generation. For all trials, $64 * (dim)^2$ Sobol points were used as the LDS, with an $\mathcal{O}(\mathbf{d^2})$ scaling due to the complexity arguments made in [17]. The r_k parameter was set to $\sigma = 2.0$, and a set reduction ratio $\gamma = \frac{N}{2} = 0.5$ was used. All tests were run on a local machine with a 2.4GHz 8-Core processor.

2D Six-Hump Camel Function

	MLSL (tol: 10^{-3})	PRS (tol: 10^{-3})	MLSL (tol: 10^{-8})	PRS (tol: 10^{-8})
MMA	292	4325	366	9158
SLSQP	304	4185	327	5790
VAR2	294	3110	307	3812
BOBQYA	344	9078	415	13900

Table 1: Comparison of objective function call numbers for MLSL vs PRS with varying local search procedures for the Six-Hump Camel Function.

2D Shubert Function

	MLSL (tol: 10^{-3})	PRS (tol: 10^{-3})	MLSL (tol: 10^{-8})	PRS (tol: 10^{-8})
MMA	1022	8701	1053	9090
SLSQP	FAIL	FAIL	FAIL	FAIL
VAR2	502	7713	679	5708
BOBQYA	797	6867	1181	6867

Table 2: Comparison of objective function call numbers for MLSL vs PRS with varying local search procedures for the Shubert Function.

7D Schwefel Function

	MLSL (tol: 10^{-3})	PRS (tol: 10^{-3})	MLSL (tol: 10^{-8})	PRS (tol: 10^{-8})
MMA	8014	46020	12691	86815
SLSQP	FAIL	FAIL	FAIL	FAIL
VAR2	6396	37195	7941	52701
BOBQYA	53644	469862	86269	758454

Table 3: Comparison of objective function call numbers for MLSL vs PRS with varying local search procedures for the 7D Schwefel Function.

A FAIL cell in the table indicates that the MLSL variant failed to converge to the true global minimum, even after the Sobol sequence number was artificially increased by up to a factor of 10.

While performance is similar amongst gradient-dependent local search methods, it can be seen that as the complexity of the objective function goes up (either in dimension or by the criteria in [11]), the discrepancy in the number of function calls between MLSL with a gradient-free

local minimizer and a gradient-based one grows (with gradient-free local minimizers requiring more objective function evaluations in order to converge to the global minimum).

This discrepancy noticeably holds true for increases in dimensionality of the objective function. We computed the function calls for MLSL with MMA,VAR2, and BOBQYA (and left out SLSQP due to failing to evaluate the true minimum) local optimizers, as well as PRS with VAR2 as a local optimizer on SCHWEF with varying dimensions (1D-6D). The reason we chose to perform PRS with VAR2 is due to it consistently taking less function calls to evaluate the minimum on the three test problems. Figure 1 shows the function call numbers for the 5 algorithms with respect to the dimensionality of the Schwefel function. All LDS and r_k parameters were kept the same as above, and a domain tolerance of 10^{-3} was considered.

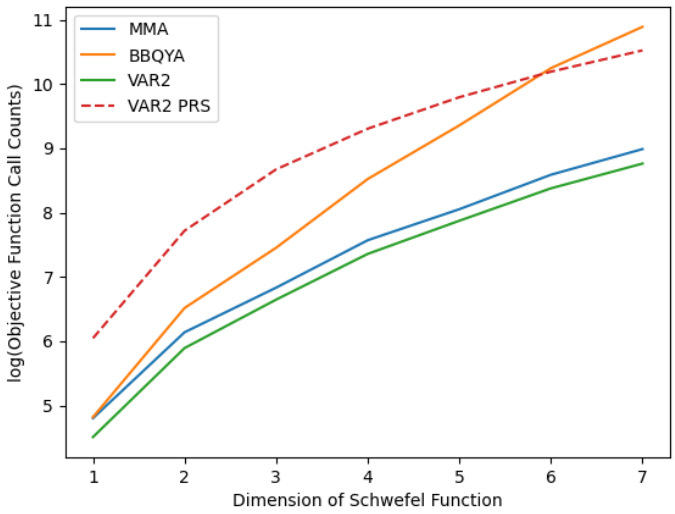

Figure 1: Benchmarked compute time of MLSL (red) compared PRS (blue) while varying σ and retaining the set reduction factor $\gamma = 0.5$.

Figure 1 shows that while there is only a marginal performance difference in function call numbers between MLSL with different gradient-based local solvers (VAR2 and MMA), MLSL with the gradient-free solver (BBQYA) requires substantially more function calls to converge to the global minimum. This discrepancy increases with dimension, and a rough order estimate of this increase for the specified dimensions by inspection of Figure 1 is $\mathcal{O}(\exp\{\mathbf{c} \times \mathbf{dim}\})$ where c is some positive undetermined constant. In fact, for the Schwefel test function, we notice that PRS/VAR2 takes less function calls to compute the global minimum than MLSL/BBQYA for 6 and 7 dimensions, hence making PRS more computationally favorable algorithm in such a setup.

Of note is that past 8 dimensions, MLSL with any of the given local solvers failed to solve

the correct global minimum within the given domain tolerance. A similar MLSL failure was reported in [12] for Rastrigin functions with dimensions higher than 3.

5 Theoretical Lower Bound of Utility for $\mathbf{r_k}$ Parameters

The derivations from [10] give us a minimum distance between pairs of N Sobol points generated in $\mathcal{B}^d$ as:

$$d_{min} \geqslant vol(\mathcal{B}^d)\frac{1}{2}\sqrt{d}N^{-1} \tag{4}$$

We can therefore see from (1) that if the critical distance r_k is chosen either via either a small enough *sigma* parameter or a small γ (e.g. a higher reduction of the LDS set), then r_k can become smaller than d_{min}, and MLSL will in fact perform worse than PRS as no clustering would ever occur. We can now calculate these σ and γ thresholds with some simple algebra:

For a fixed R:

$$\sigma_{thresh} \approx \left\{\frac{1}{2}\sqrt{d}N^{-1}\right\}^d \times \frac{kR\,\omega_d}{\log(kR)\,vol(\mathcal{B}^d)} \tag{5}$$

And for a fixed σ:

$$\frac{\log(kR_{thresh})}{kR_{thresh}} \approx \left\{\frac{1}{2}\sqrt{d}N^{-1}\right\}^d \frac{\omega_d}{\sigma\,vol(\mathcal{B}^d)} \tag{6}$$

Now let us again consider the 2D 6HC function for its low-dimensional simplicity. If we again use a Sobol sequence as our LDS points, then for a $\mathcal{B}^2$ defined prior, 256 Sobol points in $\mathcal{B}^2$, $\gamma = 0.5$, and a single iteration (k=1) as a simplification, then taking into account that $\omega_b = \frac{\pi}{\Gamma(2)} = \pi$, $vol(\mathcal{B}^2_{6HC}) = 24$ for 6HC, and $d_{min} = \frac{24\sqrt{2}}{256} = 0.0055$, we get:

$$\sigma_{thresh} = \left\{\frac{24 * \sqrt{2}}{256}\right\}^2 \times \frac{128\pi}{\log 128 \times 24} = 0.06 \tag{7}$$

This theoretical result indeed corresponds with the computational result shown in Figure 1. The observed threshold is likely slightly higher than the predicted σ due to the function call overhead when reducing the LDS set and evaluating equality of local minima.

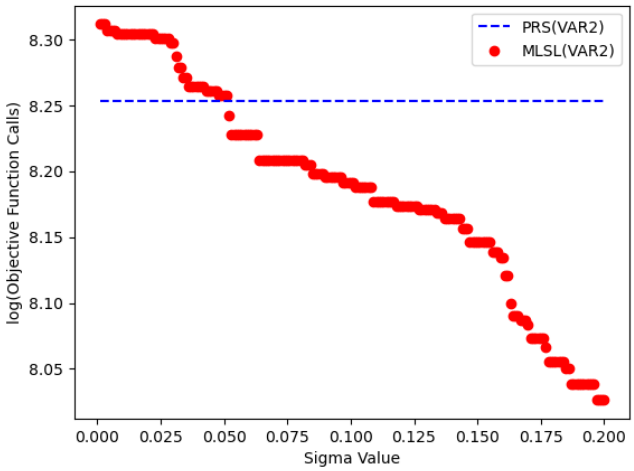

Figure 2: Benchmarked function call numbers MLSL with a VAR2 local minimizer compared to PRS/-VAR2 (blue) while varying σ and retaining the set reduction factor $\gamma = 0.5$.

Similarly, for a set $\sigma = 2.0$ and varying $R = floor(\gamma N)$, we get the R threshold of:

$$\frac{\log(R_{thresh})}{R_{thresh}} \approx \left\{\frac{24 * \sqrt{2}}{256}\right\}^2 \times \frac{\pi}{48} = 0.0011 \tag{8}$$

We can solve this graphically to get $R_{thresh} \approx 3184$. Since this result is clearly larger than $N = 256$, any γ with a reasonably-chosen σ should always perform better than PRS, the computational result shown in Figure 2 confirms this. Even for no set reduction, MLSL is still takes at least an order of magnitude less function calls than PRS to compute with a reasonable σ assignment.

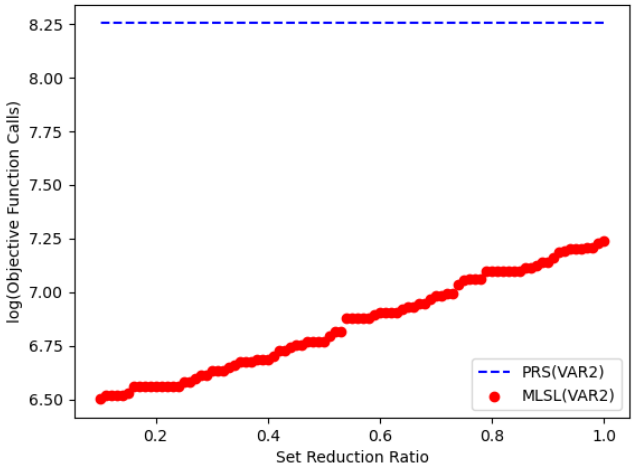

Figure 3: Benchmarked compute time of MLSL (red) compared PRS (blue) while varying the set reduction factor γ and retaining $\sigma = 2.0$.

Therefore, if one were to require small σ values (for example, if basins of attraction of the objective function are small relative to the search space), then one should take care of choosing a value that is high enough for performance to still be better than PRS.

6 Comparison with Competing Algorithms

In this section we compare the performance of MLSL with three global optimization algorithms. Algorithms which solve for global minima via only a bounding-box constraint are explicitly chosen in order to maintain similar minimization spaces.

The first comparison is done Controlled Random Search with Local Mutation (CRSLM) [14]. As with MLSL, CRSLM uniformly fills the bounded domain of the objective function with population points. However, this population set is not clustered like in MLSL, but is contracted by replacing "bad" points (based on objective function values) in the set with updated trial points. These trial points are defined as reflections of a point in the population across a Simplex defined a subset of the population. This replacement is akin to the Nelder Mead Algorithm [13]. Local mutation is performed when trial points fail to replace "bad" points by generating new trial points through "exploration" via coordinate-wise reflections around population points with the smallest objective-function values.

The second comparison is done with the DIviding RECTangles algorithm (DIRECT) [15]. DIRECT does not evaluate a set of population points like MLSL, but works by iteratively partitioning the bounding box search space of the objective function into hyper-rectangles, identifying potentially optimal rectangles, and performing local searches on the midpoints of these rectangles.

The third comparison is done with DIRECT-L, which is a variant of DIRECT that is biased towards local searches. It is therefore expected to perform better on objective functions with fewer local minima.

In Table 4 below, function call counts are given for MLSL, CRSLM, DIRECT, and DIRECT-L implemented with VAR2 as a local minimizer (chosen due to its robust performance from Section 4) with a domain tolerance of 10^{-8}. The evaluation is done on the 2D 6 Hump Camelback Function, the 2D Schuber function, and 2D/8D/10D Schwefel functions.

	MLSL	CRSLM	DIRECT	DIRECT-L
6HC	294	2670	198	187
SCHUBERT	502	11879	59232	19200
SCHWEFEL-2D	363	7235	448	710
SCHWEFEL-8D	FAIL	FAIL	1622217	FAIL
SCHWEFEL-10D	FAIL	FAIL	15180834	FAIL

MLSL outperforms CRSLM in terms of function calls on all three test functions. This is likely due to the overhead of the exploration phase of CRSLM where new candidate points are generated, all of whose function values need to be assessed. MLSL vastly outperforms DIRECT and DIRECT-L on SCHUBERT. This result has in fact been recently shown in [16] as it has to do with sub-optimal hyper-rectangle subdivision for functions with steep basins of attraction. However, DIRECT and DIRECT-L performance is similar for 6HC and low-dimensional SCHWEFEL, and only DIRECT converges to the global minimum for higher-dimensional SCHWEFEL. This is most likely due to the size of the population set needed for Monte Carlo-based global optimization procedures (such as MLSL and CRSLM). This set size requirement is believed to grow anywhere from supra-linearly to exponentially depending on the test problem. Similar increases in set size requirements have been shown for deterministic grids such as Sobol sequences [17].

7 Discussion

It can be seen that some assumptions of the objective function and with tuning of certain parameters, MLSL can be used as a robust stochastic global optimization tool when one values compute time and is working with low-dimensional differentiable test functions. Notably, MLSL performance outshines its competitors when the test-function is low-dimensional, but has steep tight basins of attraction for global minima relative to the search space .

Therefore, MLSL is recommended for users which are optimizing problems which involve such functions and are strapped for compute time (*e.g. quickly optimizing the fastest bike route between MIT and Fenway Park, where one can either take Mass Ave. across the Charles river (the steep global minimum), or take a plethora of drastically longer routes along faraway bridges*).

However, MLSL performance drastically drops and MLSL becomes completely infeasible with an large increase in dimensionality. Table 4 illustrates that for high dimensional SCHWEFEL, neither MLSL, nor other algorithms involving the evaluation of test points is no longer able to converge to the global minimum past a certain dimension (due to the explanation at the end of Section 6). Therefore, for high-dimension objective functions, an alternative global minimization procedure which does not rely on the evaluation of population points, such as DIRECT, should be chosen. Furthermore, if the gradient of the objective function is not available (e.g. if the function is non-differentiable), then the performance of MLSL can also degrade. At high-enough dimensions (but low enough for MLSL to converge), even PRS becomes faster due to the overhead of clustering as seen in Section 4. In this case, either PRS or an algorithm that specifically built to never rely on gradients such as Simulated Annealing [18] or more sophisticated genetic algorithms should be considered.

8 Code

All MLSL code is available for use at: `https://github.com/markolchanyi/MLSL_Julia`.

References

[1] Marti, Rafael. "Multi-Start Methods", *Handbook of Metaheuristics* (2003): 355-368.

[2] Boender, C. G_E., and Ryszard Zieliński. "A sequential Bayesian approach to estimating the dimension of a multinomial distribution." *Banach Center Publications* 16.1 (1985): 37-42.

[3] Boender, C.G.E., Rinnooy Kan, A.H.G., Timmer, G.T. et al. (1982) "A stochastic method for global optimization." Mathematical Programming, 125–140.

[4] Kucherenko, Sergei and Sytsko, Yury. (2005) "Application of deterministic low-discrepancy sequences in global optimization," *Computational Optimization and Applications."* vol. 30, pp. 297-318.

[5] Sobol,I.M. (1967), "Distribution of points in a cube and approximate evaluation of integrals". *Zh. Vych. Mat. Mat. Fiz.* vol. 7, pp. 784–802.

[6] C.G.E. Boender, (1984) "The Generalized Multinomial Distribution: A Bayesian Analysis and Applications", Ph.D. Dissertation, Erasmus Universiteit Rotterdam, Centrum voor Wiskunde en Informatica: Amsterdam

[7] A. Torn., M. Afi., and S. Vjitanen,(1999) "Stochastic global optimization, problem classes and solution techniques," *Journal of Global Optimization*. vol. 14, pp. 437–447.

[8] Johnson, Steven G., "The NLopt nonlinear-optimization package", `http://github.com/stevengj/nlopt`

[9] Neumaier, Arnold., Shcherbina, Oleg ., Huyer, Waltraud ., Vinko, Tamas., (2005). "A Comparison of Complete Global Optimization Solvers." *Mathematical Programming*.

[10] Sobol, I.M., Shukhman, B.V., (2007). "Quasi-random points keep their distance." *Mathematics and Computers in Simulation*.

[11] Ali R. Al-Roomi (2015). "Unconstrained Single-Objective Benchmark Functions Repository" [https://www.al-roomi.org/benchmarks/unconstrained]. *Dalhousie University, Electrical and Computer Engineering.*

[12] Pal, Laslo., (2013) "Benchmarking a Multi Level Single Linkage Algorithm with Improved Global and Local Phases."*Genetic and Evolutionary Computation Conference Workshop on Black-Box Optimization Benchmarking*

[13] Nelder, J. A., and Mead, R.,(1965) "A Simplex Method for Function Minimization," *Computer Journal*, Vol. 7, 308–313

[14] P. Kaelo and M. M. Ali,.(2006) "Some variants of the controlled random search algorithm for global optimization." *J. Optim. Theory Appl.* vol. 130, 253-264.

[15] D.R. Jones, C.D. Perttunen, and B. E. Stuckmann,.(1993) "Lipschitzian optimization without the lipschitz Constant." *J. Optimization Theory and Applications* vol. 79, pp. 157.

[16] Jones, D.R., Martins, J.R.R.A.,(2021) "The DIRECT algorithm: 25 years Later." *J Glob Optim* 79, 521–566.

[17] Polyak, Boris., Shcherbakov, Pavel ., (2017) "Why Does Monte Carlo Fail to Work Properly in High-Dimensional Optimization Problems?" *Journal of Optimization Theory and Applications.* vol. 173, pp. 612–627.

[18] Bertsimas, Dimitris., Tsitsiklis, John.,(1993) "Simulated Annealing." *Statistical Science* vol. 8, pp. 10-15.