Benchmarking a Multi Level Single Linkage Algorithm with Improved Global and Local Phases

László Pál

Sapientia Hungarian University of Transylvania
Miercurea-Ciuc, Romania

Genetic and Evolutionary Computation Conference
Workshop on Black-Box Optimization Benchmarking
Amsterdam, The Netherlands, July 06-10, 2013

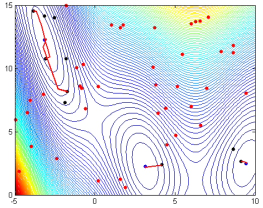
Outline

Stochastic Multistart Methods

- Global optimization techniques from the early 1980s
- Has two phases:
  - *global phase*: consists of sampling
  - *local phase*: it is based on local searches
- Multistart: inefficient in that it performs local searches starting from all sample points
- Techniques for reducing the number of local searches: *clustering* methods and *Multi Level Single Linkage (MLSL)* algorithms

Clustering Methods

- Form groups (clusters) of points around the local minimizers from a uniform sampled domain and start local searches no more than once in each of those groups
- Clusters are formed stepwise, starting from a *seed point*
- A point is added to the cluster using a *clustering rule*



- Clustering procedures are: *Density Clustering (DC)*, and *Single Linkage Clustering (SL)*

Clustering Methods – Implementations

- Boender at al. (1982, 1987)
- Csendes (1988) – GLOBAL method (Fortran and C++)[1]
- Csendes, Pál, at al. (2008) – MATLAB version of GLOBAL[2]
- Real-word applications, benchmarking:
    - Theoretical chemical problems, bioprocess analysis, climate control, integrated process design for wastewater treatment plants
    - BBOB-2009: usually performed well for moderate number of local minima and moderate size of region of attractions. It is fast in the initial phase of the optimization

[1]T. Csendes. Nonlinear parameter estimation by global optimization - efficiency and reliability, *Acta Cybernetica*, 8:361-370, 1988.

[2]T. Csendes, L. Pál, J.O.H. Sendín, J.R. Banga. The GLOBAL Optimization Method Revisited, *Optimization Letters*, 2:445–454, 2008.

Multi Level Single Linkage (MLSL) algorithm [3]

- Has been derived from clustering methods
- Local search procedure is applied to every sample point, except if there is another sample point within some *critical distance* which has a lower function value
- The algorithm has good asymptotic properties: the asymptotic probabilistic correctness and probabilistic guarantee of *finding all local minimizers*

[3] A.H.G. Rinnooy Kan, G.T. Timmer. Stochastic global optimization methods part I level methods. *Mathematical Programming*, 39:57–78, 1987.

Multi Level Single Linkage (MLSL) algorithm

Algorithm 1: The MLSL algorithm

```
1.  $X^* \leftarrow \emptyset$; $k \leftarrow 0$
2.  repeat
3.      $k \leftarrow k + 1$
4.      Generate $N$ points $x_{(k-1)N+1}, \ldots, x_{kN}$ with uniform distribution on $X$.
5.      Determine the reduced sample ($X_r$) consisting of the $\gamma kN$ best points from
        the cumulated sample $x_1, \ldots, x_{kN}$.
6.      for $i \leftarrow 1$ to $\mathrm{length}(X_r)$ do
7.          if NOT (there is such a $j$ that $f(x_j) < f(x_i)$ and $\|x_j - x_i\| < r_k$) then
8.              Start a local search method ($LS$) from $x_i$.
9.              $x^* \leftarrow LS(x_i)$
10.             $X^* \leftarrow X^* \cup \{x^*\}$
11.         end
12.     end
13. until Some global stopping rule is satisfied.
14. return The smallest local minimum value found.
```

Multi Level Single Linkage (MLSL) algorithm

- The critical distance is given by the following formula

$$r_k(x) = \frac{1}{\sqrt{\pi}} \left(\Gamma(1 + \frac{n}{2}) \cdot m(X) \cdot \frac{\zeta \ln(kN)}{kN} \right)^{1/n}.$$

- Theoretical properties:
  - probability of starting a local search decreasing to 0, if $\zeta > 2$
  - finite number of local searches with probability 1, if $\zeta > 4$, even if the algorithm is run forever
- MLSL can be ineffective when the objective function has a large number of local minimizers which are close to each other and have small basins of attraction

The aim of this study

Aims

- To improve the *global* and a *local phase*s of the MLSL method

Global phase

- Sample generated from a Sobol quasi-random sequence
- A few percent of the population is further improved by using DE (differential evolution)

Local phase

- MLSL+(fminunc, fmincon, simplex) vs TOMLAB/OQNLP
- Parameter tuning
- Local search pool

The improved MLSL method

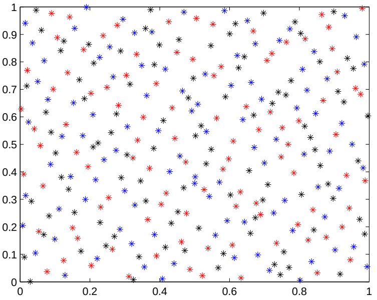
Sobol sequences

- Low-discrepancy sequences have been used instead of purely random samples
- We use sample points from Sobol quasi-random sequences which fill the space more uniformly
- Sobol low-discrepancy sequences are superior to pseudorandom sampling especially for low and moderately dimensional problems [a]

[a] S. Kucherenko, Y. Sytsko. Application of deterministic low-discrepancy sequences in global optimization, *Computational Optimization and Applications*, 30:297–318, 2005.

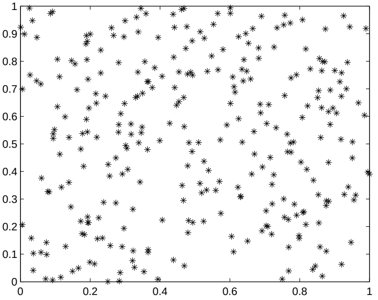
The improved MLSL method – Sobol sequences

300 points from Sobol and pseudorandom sequences



Sobol sequence

(blue=1..100, red=101..200, black=201..300)



Pseudorandom sequence

The improved MLSL method

DE iterations

- A few percent of the sample points are improved by using crossover and mutation operators similar to those used in the DE method $\rightarrow$ *Hybrid MLSL (HMLSL)*
- In other words, a few DE iterations are applied to the best points of the actual sample
- This last step is executed in each iteration before the local phase of the optimization
- The aim of these improvements are to help the MLSL method to overcome the difficulties arising in problems with a *large number of local optima* or in the cases when the *local search method cannot make further improvements*

Experimental settings

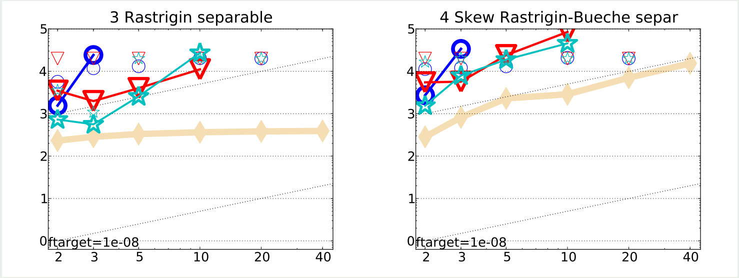
- The experiments were made using the COCO (COmparing Continuous Optimisers) platform
- Evaluations budget: $2 \cdot 10^4$
- **MLSL** parameters: sample size ($N = 50D$), reduced sample ($N_r = 5D$), local search (`fmincon` from MATLAB)
- **HMLSL** parameters: posses the same parameter settings as the MLSL algorithm. Additionally we apply $4 \cdot D$ DE iterations to the reduced sample
- **DE** parameters: population size for DE was set to $5 \cdot D$, while the mutation (F) and crossover (CR) rates to 0.5. The mutation operator uses the "DE/best/2/exp" strategy:

$$\mathbf{v} = \mathbf{x}_{best} + F \cdot (\mathbf{x}_{r1} - \mathbf{x}_{r2} + \mathbf{x}_{r3} - \mathbf{x}_{r4}).$$

ERTs in different dimensions

Rastrigin functions

Legend: ○:MLSL , ▽:DE, ⋆:HMLSL

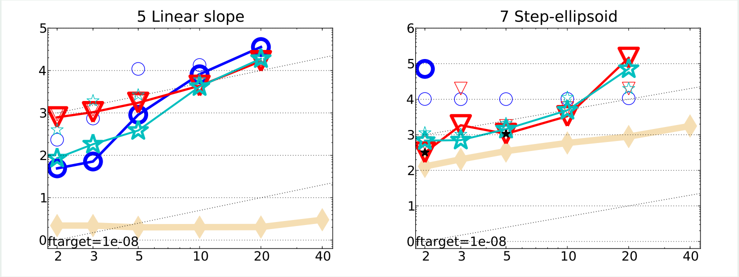


- MLSL never solves for $D \geq 3$
- HMLSL is even faster than DE

ERTs in different dimensions

Linear slope, Step ellipsoid

Legend: ∘:MLSL , ▽:DE, ⋆:HMLSL

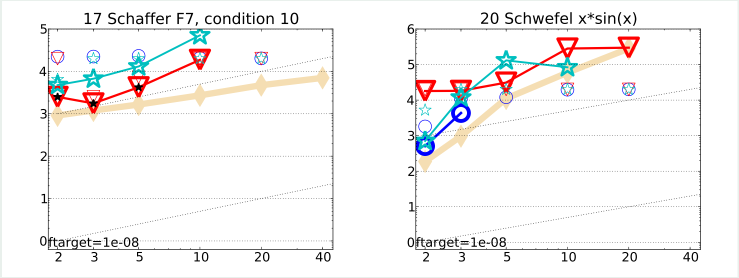


- Linear slope: HMLSL is faster than MLSL and DE for $D \geq 5$
- Step ellipsoid: the DE iterations helps HMLSL to overcome the difficulties (getting stuck on plateaus)

ERTs in different dimensions

Schaffer and Schwefel functions

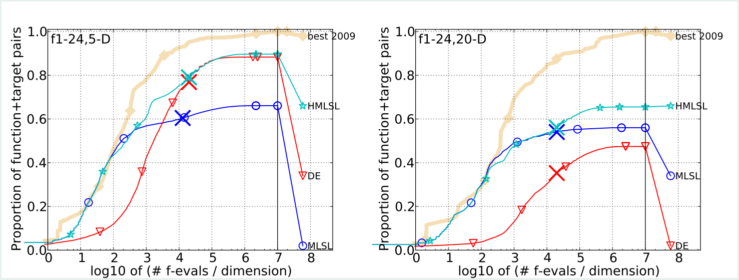
Legend: ○:MLSL , ▽:DE, ⋆:HMLSL



- Schaffer function: one of the most difficult problem for MLSL, solved now up to $D = 10$
- Schwefel function: it can be solved up to $D = 10$ by HMLSL

Proportion of solved instances

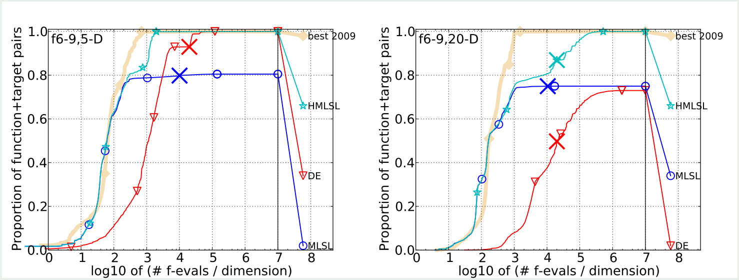
All functions



- HMLSL inherits the speed of MLSL on the initial and middle phase of the optimization
- Due to the DE iterations HMLSL is usually faster than DE and MLSL in the final phase

Proportion of solved instances

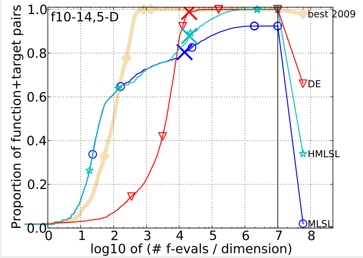
Functions with moderate conditioning

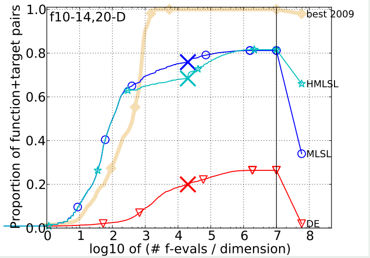


- HMLSL is significantly better than MLSL (due to solving f_7) in 5 and 20-D and also better than DE in 20-D (DE cannot solve f_6)

Proportion of solved instances

Ill-conditioned functions

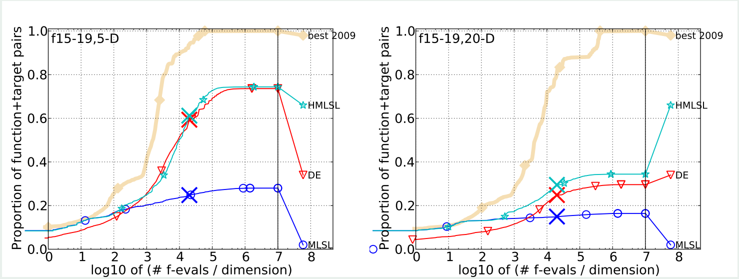




- HMLSL is significantly faster than DE up to $10^4 \cdot D$ in 5-D
- HMLSL is even faster than the best algorithm from BBOB-2009 up to $10^2 \cdot D$

Proportion of solved instances

Multi-modal functions



- HMLSL is significantly faster than MLSL by solving the f_{15}, f_{17}, and f_{18} functions in 5-D

Local search methods

- The main goal: to test MLSL with different local search methods
- Local search methods:
  - `fminunc` from MATLAB
    - Quasi-Newton method for unconstrained nonlinear optimization
    - It uses the BFGS update formula; gradient approximation with finite differences
  - `fmincon` from MATLAB
    - A solver for constrained nonlinear problems
    - It is based on trust region strategy
    - Calculates the Hessian by a quasi-Newton approximation
  - Nelder-Mead simplex
    - A direct search method

The TOMLAB/OQNLP solver

- OQNLP(OptQuest/NLP)[4]: multistart heuristic method for smooth constrained nonlinear problems
- The solver uses a scatter-search mechanism for generating start points
- The applied local search is a generalized gradient projection method
- Based on some recent comparative studies [5], OQNLP show superior performance in terms of refining a near-optimal solution

[4] Z. Ugray, L. Lasdon, J. Plummer, R. Glover, J. Kelly, and R. Marti. Scatter Search and Local NLP Solvers: A Multistar Framework for Global Optimization. *INFORMS Journal on Computing*, 19(3):328–340, 2007.

[5] L. M. Rios and N. V. Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 19(3):1-47, 2012.

Experimental settings

MLSL method

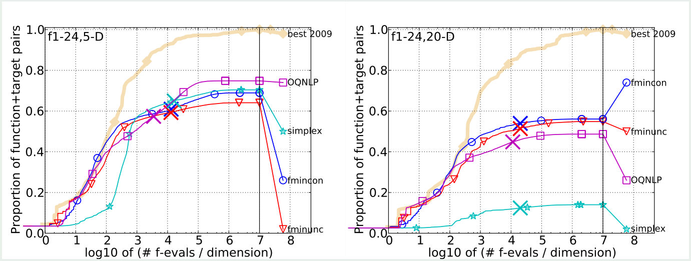
- Evaluations budget: $2 \cdot 10^4$
- Sample size ($N = 50D$), reduced sample ($N_r = 5D$)
- Local searches: `fminunc`, `fmincon`, Nelder-Mead simplex
- Max. func. evals for local searches: 10% of the total budget; $TolFun = 10^{-12}$

OQNLP solver

- Iteration limit: $300 \cdot D$
- Local search: gradient projection method

Proportion of solved instances

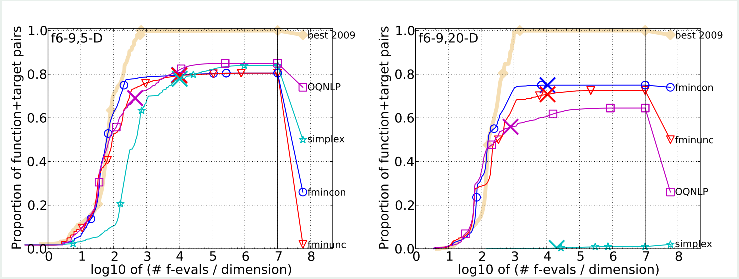
All functions



- `fminunc` is faster than `fmincon` just for small budgets (below $100D$)
- The simplex method is slow in the initial phase and is faster than the gradient type methods starting from $1000D$ (in 5-D)
- OQNLP is faster than MLSL in the final stage in 5-D

Proportion of solved instances

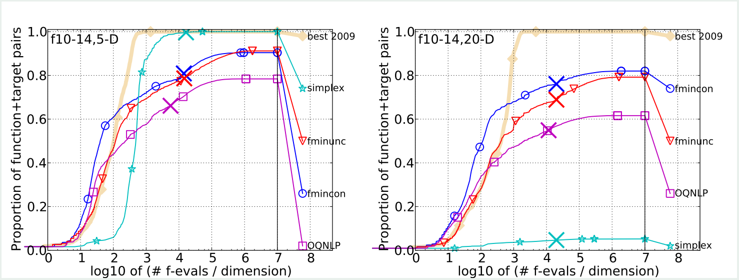
Functions with moderate conditioning



- Although `fmincon` also suffer from ill-conditioning, it is more robust than `fminunc` (especially in higher dimensions)

Proportion of solved instances

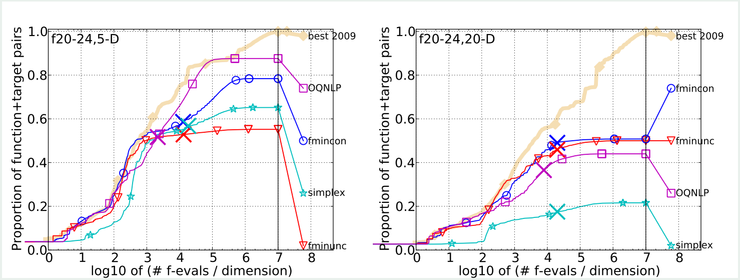
Ill-conditioned functions



- $\texttt{fmincon}$ superiority over $\texttt{fminunc}$ is more pronounced
- For $\#FEs > 700D$ simplex becomes the best competitor in 5-D

Proportion of solved instances

Multi-modal functions



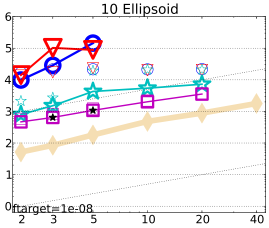
- `fmincon` usually is faster (the probability of success is also better) than `fminunc` on the f_{21}, f_{22}, and f_{23} functions

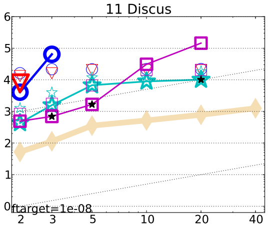
Proposals for local search improvements

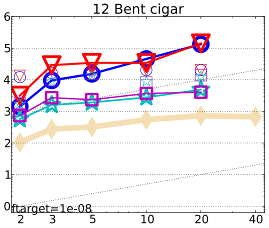
Gradient type methods (`fminunc`, `fmincon`)

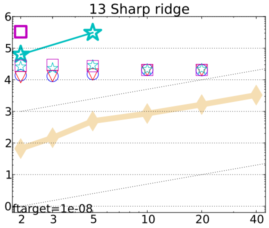
- Poor results for ill-conditioned problems, due to the rounding errors
- Starting local search with different parameter settings
- Setting up the finite differencing parameters in MATLAB:
  - *FinDiffType*: used to estimate gradients, are either 'forward' (the default), or 'central'
  - *DiffMinChange*, *DiffMaxChange*: control the perturbation levels or step size (Δx) for the current point x used in the calculation
  - *FinDiffRelStep*: control the step size in the calculation (default values: 10^{-8} in forward mode, 10^{-6} in central mode). Introduced in MATLAB R2011b.
  - It is not a trivial task to set up these parameters

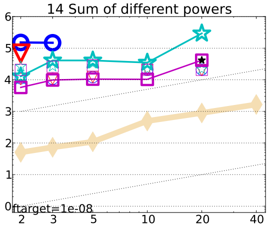
ERT results for $\mathit{FinDiffRelStep} = 10^{-10}$











○:fmincon , ⋆:cfmincon, ▽:fminunc, □:cfminunc

Proposals for local search improvements

Local search pool

- There is no local search which performs well on every problem
- Hyperheuristic: heuristic which chooses between heuristics
- The local search is chosen from a set of predefined methods based on some criterion: reduction on function value, time spent in optimization, etc.
- The local search methods competing with each other inspired from the principles of reinforcement learning
- Methods which performs poorly are penalized using tabu lists

Conclusions

- Global phase:
  - The new HMLSL method is as fast as the MLSL method in the initial and middle phase while in the final stage of the optimization it is usually faster than the MLSL and DE
- Local phase:
  - fmincon is superior to fminunc in the middle and final stage
  - Significant improvements can be achieved by re-running the local search methods with different parameters for finite differencing
  - Local search pool may also help
- Future works:
  - Sampling using scatter search
  - Using adaptive DE variants inside the MLSL method

THANK YOU FOR YOUR ATTENTION!