Enhancing Simulation-Driven Optimization by Machine-Learning

Yoel Tenne

***Abstract*—Computer simulations are being extensively used in engineering design optimization to evaluate candidate designs instead of real-world experiments. Often for some of the candidate designs the simulation will fail for an unknown reason, which leads to wasted computer resources and poor final results. To handle this challenge both effectively and efficiently this study presents an implementation in which classifiers, borrowed from the domain of machine learning, are integrated into the optimization process to predict if a candidate design is valid or not prior to evaluating it with the simulation. To further enhance the optimization effectiveness two different search methods are used. Numerical experiments show the merit of the proposed implementation.**

***Index Terms*—Expensive black-box functions, machine learning, classifiers, metamodels.**

I. Introduction

The use of computer simulations in engineering design continues to grow since they allow for a more efficient design process. The use of such simulations transforms the design process into an optimization problem with the following unique features [1]:

The simulation acts as the objective function since it assigns an output value to a candidate design, but its analytic expression is unknown, namely, it is as a black-box function. This is since it is often a legacy code or a commercial software whose inner mechanics are inaccessible.

Each simulation run is often computationally expensive which in turn severely limits the possible number of evaluations.

Both the real-world physics being simulated and the numerical approximations often result in an objective with a complicated landscape and this further exacerbates the optimization difficulty.

Such problems are appropriately termed as expensive black-box optimization problems and due to their prevalence they have received significant attention [1]. An additional challenge which is often encountered is that for some candidate designs the simulation will fail without providing an output value. In this study such designs are termed as simulator-infeasible (SI), while valid designs are termed simulator-feasible (SF). The presence of SI designs in an optimization process implies that: i) the objective function is discontinuous since it does not have an objective value at these vectors, and this discontinuity increases the optimization difficulty, and ii) considerable computer resources can be wasted by attempting to evaluate such designs without obtaining any beneficial information. In this study it is assumed that the simulation evaluations are deterministic, namely, a SF design will consistently succeed and conversely a SI design will consistently fail.

While multiple studies have reported encountering SI designs, for example [!], the common approaches to handle them have significant shortcomings, for example they discard such vectors all together or they assign such vectors a fictitious penalized value, but both of these techniques can degrade the effectiveness of the optimization search, Accordingly, to handle such designs more effectively and efficiently this study presents an implementation in which a classifier is incorporated into the search to predict if candidate designs are likely to be SI or not. These predictions are then used by the optimization algorithm to deflect the search to designs which are likely to be SF. The performance of the proposed implementation was evaluated with an engineering test problem and results the merit of the proposed approach. The remainder of this paper is organized as follows: Section II provides the pertinent background information, Section III describes the proposed implementation, Section IV then gives a performance analysis, and lastly Section V concludes the paper.

II. Background

Simulation-driven optimization problems are increasingly common in engineering and accordingly several approaches have been studied which are tailored for such problems. One such established approach is that of using approximation models, also termed in the literature as metamodels or surrogates, which approximate the true expensive function and provide predicted objective values at a lower computational cost [1-3]. Metamodels variants include radial basis functions (RBF) and Kriging which originated in geostatistics, artificial neural networks from the domain of machine learning, and polynomial approximations from applied mathematics. During the optimization process the metamodel provides predicted objective values instead of the simulation thereby significantly reducing the overall computational cost.

With respect to the simulation-failure, SI designs have been mentioned repeatedly to in the literature, for example in references [4-8]. Given their negative impact on the optimization process several techniques have been examined to handle them, for example by assigning them a penalized value [4] or by discarding them altogether [5]. However, such techniques exhibit several demerits, namely, using a penalized vector during the metamodel training can severely degrade its prediction accuracy, while simply discarding SI vectors results in loss of valuable information. As an

Manuscript received September 28, 2018; revised May 1, 2019.

Yoel Tenne is with School of Mechanical Engineering, Ariel University, Israel (e-mail: y.tenne@ariel.ac.il).

DOI: 10.7763/IJMO.2019.V9.714

example, Figure 1 shows two Kriging metamodels which were used to approximate the Rosenbrock function: (a) shows the resultant metamodel when 30 regular vectors were used, while (b) shows the resultant metamodel when 20 penalized vectors were added to the previous training sample, with a penalized value which taken as the worst objective value from the 30 vector sample. Evidently the resultant metamodel is severely deformed with many artificial optima, and this would exacerbate the optimization difficulty.

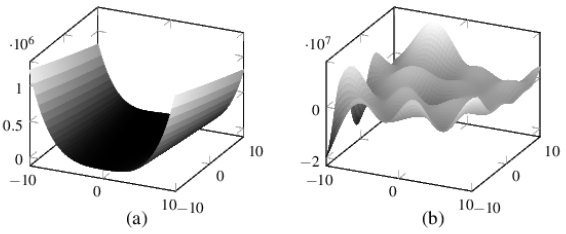

Fig. 1. Metamodels of the Rosenbrock function: (a) Based on 30 valid vectors, (b) the sample in (a) plus 20 vectors with a penalized value.

III. Proposed Implementation

To address the issues discussed above this section presents an implementation in which vectors are screened prior to any evaluation to predict if they are SF or SI. To further improve the search effectiveness the framework uses multiple search algorithms and dynamically uses the best predicted vector throughout the search. Lastly, to ensure convergence in the presence of inherent metamodel prediction errors the search in performed based on the trust-region approach [9,10].

The proposed algorithm operates in four main steps, as follows:

Step 1: A sample of vectors is generated by using the Latin hypercube sampling (LHS) method [11] to obtain a space-filling sample, as this improves the prediction accuracy of the resultant metamodel. The sample vectors are evaluated with the true expensive function and are cached in a memory storage.

Step 2: The main optimization loop now begins and a Kriging metamodel is trained by using only the SF vectors stored in the memory cache. Next, a classifier is trained by using all the cached vectors, namely, both the SF and SI ones. A classifier is a mapping from $\mathbb{R}^n \rightarrow \mathbb{N}$, namely, given a set of n-dimensional real vectors which have been associated with a set of groups or classes, then for a new vector the classifier assigns the vector an integer index which indicates to which class is this vector most likely to belong. In this study the kNN (k nearest neighbours, k=3) classifier was used [1,12].

Step 3: After training the metamodel and classifier an optimization search step is performed. Due to the high computational cost of simulation evaluations only a small number of evaluated vectors will be available and therefore the resultant metamodel will exhibit prediction inaccuracies. The latter can severely hamper the search and may even lead to convergence to a false optimum [13]. Therefore, to ensure convergence to an optimum of the true expensive function the search is performed based on the trust-region (TR) approach [14], namely, the search is restricted to the predefined volume

$$\tau: \|x - x_b\| \leq r$$

where x_b is the trust-region center and is the best vector found so far and r is the TR radius. To improve the effectiveness of the TR search step two search algorithms are used: a real-coded evolutionary algorithm (EA) [15] and a simmulated-annealing (SA) optimizer [16], and their main parameters are given in Table 1. Each of the latter is run in turn which results in two predicted optima, and the one which is predicted to have a better objective value is selected for evaluation with the true objective function, namely, the simulation.

TABLE I: Main Parameters for the Search Algorithms.

EA: Population size: 100 Max generations: 100 Selection: Stochastic Universal Selection (SUS) Recombination: Intermediate, probability=0.6 Mutation: Breeder Genetic Algorithm, probability=0.1 Elitism: 10%
Simulated annealing: Initial Temperature: $T_i = 1000\ d, d = dimnesion$ Final Temperature: $T_f = 10^{-8}$ Temperature profile: $T(t) = 0.95\ T(t-1)$ Acceptance function: $p(T) = \exp(-\frac{f(x_n)-f(x_c)}{T(t)})$

Step 4: The best solution vector found during the search, x^*, is evaluated with the true expensive function, namely, the simulation, and based on the obtained value the following updates are performed, assuming a minimization problem:

If $x^* < x_b$ then the metamodel is considered accurate since the trial step was successful, and therefore the trust-region radius is doubled.

If $x^* \geq x_b$ and the number of training vectors in the TR is below a preset threshold then the metamodel is deemed as inaccurate since the trial-step failed, but this failure is attributed to having too few training vectors in the TR. Accordingly, a new vector is sampled in TR and is added to the memory cache.

If $x^* \geq x_b$ and the number of training vectors in the TR is above the preset threshold then the metamodel is again deemed as inaccurate but now the failure is attributed to the TR being too large, namely, the prediction was made at a point which is too far from the training vectors and there the prediction was inevitably of low accuracy. Accordingly, the TR radius is halved.

As the last step in the optimization loop, the new vectors which have been evaluated in the current iteration are added to the memory cache and another loop is performed. The process repeats until a convergence criterion is reached or if the maximum allowed number of simulation calls is reached.

IV. Performance Evaluation

For its evaluation the proposed framework was applied to an engineering problem of airfoil shape optimization. In this problem the goal is to find an airfoil shape which minimizes the drag (aerodynamic friction) while maximizing the aerodynamic lift force, To ensure structural integrity a constraint was added such that the airfoil thickness (t) between 20% to 80% of its chord line had to be larger than a

critical value $t^* = 0.1$. The combined objective function used was

$$f = -\frac{c_L}{c_D} + \Omega$$

$$\Omega = \begin{cases} \frac{t^*}{t} \cdot \left|\frac{c_L}{c_D}\right| & if\ t < t^* \\ 0 & otherwise \end{cases}$$

where c_L, c_D are the lift and drag coefficients, respectively, and Ω is a penalty term which added for violation of the thickness constraint. Candidate airfoils were generated by using the Hicks-Henne method [17], namely:

$$y = y_b + \sum_{i=1}^{n} \theta_i \sigma_i$$

$$\sigma_i = \sin\left(\pi x^{(\log(0.5)/\log(\frac{i}{k+1}))}\right), \qquad i = 1 \ldots k$$

where y_b is a baseline airfoil shape, taken here as the NACA0012 airfoil, $\theta_i \in [0,1]$ are coefficients, and $\sigma_i(x)$ are the shape basis functions, where k is the user-prescribed number of basis to use. The goal is then to find the values of the coefficients θ_i which would define the optimal airfoil shape. The lift and drag coefficient of each candidate airfoil were calculated with the aerodynamic simulation code XFoil for subsonic airfoils [18]. Each evaluation required between 5 to 30 seconds on a desktop computer, thereby allowing a realistic simulation-driven optimization process to be completed in a manageable time period. Figure 2 gives the layout of the airfoil problem formulation.

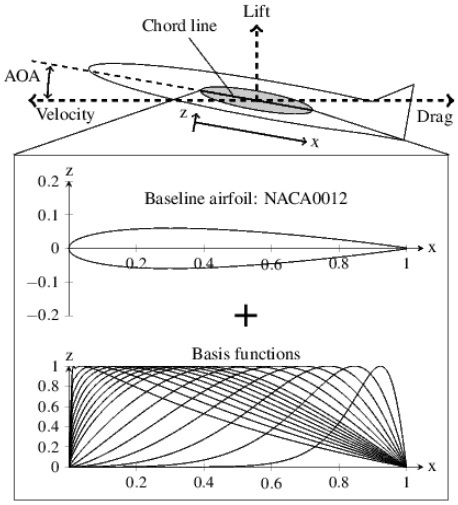

Fig. 2. The layout of the airfoil problem.

The airfoil problem has been used since it is both simulation-driven and since it contains SI vectors, namely, the Xfoil code will crash for some of the candidate airfoil designs. Particularly, the extent of SI vectors strongly depends on the angle-of-attack (AOA) parameter which is the angle between the airfoil chord line and aircraft velocity, since as the AOA increases the flow becomes more turbulent and therefore more difficult to model numerically. Accordingly, three AOA settings were used in the tests: 20, 30, and 40 degrees, to assess the performance in search spaces with an increasing density of SI vectors. Additional problem settings used were a flight speed of Mach=0.75, and a flight altitude of 32 Kft. For comparison the proposed implementation was benchmarked against two reference algorithms from the literature: i) KEAS which is an evolutionary algorithm which uses periodic sampling of new vectors to refresh a Kriging metamodel [19,20], and the EICMAES algorithm which combines the Expected Improvement approach with a covariance matrix adaption evolutionary strategies search [6]. The KEAS algorithm discarded SI vectors without incorporating them into the training set while the EICMAES algorithm in this study used the penalty approach, namely, in case a SI vector was encountered it was assigned a penalized objective value which was set to 10 times the worst objective value from the initial sample in Step 1. The number of calls to the Xfoil simulation was a limited to 200 to represent a real-world computationally expensive scenario and the size of the initial sample was 20, namely 10% of the total simulation calls. To ensure a valid statistical analysis 30 runs were repeated with each algorithm in each AOA setting, and Table 2 gives the resultant test statistics.

It follows from the test results that the proposed algorithm consistently outperformed the reference algorithms as evident from the mean and median statistics. These results are attributed mainly to the integration of the classifier into the process which diverted the search away from SI vectors, instead of using penalized vectors or simply discarding the SI thereby losing valuable information. Overall the analysis shows the merit of the proposed implementation of incorporating a machine learning component into the optimization search to handle SI vectors.

TABLE II: TEST STATISTICS

AOA [degrees]	Statistic	Proposed	KEAS	EICMAES
20	Mean	**-1.01E+1**	-0.81E+0	-0.92E+1
	Median	**-1.05E+1**	-0.78E+0	-0.84E+1
30	Mean	**-3.32E+0**	-3.02E+0	-2.89E+0
	Median	**-3.16E+0**	-2.95E+0	-2.77E+0
40	Mean	**-2.91E+0**	-2.68E+0	-2.78E+0
	Median	**-2.78E+0**	-2.54E+0	-2.71E+0

AOA: Angle of attack.
In each line the best statistic is emphasized.

V. CONCLUSION

In modern engineering it is common to use computer simulations for evaluation of candidate designs, and this setup defines an optimization problem of an expensive black-box function. In such problems there will often exist vectors for which the simulation fails without providing an output value. This in turn can degrade the search effectiveness and lead to a poor final result. To effectively and efficiently address this issue this study has presented an implementation in which a classifier is incorporated into the optimization search. The classifier's role is to predict prior to any evaluation if a candidate vector is expected to cause the simulation to fail and this prediction is then used to bias the search towards vectors for which the simulation is expected to succeed. To further improve the search effectiveness the proposed implementation used two search algorithms, a real-coded evolutionary algorithm and a simulated annealing search, such that the best vector found out of the two was used. Performance analysis based on an airfoil shape

optimization problem showed the effectiveness of the implementation presented and highlights the merit of integrating a classifier into the simulation-driven optimization search.

REFERENCES

[1] Y. Tenne and C. K. Goh, "Computational intelligence in expensive optimization problems," *vol. 2 of Evolutionary Learning and Optimization*, Berlin: Springer, 2010.

[2] F. A. C. Viana, R. T. Haftka, and L. T. Watson, "Efficient global optimization algorithm assisted by multiple surrogate technique," *Journal of Global Optimization*, vol. 56, no. 2, pp. 669–689, 2013.

[3] T. Wortmann, A. Costa, G. Nannicini, and T. Schroepfer, "Advantages of surrogate models for architectural design optimization," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 29, no. 4, pp. 471–481, 2015.

[4] K. Rasheed, H. Hirsh, and A. Gelsey, "A genetic algorithm for continuous design space search," *Artificial Intelligence in Engineering*, vol. 11, pp. 295–305, 1997.

[5] D. Büche, N. N. Schraudolph, and P. Koumoutsakos, "Accelerating evolutionary algorithms with Gaussian process fitness function models," *IEEE Transactions on Systems, Man, and Cybernetics–Part C*, vol. 35, no. 2, pp. 183–194, 2005.

[6] M. T. M. Emmerich, A. Giotis, M. Özedmir, T. Bäck, and K. C. Giannakoglou, "Metamodel-assisted evolution strategies," in *Proc. the 7th International Conference on Parallel Problem Solving from Nature–PPSN VII (J. J. Merelo Guervós, ed.)*, no. 2439, pp. 361–370, Springer, 2002.

[7] R. G. Regis and C. A. Shoemaker, "A quasi-multistart framework for global optimization of expensive functions using response surface models," *Journal of Global Optimization*, vol. 56, pp. 1719—1753, 2013.

[8] Y. Tenne, K. Izui, and S. Nishiwaki, "A computational intelligence algorithm for expensive engineering optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 5, pp. 1009–1, 2012.

[9] A. R. Conn, K. Scheinberg, and L. N. Vicente, "Introduction to derivative-free optimization MPS-siam series on optimization, philadelphia," Pennsylvania: SIAM, 2009.

[10] R. G. and Regis and C. A. Shoemaker, "ORBIT: Optimization by radial basis function interpolation in trust-regions," vol. 30, no. 6, pp. 3197–3219, 2008.

[11] M. D. McKay, R. J. Beckman, 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.

[12] S. Handoko, C. K. Kwoh, and Y.-S. Ong, "Using classification for constrained memetic algorithm: A new paradigm," in *Proc. the 2008 IEEE International Conference on System*s, Man, and Cybernetics, pp. 547–552, Elsevier, 2008.

[13] Y. Jin, M. Olhofer, and B. Sendhoff, "A framework for evolutionary optimization with approximate fitness functions," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp. 481–494, 2002.

[14] S. Grarton and L. N. Vicente, "A surrogate management framework using rigorous trust-region steps," *Optimization Methods and Software*, vol. 29, no. 1, pp. 10–23, 2014.

[15] A. Chipperfield, P. Fleming, H. Pohlheim, and C. Fonseca, "Genetic algorithm toolbox for use with MATLAB, version 1.2," *Department of Automatic Control and Systems Engineering*, University of Sheffield, Sheffield, 1994.

[16] A. Belloni, H. Liang, T. Narayanan, and A. Rakhlin, "Escaping the local minima via simulated annealing: Optimization of approximately convex functions," *The Journal of Machine Learning Research*, 40:240–265, 2015.

[17] R. M. Hicks and P. A. Henne, "Wing design by numerical optimization," *Journal of Aircraft*, vol. 15, no. 7, pp. 407–412, 1978.

[18] M. Drela and H. Youngren, *XFOIL 6.9 User Primer*, *Department of Aeronautics and Astronautics, Massachusetts Institute of Technology*, Cambridge, MA, 2001.

[19] A. Ratle, "Optimal sampling strategies for learning a fitness model," in *Proc. the 1999 IEEE Congress on Evolutionary Computation–CEC 1999*, pp. 2078–2085, IEEE, 1999.

[20] J. D. Martin and T. W. Simpson, "Use of kriging models to approximate deterministic computer models," *AIAA Journal*, vol. 43, no 4, pp.853-863, 2005.

Yoel Tenne received his PhD in mechanical engineering at Sydney University, Australia. Afterwards he was an australian endeavour fellow at the Korea Advanced Institute of Science and Technology (KAIST) and a Japan Society for Promotion of Science (JSPS) Fellow at Kyoto University, Japan. He currently a senior lecturer at Ariel University, Israel. His fields of research include applied optimization, computational intelligence, and systems engineering. In these respective fields he has authored over 50 publications,