DOI: 10.13009/EUCASS2017-181

DOI: ADD DOINUMBER HERE

Computationally expensive aerodynamic design optimization framework with adaptive search strategies and data fusion

LONG Bingxiang[†] *, CHEN Qin*[⋆‡] *, LIAO Daxiong*[⋆] *, WU Shenghao*[†] *and YI Xingyou*[†]
[⋆]*State Key Laboratory of Aerodyanmics, China Aerodynamic R&D Center*
Mianyang, Sichuan, China, 621000
[†]*Facility Design and Instrumentation Institute, China Aerodynamic R&D Center*
Mianyang, Sichuan, China, 621000
bingxiang2011@126.com · chenqin@cardc.cn · liaodaxiong@cardc.cn
[‡]Corresponding author

Abstract

With the rapid development of computational infrastructure and CFD techniques and tools, analysis-based aerodynamic optimization becomes more reliable and more effective. However, handy toolkit for engineers that closely integrated with the productive CFD calculating environment to make this optimal design method applicable in real engineering applications are still unavailable. In this paper, we proposed a computationally effective aerodynamic design optimization framework using Guassian process modeling, LP-τ experimental design method, and evolution algorithm on a high performance distributed cluster. The Gaussian process modeling method fuses experimental data, engineering empirical results, and CFD results to a surrogate model with more information. Three in-fill methods, confidence interval selection according to a certain threshold, probability of improvement, and expected improvement have been proposed to define optimal searching process when surrogate models are utilized. We propose an adaptive in-fill methods varies during the searching process to make the search process adaptive. The optimization search algorithm here with LP-τ experimental design methods and adaptive in-fill strategy exhibits a good balance between exploration and exploitation.

1. Introduction

When using surrogate model to find optimum of expensive function $f(x)$, where evaluations of $f(x)$ are computationally expensive, the procedure as follows proves to be effective: 1) select $X = \{x_i, i = 1 \dots N\}$ as initial trial to be evaluated by design of experiments, $Y = \{y_i = f(x_i), i = 1 \dots N\}$; 2) building surrogate model $\bar{f}$; 3) exploring design space and improving $\bar{f}$. Different iteration of the second and third steps above, i.e., building models, select candidates, evaluate candidates, and improve model, have been proposed to define different framework. For real life application implement by domain engineers, the iteration usually is carried only once without improving the model. While researchers in optimization algorithm define more delicate process to gain better performance, which are usually called efficient global optimization (EGO).[3,6,13]

To implement EGO frameworks for real-life application, the choice of surrogate model and selection of candidates are crucial.[12] The choice of surrogate models define the basic characteristics of optimization framework. In engineering, response surface method is commonly used for local search. When the research achievements in machine learning (ML) being imported to optimization domain, Kernel-based methods like Gaussian process model prove to be effective and show good potential.[7] And artificial neural network (ANN)[1] methods also exhibit excellent performance and highly flexible characteristics. The algorithm to select candidates by surrogate model depends on the characteristics of surrogate.

For Gaussian process model, predictions of trail design points are Gaussian distribution. The deviation information can be a good way to fuse uncertainty information presented in training data set. The following in-fill rules can b be defined to allocate candidates: confidential intervals rule - min $\bar{f}(x) + \epsilon\sigma_{\bar{f}(x)}(\epsilon \in [0, 3])$, probability of improvement rule - max $P\left(f(x^*) \geq f(x)\right)$, and expected improvement rule - max $E\left[f(x^*) - f(x)\right]$. When $\epsilon = 0$, confidential interval in-fill rule is the commonly used method by engineers. And expected improvement method is considered more effective in balancing exploration and exploitation automatically. While in real life application, the performance of different in-fill

DOI: 10.13009/EUCASS2017-181

rules varies with different characteristics of problems.[5] Adaptive in-fill rules are proposed[2,9] to automatically change the search strategies in different stage of optimization and for different problems.
This paper proposed another framework based on LP-τ experiment design method and Gaussian process model. All candidates allocated by continuously sample a Sobol sequence, filtered with adaptive in-fill rules. The framework is tested against 150 test functions, then applied on a real-life engineering problem. The following contents are arranged as a procedure description, benchmark function test results, real-life application, and conclusion.

2. Procedure and algorithm

2.1 Framework description

The problem solved by the framework are unconstrained minimization problem as eq. (1), where single objective $f : D \mapsto R$ is computationally expensive to evaluate. Due to the expensive evaluation of f, the main cost of the optimization can be described by N_E, $D \subseteq R^n$ is the design space to be explored, and n is the dimension of the design space. For real-life problem with constrains, it's easy to transform the constrained problem to unconstrained version via method of Lagrange multipliers.

$$\min_{x \in D} f(x) \tag{1}$$

---

1. Initialization: Sobol sequence $x_1, x_2, \ldots \in D$
2. LP-τ Design of Experiments from Sobol sequence $(x_1, x_2, \ldots, x_{N0})$
3. Evaluation $\{f(x), \Delta_{f(x)} | x \in \{x_1, x_2, \ldots, x_{N0}\}\}$
4. Iterate through the following until Stop Criteria ($N \leq N_E$) satisfied:
   (a) Building Gaussian Process Model
   (b) Find attraction centers according to in-fill rules
   (c) Allocate candidates by applying filters of attraction centers on Sobol sequence (N_S)
   (d) Evaluation candidates

Figure 1: Optimization framework based on LP-τ DOE and Gaussian process model

The key parts of the algorithm framework consist of three components, LP-*tau* design of experiments, i.e., the Sobol sampling sequence $x_1, x_2, \ldots \in D$; surrogate model $\bar{f} : D \mapsto C(\bar{f})$, where $C(\cdot)$ is the domain of function $f(x)$; in-fill rules $g : C(\bar{f}) \mapsto R$. In this framework, all candidates to be evaluated are selected from the same DOE sequence, which we select Sobol sequence based on its characteristics. Surrogate model are trained with evaluated design points, $X_e = \{x_1, x_2, \ldots, x_{n_e}\}$, which act as approximation function of f, denoted as $\hat{f}$. For Gaussian process model used here, the surrogate is a function that $D \mapsto N(\cdot, \cdot)$, where $N(\cdot, \cdot)$ means a Gaussian distribution:

$$\hat{f}(x) \sim N(\bar{f}(x), \sigma_f^2(x)) \tag{2}$$

By using in-fill rules defined in the framework, the original optimization problem will be transform into a new problem as Eq. (3). The forms of function $g(\cdot)$ will be given in the follow section. The in-fill rule and its related function $g(\cdot)$ is regarded equivalent in the follow description.

$$\min_{x \in D} g(\hat{f}(x)) \tag{3}$$

By solve the surrogate optimization problem, we can get x_t as attraction center. If there are several in-fill rules, $g_1, g_2, \ldots$, a attraction set will be generated, $X_c = \{x_{t1}, x_{t2}, \ldots\}$. The filter functions can be defined on x_{ti} as follows:

$$F_{x_{ti}} : D \mapsto \{\text{True}/1, \text{False}/0\} \tag{4}$$

N_S candidates $X_c = \{x_{c1}, x_{c2}, \ldots x_{cN_S}\}$ can be selected from the Sobol sequence $x_1, x_2, \ldots \in D$. Candidates in X_c will be evaluated with expensive function $f(x)$, where the uncertainty of $f(x)$, i.e., $\Delta_{f(x)}$ also presented. X_c and $\{f(x), \Delta_{f(x)} | x \in X_c\}$ will be used to update surrogate model $\hat{f}$, which finish a complete iteration.

DOI: 10.13009/EUCASS2017-181

2.2 Quasi-random sequences

LP-τ DOE utilize quasi-random sequence called Sobol sequence, which was first defined by Russian mathematician IIya M. Sobol in 1967. The aim was to achieve best convergence performance to calculate integration $\int f(x)$ by Monte Carlo methods. In Eq. (5), sequence x_i proposed by Sobol gives best convergence against n.

$$\int f(x) = \lim_{n\to\infty} \frac{1}{n} \sum_{i} f(x_i) \tag{5}$$

The logic to use Sobol sequence in the framework lies on consideration as follows. Based on former experience in EGO practice, candidates directly allocated by surrogate model by in-fill rules tends to be converged to premature location in the design space, which makes singularity occurs in surrogate model training and which also stall the search process. While the low discrepancy of Sobol sequence can improve this behavior, which guarantees good exploration in the design space. As a quasi-random sequence, Sobol sequence can be generated more easily than sequences defined other DOE methods.

2.3 Gaussian model for data fusion

The efficient global optimization (EGO) based on Gaussian process model are well defined, where the surrogate model can handle noisy input in evaluation. The methods and related formulation are well defined and documented.[8,10,11] The Gaussian process model proves to be efficient to fuse data in different sources in our framework. The only assumption made in the framework is that evaluations are independent, thus the correlation of uncertainty of different evaluations are zero, i.e., $\mathrm{Cor}(\Delta_{f(x_1)}, \Delta_{f(x_2)}) = 0$.

2.4 Filter defined by In-fill rules

For simpler surrogate models, use prediction $\hat{f}(X)$ as real performance $f(x)$ is straight forward for engineers. But for Gaussian process model, more information is presented when predicting. By using $N(\bar{f}, \sigma^2\bar{f})$, meaningful in-fill rules have been defined as confidence interval, probability of improvement, expected improvement, and quantile-based estimation. For different problems, and in different stage of searching process, the in-fill rules have different performance. Chen et al[2] proposed an adaptive procedure based on probability management of several in-fill rules. In the framework defined here, Sobol sequence is used with filters defined by in-fill rules. Here we use 9 in-fill rules showed in Table 1

Table 1: In-fill rules used in the framework

In-fill rules	**Expression**
Confidence interval (CI_ϵ)	$\min \bar{f}(x) + \epsilon\sigma_{\bar{f}(x)}$
Probability of improvement (PI)	$\max P\left(f(x^*) \geq f(x)\right)$
Expected improvement (EI)	$\max E\left[f(x^*) - f(x)\right]$
For CI_ϵ in-fill rules, $\epsilon = -3, -2, -1, 0, 1, 2, 3$	

Nine attraction center can be obtained by solving Eq. (3). Filter functions are defined by the following passing threshold.

$$P(d) = e^{-\ln 2 \frac{d}{d_0}} \tag{6}$$

$d = |x_c - x_a|$ defines distance between a sample in Sobol sequence (x_c) and attraction center (x_a). For each x_c, a random number with uniform distribution will be generated as r. Only if $r < P(d)$, x_c will be accepted. For a given x_c, accepted by any filter function will be sufficient. d_0 denotes the attraction radius, where the accept ratio be 50%.

3. Results of test functions

Andrea Gavana[4] lists more than 180 test functions, we choose 150 of them that can be defined in R^2 to benchmark our framework.

N_0 is set to 21 for all instances, N_E is set to 40. Attraction radius is defined as a decrease sequence related to N linearly, which start from one tenth of the design space range for each dimension, and which end to one ten-thousandth. For

DOI: 10.13009/EUCASS2017-181

each function, the algorithm is carried out 10 times, then mean, std, worse, and best of the f values achieved with 40 evaluation are reported.
For 100 test functions in Table 2 and Table 3, 34 functions are solved for all 10 runs in precision that can be accepted in practice, which are marked bold in the column Mean. Further, 26 functions are solved in some of the 10 runs, which are bold in the column Best. For 50 test functions in Table 4, global optimum located in the center of the design space, which are the first sample of Sobol sequence $(0.5, 0.5)$. Shift should be applied to test the framework on these functions.

4. Real-life application

4.1 Problem description

For subsonic diffuser aerodynamic design in wind tunnel, we can simplify the real problems as skinned pipe line showed in Fig. 2, which geometry can be defined by semi-height at two position in the diffusing section. The aerodynamic performance can be defined as the total pressure loss coefficient (k) of the diffusing section, see Eq. (7). Thus the design problem is defined by a optimization problem $R^2 \mapsto R$. Though this problem can be defined as more complicated profile in the diffusing section, this version of simplified pipe gives acceptable performance in our application.

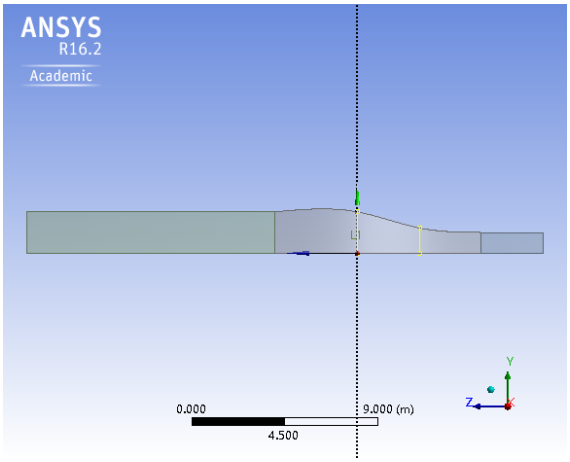

Figure 2: Diffuser geometry and design parameters

$$k = \frac{\bar{P}^0_{inlet} - \bar{P}^0_{outlet}}{0.5\bar{\rho}\bar{V}^2} \tag{7}$$

The performance evaluation is carried out with Ansys® Workbench, where the project is showed as Fig. 3. Through Ansys® IronPython script, new design point with two input parameters is created, then the geometry is automatically modeled, then meshing, then CFX-pre, CFX flow simulation, and then CFX-post to get k.
The multi-block meshing with wall inflation in Ansys® proves to be reliable and convenient. A example of mesh is showed in Fig. 4
CFX result example is show in Fig. 5, CFX-post CCL is used to calculate Eq. eq:k, in which flow parameters are area-averaged in inlet and outlet of the diffusing section.

4.2 Optimization results

The design space are $[1, 2] \times [1, 2]$. Evaluation takes 30 minutes to converge. The parameter of the optimization framework are as follows: $N_0 = 21$, $N_E = 43$, and $N_S = 1$. There are 5 design points which the performances are estimated via engineering calculation. The convergence of the optimization are shown in Fig. 6, where we have to admit the best solution is allocated in the pure LP-τ design phrase.
Although the searching iterations don't allocate better solution, Gaussian process models are improved through iteration, which can be shown in Fig. 7 and Fig. 8

DOI: 10.13009/EUCASS2017-181

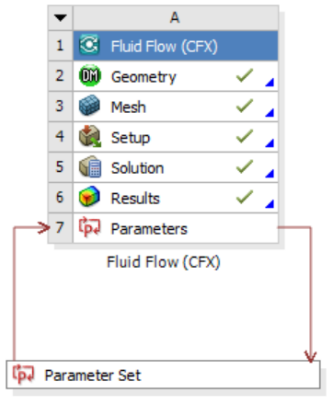

Figure 3: ANSYS® Workbench project

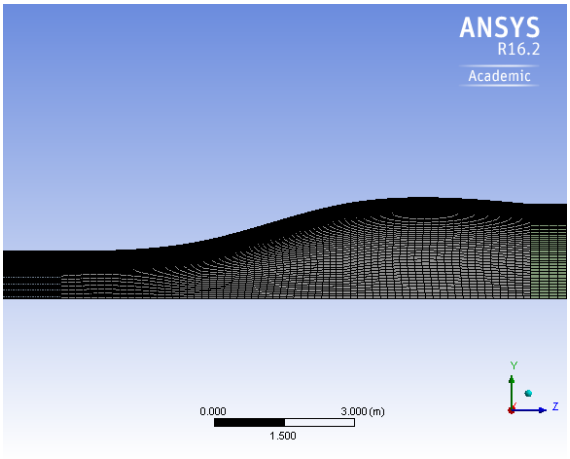

Figure 4: Mesh by ANSYS® Meshing™

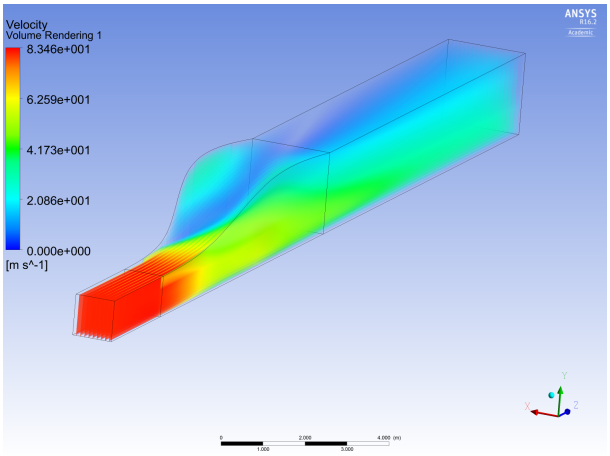

Figure 5: Example result by ANSYS® CFX

5. Discussion

5.1 Filter function definition

The distance $d = |x_c - x_a|$ in definition of filter functions can take different forms since there are different definition of distance in math point of view. Distance definition can affect the behavior of filter functions. In the 2D problems here,

DOI: 10.13009/EUCASS2017-181

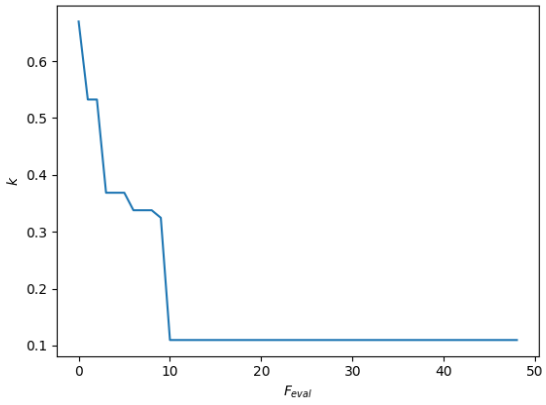

Figure 6: Convergence result of diffuser optimization

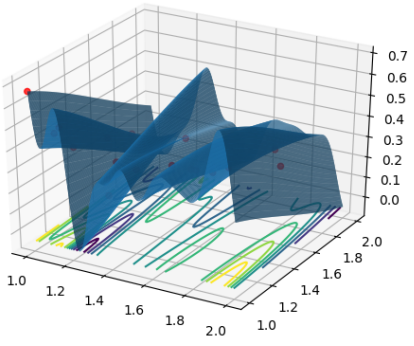

Figure 7: Gaussian process model with 21 explored locations

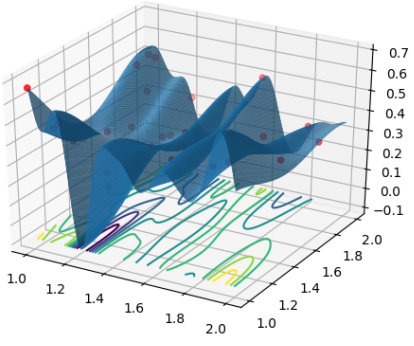

Figure 8: Gaussian process model with 34 explored locations

Euclid distance is sufficient. While for problems with higher dimensions, Euclid distance will make the filter function too difficult to accept a candidate. In higher dimension cases, $d = \|x_c - x_a\|_\infty$ will be more appropriate. Further investigation about the behavior of filter functions and dimension effect may reveal more details.

DOI: 10.13009/EUCASS2017-181

5.2 Gaussian process model training

In the benchmark function optimization and real-life application, there are over-fit when training the Gaussian process model. This over-fit will produce unrealistic landscape which make the searching process meaningless. In further research, two paths should be follow to overcome this. Firstly, the structure of Gaussian process model should be systematically investigated. Secondly, we should try to improve training procedure. In this paper, differential evolution algorithm is used to training Gaussian process model with all data available. Training procedure that is more reliable and robust must be found to improve the effectiveness of the searching.

6. Conclusion

In this paper, we proposed a computationally effective aerodynamic design optimization framework using Guassian process modeling, LP-τ experimental design method, and evolution algorithm on a high performance distributed cluster. The Gaussian process modeling method fuses experimental data, engineering empirical results, and CFD results to a surrogate model with more information. Three in-fill methods, confidence interval selection according to a certain threshold, probability of improvement, and expected improvement have been proposed to define optimal searching process when surrogate models are utilized. We propose an adaptive in-fill methods varies during the searching process to make the search process adaptive. The optimization search algorithm here with LP-τ experimental design methods and adaptive in-fill strategy exhibits a good balance between exploration and exploitation.
The research work in this paper is still in its preliminary stage, further investigation should be carried out in the following three topics: 1) why the framework fail in 40 percent functions? What's the properties in the test functions makes EGO difficult to solve? 2) structure and training of Gaussian process model; 3) apply the framework to problems with higher dimensions.

7. Acknowledgments

The work was supported by State Key Laboratory of Aerodyanmics, China Aerodynamic R&D Center. And Ansys ® Workbench is provided freely under a education license.

References

[1] Nessrine Azzouz, Slim Bechikh, and Lamjed Ben Said. Steady state ibea assisted by mlp neural networks for expensive multi-objective optimization problems. *Genetic and evolutionary computation conference*, pages 581–588, 2014.

[2] Qin Chen, Bingxiang Long, and Qingfu Zhang. Black-box expensive multiobjective optimization with adaptive in-fill rules. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016.

[3] Marcus Frean and Phillip Boyle. Using gaussian processes to optimize expensive functions. *Australian joint conference on artificial intelligence*, pages 258–267, 2008.

[4] Andrea Gavana. Test functions index, http://infinity77.net/global_optimization/test_functions.html, Retrived in 2017/06/02.

[5] Likeng Huang, Zhenghong Gao, and Dehu Zhang. Research on multi-fidelity aerodynamic optimization methods. *Chinese Journal of Aeronautics*, 26(2):279–286, 2013.

[6] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

[7] Jack P C Kleijnen, Wim C M Van Beers, and Inneke Van Nieuwenhuyse. Constrained optimization in expensive simulation: Novel approach. *European Journal of Operational Research*, 202(1):164–174, 2010.

[8] Patrick Koch, Tobias Wagner, Michael Emmerich, Thomas Back, and Wolfgang Konen. Efficient multi-criteria optimization on noisy machine learning problems. *Applied Soft Computing*, 29:357–370, 2015.

[9] Guiyuan Lei. Adaptive random search in quasi-monte carlo methods for global optimization. *Computers & Mathematics With Applications*, 43:747–754, 2002.

DOI: 10.13009/EUCASS2017-181

[10] Victor Picheny and David Ginsbourger. Noisy kriging-based optimization methods: A unified implementation within the diceoptim package. *Computational Statistics & Data Analysis*, 71:1035–1053, 2014.

[11] Victor Picheny, Tobias Wagner, and David Ginsbourger. A benchmark of kriging-based infill criteria for noisy optimization. *Structural and Multidisciplinary Optimization*, 48(3):607–626, 2013.

[12] Hongtao Wang, Xiaocheng Zhu, and Zhaohui Du. Aerodynamic optimization for low pressure turbine exhaust hood using kriging surrogate model. *International Communications in Heat and Mass Transfer*, 37(8):998–1003, 2010.

[13] Qingfu Zhang, Wudong Liu, Edward Tsang, and Botond Virginas. Expensive multiobjective optimization by moea/d with gaussian process model. *IEEE Transactions on Evolutionary Computation*, 14(3):456–474, 2010.

DOI: 10.13009/EUCASS2017-181

Table 2: Results of test functions (2 dimensions, 40 function evaluations, 10 runs)

Function Name	f^*	**Mean**	**Std**	**Worse**	**Best**
AMGM	0.0000	**0.0000**	0.0000	0.0000	0.0000
Adjiman	-2.0218	-1.9978	0.0208	-1.9678	**-2.0197**
Alpine02	-6.1295	**-6.1271**	0.0020	-6.1223	-6.1291
Beale	0.0000	1.6274	0.7867	2.6622	**0.1829**
Bird	-106.7645	-104.6640	5.2090	-89.0845	**-106.7500**
Branin01	0.3979	0.4006	0.0022	0.4045	**0.3983**
Branin02	5.5590	7.8230	0.4719	8.2762	6.5164
Brent	0.0000	**0.0000**	0.0000	0.0000	0.0000
Brown	0.0000	0.2316	0.1091	0.3157	**0.0059**
Bukin06	0.0000	7.3366	3.3382	13.7429	2.2434
CarromTable	-24.1568	**-24.1430**	0.0071	-24.1366	-24.1517
Chichinadze	-42.9444	-39.1370	3.5182	-31.6133	**-42.4286**
CrossInTray	-2.0626	**-2.0613**	0.0006	-2.0604	-2.0624
Cube	0.0000	0.9708	0.0813	1.0000	0.7275
Damavandi	0.0000	2.0000	0.0000	2.0000	2.0000
Deb01	-1.0000	**-1.0000**	0.0000	-1.0000	-1.0000
Deb02	-1.0000	**-0.9745**	0.0242	-0.9283	-0.9988
Decanomial	0.0000	1409.0824	711.1093	1864.0293	10.5047
Deceptive	-1.0000	-0.7586	0.1018	-0.6307	**-0.9326**
DeckkersAarts	-24776.0000	-11211.7383	10099.1964	0.0000	**-24409.6889**
DixonPrice	0.0000	0.1404	0.1584	0.5680	**0.0037**
EggHolder	-959.6407	-573.1228	61.5681	-481.6588	-692.5314
ElAttarVidyasagarDutta	1.7128	145.1829	9.6358	150.0000	125.5339
Exp2	0.0000	2.7021	0.4114	2.9793	1.9381
FreudensteinRoth	0.0000	7.0433	8.4623	28.0158	**0.0002**
Giunta	0.0645	**0.0645**	0.0001	0.0647	0.0645
GoldsteinPrice	3.0000	29.1838	7.7640	32.6875	7.0394
Hansen	-176.5400	**-79.7909**	41.4710	-17.5894	-135.8661
HolderTable	-19.2085	-17.8052	1.5232	-15.1402	**-19.2019**
Hosaki	-2.3458	**-2.3325**	0.0211	-2.2726	-2.3454
JennrichSampson	124.3622	**376.9721**	148.1732	694.4142	221.0884
Judge	16.0817	17.0777	0.7701	18.5338	**16.0998**
Katsuura	1.0000	**1.0000**	0.0000	1.0000	1.0000
Langermann	-5.1621	-3.2536	0.1106	-2.9316	-3.3163
Leon	0.0000	**0.0238**	0.0307	0.1070	0.0012
Levy03	0.0000	0.1239	0.0892	0.2267	**0.0017**
Levy05	-176.1375	-25.1090	31.6967	0.4635	-111.8958
Levy13	0.0000	0.3467	0.2439	1.0738	**0.1937**
McCormick	-1.9132	**-1.9131**	0.0000	-1.9131	-1.9132
Michalewicz	-1.8013	**-1.7956**	0.0084	-1.7768	-1.8011
Mishra01	2.0000	**2.0016**	0.0009	2.0028	2.0003
Mishra02	2.0000	**2.0054**	0.0060	2.0188	2.0001
Mishra03	-0.1847	-0.1295	0.0000	-0.1295	-0.1295
Mishra04	-0.1994	0.0000	0.0000	0.0000	0.0000
Mishra05	-0.1198	-0.0984	0.0102	-0.0875	**-0.1174**
Mishra06	-2.2839	-1.9268	0.1971	-1.6622	**-2.2622**
Mishra07	0.0000	**0.0203**	0.0588	0.1968	0.0000
Mishra08	0.0000	1571.4000	431.4713	1864.0293	584.0476
Mishra10	0.0000	**0.0000**	0.0000	0.0000	0.0000
NewFunction01	-0.1789	0.1007	0.0896	0.2330	**-0.0744**

DOI: 10.13009/EUCASS2017-181

Table 3: Results of test functions (2 dimensions, 40 function evaluations, 10 runs)-Continued

Function Name	f^*	Mean	Std	Worse	Best
NewFunction02	-0.1972	-0.0056	0.0168	0.0000	-0.0561
NewFunction03	-1.0198	**-0.9369**	0.0771	-0.8187	-1.0057
OddSquare	-1.0084	-0.4873	0.3902	-0.0420	**-1.0010**
Parsopoulos	0.0000	**0.0147**	0.0135	0.0424	0.0000
PenHolder	-0.9635	**-0.9635**	0.0000	-0.9635	-0.9635
Penalty01	0.0000	0.4562	0.1847	0.5694	**0.0313**
Penalty02	0.0000	0.2000	0.0000	0.2000	0.2000
PermFunction01	0.0000	**0.0127**	0.0097	0.0337	0.0014
PermFunction02	0.0000	0.3496	0.2293	0.7902	0.0414
Price01	0.0000	38.0327	15.3823	50.0000	15.8203
Price03	0.0000	1.0000	0.0000	1.0000	1.0000
Price04	0.0000	**0.0000**	0.0000	0.0000	0.0000
Qing	0.0000	5.0000	0.0000	5.0000	5.0000
Quadratic	-3873.7242	-3865.8392	10.7754	-3840.3172	**-3873.7091**
Quintic	0.0000	7.1802	1.3374	8.0000	4.5036
Rana	-928.5478	**-928.5479**	0.0000	-928.5479	-928.5479
Ripple01	-2.2000	-1.7317	0.1253	-1.5556	-1.9564
Ripple25	-2.0000	-1.7142	0.0964	-1.6503	**-1.9983**
Rosenbrock	0.0000	2.5766	1.7469	5.6338	0.7461
RosenbrockModified	34.3700	74.2711	0.2124	74.7434	74.0185
Schaffer03	0.0016	0.4947	0.0074	0.4999	0.4734
Schaffer04	0.2926	0.4948	0.0072	0.5000	0.4808
Schwefel04	0.0000	**0.0958**	0.1680	0.5618	0.0014
Schwefel06	0.0000	1.5720	1.5390	5.8057	0.2632
Schwefel26	0.0000	177.1326	197.4706	468.3292	**0.0119**
Schwefel36	-3456.0000	-459.6590	935.4378	-0.0000	-2684.9361
Shubert01	-186.7309	-37.3417	7.2438	-28.3636	-51.9848
Shubert03	-24.0625	-21.3099	1.5290	-19.2920	**-24.0194**
Shubert04	-29.0160	-20.8855	4.0198	-16.7376	-28.9561
SixHumpCamel	-1.0316	-0.4799	0.3223	-0.0808	-0.9200
Step	0.5000	**0.5000**	0.0000	0.5000	0.5000
Stochastic	0.0000	0.2328	0.1559	0.5956	0.0402
StretchedV	0.0000	**0.0000**	0.0000	0.0000	0.0000
StyblinskiTang	-78.3323	**-78.0313**	0.1782	-77.8280	-78.3062
TestTubeHolder	-10.8723	-10.1062	0.3708	-9.5155	**-10.7861**
Treccani	0.0000	**0.0000**	0.0000	0.0000	0.0000
Trefethen	-3.3069	-0.8112	0.8650	0.5160	-2.5963
Trigonometric01	0.0000	**0.0000**	0.0000	0.0000	0.0000
Trigonometric02	1.0000	18.5506	0.0000	18.5506	18.5506
Tripod	0.0000	1.0000	0.0000	1.0000	1.0000
Ursem01	-4.8168	**-4.8147**	0.0061	-4.7965	-4.8168
UrsemWaves	-8.5536	-7.1195	0.5801	-6.4067	-7.6418
Vincent	-2.0000	**-1.9944**	0.0077	-1.9736	-2.0000
WayburnSeader01	0.0000	46.5408	46.3653	136.4109	4.1740
WayburnSeader02	0.0000	88.2379	0.0000	88.2379	88.2379
Whitley	0.0000	1.8398	0.0000	1.8398	1.8398
Zacharov	0.0000	0.1977	0.2005	0.7272	**0.0231**
Zettl	-0.0038	0.1104	0.0536	0.1899	**0.0424**
Zimmerman	0.0000	1118.5439	243.1506	1300.0000	473.1644
Zirilli	-0.3523	**-0.3461**	0.0085	-0.3224	-0.3520

DOI: 10.13009/EUCASS2017-181

Table 4: Test functions with global minimum centered in search space

Function Name	x^*	$[(x[0]_{lb}, x[0]_{ub}), (x[1]_{lb}, x[1]_{ub})]$
Ackley	[0, 0]	[(-30.0, 30.0), (-30.0, 30.0)]
Alpine01	[0, 0]	[(-10.0, 10.0), (-10.0, 10.0)]
BartelsConn	[0, 0]	[(-5.0, 5.0), (-5.0, 5.0)]
Bohachevsky	[0, 0]	[(-15.0, 15.0), (-15.0, 15.0)]
Bukin02	[-10.0, 0.0]	[(-15.0, -5.0), (-3.0, 3.0)]
Bukin04	[-10.0, 0.0]	[(-15.0, -5.0), (-3.0, 3.0)]
Cigar	[0, 0]	[(-100.0, 100.0), (-100.0, 100.0)]
CosineMixture	[0, 0]	[(-1.0, 1.0), (-1.0, 1.0)]
CrownedCross	[0, 0]	[(-10.0, 10.0), (-10.0, 10.0)]
DeflectedCorrugatedSpring	[5.0, 5.0]	[(0, 10.0), (0, 10.0)]
DropWave	[0, 0]	[(-5.12, 5.12), (-5.12, 5.12)]
Easom	[0, 0]	[(-100.0, 100.0), (-100.0, 100.0)]
EggCrate	[0.0, 0.0]	[(-5.0, 5.0), (-5.0, 5.0)]
Exponential	[0.0, 0.0]	[(-1.0, 1.0), (-1.0, 1.0)]
Griewank	[0, 0]	[(-600.0, 600.0), (-600.0, 600.0)]
HimmelBlau	[0.0, 0.0]	[(-6, 6), (-6, 6)]
Matyas	[0, 0]	[(-10.0, 10.0), (-10.0, 10.0)]
Mishra11	[0.0, 0.0]	[(-10.0, 10.0), (-10.0, 10.0)]
MultiModal	[0.0, 0.0]	[(-10.0, 10.0), (-10.0, 10.0)]
Pathological	[0, 0]	[(-100.0, 100.0), (-100.0, 100.0)]
Pinter	[0.0, 0.0]	[(-10.0, 10.0), (-10.0, 10.0)]
Plateau	[0.0, 0.0]	[(-5.12, 5.12), (-5.12, 5.12)]
Price02	[0.0, 0.0]	[(-10.0, 10.0), (-10.0, 10.0)]
Rastrigin	[0, 0]	[(-5.12, 5.12), (-5.12, 5.12)]
RotatedEllipse01	[0.0, 0.0]	[(-500.0, 500.0), (-500.0, 500.0)]
RotatedEllipse02	[0.0, 0.0]	[(-500.0, 500.0), (-500.0, 500.0)]
Salomon	[0.0, 0.0]	[(-100.0, 100.0), (-100.0, 100.0)]
Sargan	[0.0, 0.0]	[(-100.0, 100.0), (-100.0, 100.0)]
Schaffer01	[0.0, 0.0]	[(-100.0, 100.0), (-100.0, 100.0)]
Schaffer02	[0.0, 0.0]	[(-100.0, 100.0), (-100.0, 100.0)]
Schwefel01	[0.0, 0.0]	[(-100.0, 100.0), (-100.0, 100.0)]
Schwefel02	[0.0, 0.0]	[(-100.0, 100.0), (-100.0, 100.0)]
Schwefel20	[0.0, 0.0]	[(-100.0, 100.0), (-100.0, 100.0)]
Schwefel21	[0.0, 0.0]	[(-100.0, 100.0), (-100.0, 100.0)]
Schwefel22	[0.0, 0.0]	[(-100.0, 100.0), (-100.0, 100.0)]
SineEnvelope	[0, 0]	[(-100.0, 100.0), (-100.0, 100.0)]
Sodp	[0, 0]	[(-1.0, 1.0), (-1.0, 1.0)]
Sphere	[0, 0]	[(-5.12, 5.12), (-5.12, 5.12)]
ThreeHumpCamel	[0.0, 0.0]	[(-5.0, 5.0), (-5.0, 5.0)]
Ursem03	[0.0, 0.0]	[(-2, 2), (-1.5, 1.5)]
Ursem04	[0.0, 0.0]	[(-2.0, 2.0), (-2.0, 2.0)]
VenterSobiezcczanskiSobieski	[0.0, 0.0]	[(-50.0, 50.0), (-50.0, 50.0)]
Wavy	[0.0, 0.0]	$[(-\pi, \pi), (-\pi, \pi)]$
Weierstrass	[0.0, 0.0]	[(-0.5, 0.5), (-0.5, 0.5)]
XinSheYang01	[0, 0]	[(-5.0, 5.0), (-5.0, 5.0)]
XinSheYang02	[0, 0]	$[(-2\pi, 2\pi), (-2\pi, 2\pi)]$
XinSheYang03	[0, 0]	[(-20.0, 20.0), (-20.0, 20.0)]
XinSheYang04	[0, 0]	[(-10.0, 10.0), (-10.0, 10.0)]
YaoLiu04	[0, 0]	[(-10.0, 10.0), (-10.0, 10.0)]
YaoLiu09	[0, 0]	[(-5.12, 5.12), (-5.12, 5.12)]