Competition on Real-Parameter Single Objective Expensive Optimization: A Simple Algorithm Without Approximation

Tapabrata Ray and Md Asafuddoula

***Abstract*—Real-world optimization problems often involve expensive computer simulations to evaluate candidate solutions such as via finite element analysis, computational fluid dynamics, computational electromagnetics etc. In order to contain the computational time of the optimization exercise within affordable limits, approximations/surrogates are commonly used. Numerous *Surrogate Assisted Optimization* (SAO) strategies have been proposed over the years, one claiming superiority over another evaluated using a set of test functions. In CEC-2014, a competition was organized to assess the performance of various solution strategies on a suite of 10D, 20D and 30D single objective unconstrained optimization problems with limited number of function evaluations(resembling computationally expensive optimization problems). In this report we introduce a simple algorithm to solve such problems without using any form of approximation. The study sheds light on our *obsession* with *designer approaches* and *designer benchmarks*. While comprehensive results of other optimization strategies are not available at this stage, we hope these results will offer a sanity check of our ongoing efforts.**

I. Introduction

Population based stochastic algorithms enjoyed great success in solving a wide range of practical optimization problems [1]. While population based stochastic algorithms are a preferred choice in solving such complex nonlinear problems, they require evaluation of numerous solutions prior to convergence. The use of such algorithms in their native form is rather restricted especially when candidate designs are evaluated using computationally expensive analysis such as Finite Element Methods (FEM), Computational Fluid Dynamics (CFD), and Computational Electro Magnetic (CEM) etc. Development of new strategies to deal with computationally expensive optimization problems has long been an area of active research [2]. In order to contain the computational time of the optimization exercise within affordable limits, surrogates/approximations have been regularly used in the past. Such surrogates are typically constructed using a set of solutions. Once the surrogates are trained, they can be used within any optimization algorithm in lieu of the expensive simulation [3], [4]. Various forms of surrogates have been reported in literature ranging from response surface methods (RSM), neural network based methods [3] such as multilayer perceptrons (MLP), radial basis function networks (RBFN), Support Vector Regression [5], Kriging [6] etc. Significant challenges in the context of surrogate assisted optimization include a) means to choose appropriate number and location of training samples b) type of the approximation model c) schemes of validating the approximation model d) choice of local or global approximation model e) means to retrain surrogate models and finally schemes to manage them within the framework of an optimization algorithm.

Md Asafuddoula and Tapabrata Ray are with the School of Engineering and Information Technology, University of New South Wales, Canberra, Australia (email: md.asaf@student.adfa.edu.au, t.ray@adfa.edu.au).

While there is a large volume of literature highlighting benefits of SAO, there is very limited literature on a one-to-one comparison across a range of problems. The problems introduced as a part of the CEC-2014 competition provides us an opportunity to compare the best of surrogate assisted optimization schemes with ones which does not rely on approximation. We refrain from commenting on the fact if the problems resemble real life computationally expensive optimization problems.

A simple hybrid algorithm is proposed to solve the above set of benchmarks. The details of the approach are presented in Section II, while the performance of the approach is presented in Section III. Section IV concludes the paper with final remarks.

II. Proposed Strategy

The proposed optimization strategy starts with an initial population of N individuals generated using Latin Hypercube Sampling (LHS) [7], [8]. The initial population is then evaluated and sorted based on the fitness measure. A local search *fmincon* is invoked from the best solution. If the local search stops prematurely, a *patternsearch* [9], [10] is invoked for the remaining number of function evaluations. It is clear, that the algorithm is likely to be extremely fast as compared to any SAO form. There is certainly the possibility of being stuck at the local optimum.

The pseudocode of the algorithm is presented in Algorithm 1 .

Algorithm 1 Proposed Optimization Procedure

Require: N {Population Size}
Require: FE_{max} {Maximum Number of Function Evaluations}
FE $= 0$
pop_1 = Initialize()
Evaluate_A(pop_1)
Update(FE)
x_{best}=Find_Best(pop_1)
$x_0 = x_{best}$ {Best solution identified so far}
{x_{final}, f_{final}} = Fmincon(x_0, FE_{max}-FE)
Update(FE)
$x_0 = x_{final}$ {final solution obtain from fmincon search}
{x_{final}, f_{final}} = Patternsearch(x_0,FE_{max}-FE)
Update(FE)

III. Experimental Results

The test problems are based on eight popular test functions. The test suites include unimodal / multi-modal, continuous / discrete and separable / non-separable functions

involving 10, 20 and 30 variables. The maximum number of function evaluations are limited to 500, 1000, and 1500 for 10, 20 and 30 dimensional problems. It is also important to highlight that no external parameter tuning experiments were conducted to identify the most appropriate values for the parameters listed below. Default MATLAB settings of *fmincon* and *patternsearch* was used with the prescribed number of function evaluations.

Table I provides the search settings used in this experiment for *fmincon* and *patternsearch* optimization process.

TABLE I
SEARCH SETTINGS

POP Settings	
LHS size	D
fmincon **Search Settings**	
Option	Argument
Algorithm	interior-point
Display	iter
MaxFunEvals	FE_{max} -D
TolFun	1e-8
patternsearch **Settings**	
Algorithm	interior-point
Display	iter
MaxFunEvals	FE_{max} -FE
TolFun	1e-8

*Default MATLAB settings used for other options

Table II provides the nature of the test problems, search ranges and dimension of the problems. Most functions are shifted and / or rotated. For a problem with D dimensions, the global optimum is shifted by $o_i = [o_{i1}, o_{i1}, ..., o_{iD}]$, and o_i is randomly distributed in $[-10, 10]^D$.

A. Algorithm Complexity

The algorithmic complexity is computed for the problems by calculating the mean time of completion to the completion time of the given arithmetic operations. Table III shows the complexity of the algorithm by $\hat{T}$ and $\hat{T}$/T0 where, $\hat{T}$ is the average computing time of 20 runs and T0 is the completion time of the given arithmetic operations for 1000000 times. The table also shows the best, worst, median, std completing time of each problem for 20 runs.

B. Results of the problems

The results of our proposed approach are presented in Table IV. The results are based on 20 independent runs. The approach performed well on the first nine problems, while for the remaining ones, the results are competitive(based on our experience with other strategies). The computational complexity is extremely low and possibly lower by several orders of magnitude when compared with SAO forms.

IV. CONCLUSIONS

While computationally expensive optimization problems are rarely solved without approximation, this paper tries to challenge the norm. It raises an important question "Is the field overly obsessed with the development of complex algorithms *designer algorithms* and complex test problems *designer test functions* ?"

An extremely simple algorithm is introduced in this paper which uses a Latin Hypercube Sampling and two local search strategies to look for the optimum. Standard MATLAB implementation of Latin Hypercube Sampling, fminsearch

TABLE II
PROBLEM DESCRIPTIONS

	Prob. (Dim)	Function	Search Ranges
Prob-1	01 (10) 02 (20) 03 (30)	Shifted Sphere	[-20,20]
Prob-2	04 (10) 05 (20) 06 (30)	Ellipsoid	[-20,20]
Prob-3	07 (10) 08 (20) 09 (30)	Rotated Ellipsoid	[-20,20]
Prob-4	10 (10) 11 (20) 12 (30)	Shifted Step	[-20,20]
Prob-5	13 (10) 14 (20) 15 (30)	Shifted Ackley	[-32,32]
Prob-6	16 (10) 17 (20) 18 (30)	Shifted Griewank	[-600,600]
Prob-7	19 (10) 20 (20) 21 (30)	Shifted Rotated Rosenbrock	[-20,20]
Prob-8	22 (10) 23 (20) 24 (30)	Shifted Rotated Rastrigin	[-20,20]

TABLE III
COMPUTATIONAL COMPLEXITY

Function	Computational Complexity
1	1.542
2	2.0614
3	2.8705
4	1.3557
5	2.4895
6	3.7844
7	1.4079
8	2.573
9	3.86
10	1.1896
11	2.1452
12	3.1869
13	1.5728
14	2.5905
15	3.6986
16	1.6388
17	2.6696
18	3.6372
19	1.5995
20	2.6727
21	3.6208
22	1.7015
23	2.8633
24	4.0533

TABLE IV
FUNCTION VALUES ACHIEVED WHEN FES=500, FES=1000 AND FES=1500 FOR THE PROBLEMS WITH DECISION VARIABLES 10,20 AND 30 RESPECTIVELY. FUNCTION VALUE IS TRUNCATED TO ZERO FOR THE ERROR VALUE $\leq$ 1E-8.

Function	Best	Worst	Median	Mean	Std
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	7.6095e-005	0	3.8965e-006	1.6995e-005
7	0	0	0	0	0
8	0	0	0	0	0
9	0	0.041532	0	0.0021149	0.0092785
10	0	7	2	2.45	2.2118
11	16	92	48.5	53.05	20.387
12	112	284	149	164.15	42.0554
13	6.1678	13.0471	10.8038	10.3416	2.1972
14	10.6986	14.5745	12.1641	12.4996	1.2718
15	13.7477	16.2164	14.8364	14.9734	0.74683
16	0	0.99084	0.018452	0.14055	0.30166
17	0	0.05153	0	0.004668	0.012768
18	0	0.23081	0	0.028644	0.059197
19	0.12163	6.5691	3.5187	3.5902	1.8257
20	8.9428	18.8331	14.2454	13.8768	2.1417
21	19.8303	79.1742	25.8207	29.7959	15.1251
22	32.9506	109.3217	70.5569	69.5315	24.9695
23	62.2775	221.8737	134.5738	140.0207	41.345
24	239.8141	370.1194	300.4736	306.4639	40.9999

and patternsearch have been used with its default parameter settings. The computational complexity of the proposal algorithm is extremely low. As expected, the algorithm is capable of solving unimodal problems without any difficulty. While, one might argue that it is not efficient for multimodal problems, the results do appear competitive. It is important to highlight that there are additional factors that come into play with the use of approximations. Training a model involves an inherent optimization exercise which is far from trivial. Our preliminary investigation suggests, the proposed algorithm offers decent solutions to multimodal test functions even for higher dimensions. While comprehensive results of surrogate/approximation assisted algorithms are unavailable at this stage, we hope that this study would be of significant value to the SAO community and in particular to reflect on "how good are we in terms of surrogate assisted optimization or use of approximations within an optimization algorithm ?". We would like to leave the readers with the parting thought "did we get the test functions right ?" or "did we get the SAO algorithms right ?".

REFERENCES

[1] J. Knowles, "Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 50–66, 2005.

[2] X. Lu, K. Tang, and X. Yao, "Classification-assisted differential evolution for computationally expensive problems," in *IEEE congress on Evolutionary Computation*, 2011, pp. 1986–1993.

[3] Y. Jin, M. Husken, M. Olhofer, and B. Sendhoff, "Neural networks for fitness approximation in evolutionary optimization," in *Knowledge Incorporation in Evolutionary Computation*, ser. Studies in Fuzziness and Soft Computing, Y. Jin, Ed. Springer Berlin Heidelberg, 2005, vol. 167, pp. 281–306.

[4] L. Shi and K. Rasheed, "A survey of fitness approximation methods applied in evolutionary algorithms," in *Computational Intelligence in Expensive Optimization Problems*, ser. Adaptation Learning and Optimization, Y. Tenne and C.-K. Goh, Eds. Springer Berlin Heidelberg, 2010, vol. 2, pp. 3–28.

[5] X. Llor, K. Sastry, D. E. Goldberg, A. Gupta, and L. Lakshmi, "Combating user fatigue in igas: partial ordering, support vector machines, and synthetic fitness," in *Conference on Genetic and evolutionary computation*, 2005, pp. 1363–1370.

[6] H.-S. Chung and J. Alonso, "Multi-objective optimization using approximation model based genetic algorithms," *Technical report 2004-4325*, AIAA (2004).

[7] M. D. McKay, R. J. Beckmkan, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[8] D. H. Loughlin and S. R. Ranjithan, "Chance-constrained genetic algorithms," in *Genetic Evolutionary Computation*. San Francisco, CA: Morgan Kaufmann, 1999, pp. 369–376.

[9] C. Audet and J. E. D. Jr., "Analysis of generalized pattern searches," *SIAM Journal on Optimization*, vol. 13, no. 3, p. 889903, 2003.

[10] A. R. Conn, N. I. M. Gould, and P. L. Toint, "A globally convergent augmented lagrangian barrier algorithm for optimization with general inequality constraints and simple bounds," *Mathematics of Computation*, vol. 66, no. 217, p. 261288, 1997.