An Enhanced Evolutionary Algorithm with a Surrogate Model

Yongsheng Lian[1], Meng-Sing Liou[2], and Akira Oyama[3]

[1] Ohio Aerospace Institute, 22800 Cedar Point Road, Cleveland, OH, 44142
[2] NASA Glenn Research Center, MS 5-11, Cleveland, OH 44135,
[3] Japan Aerospace Exploration Agency, Kanagawa 229-8510, Japan

Track Category: Hybrid Method

Abstract. In this paper we present an enhanced evolutionary algorithm (EA) to solve computationally expensive design optimization problems. In this algorithm we integrate a genetic algorithm (GA) with a local search method to expedite convergence of the GA. We first use a GA to generate a population of data by evaluating real functions, then we construct computationally cheap surrogate models based on the available data. Thereafter, we perform gradient-based local searches on the surrogate models in lieu of the real functions. We apply the GA and gradient-based method alternatively until an optimum is reached. To guarantee convergence to the original problem, we use a trust region management to handle surrogate models. We investigate the effects of number of points used to construct the surrogate model, number of surrogate model constructed, and number of local search performed. Our numerical results, based on two single-objective problems and one multi-objective optimization problem, demonstrate the advantages of the hybrid GA over pure GAs.

1 Introduction

Many real design problems in aerospace and aeronautical fields are often multi-objective in nature. To solve such problems, traditionally, we need to transform a multi-objective problem into a series of single-objective problems by introducing parameters such as weight vectors, ϵ-vectors, or target vectors [5]. Each setting of these vectors is associated with a particular Pareto-optimal solution. To find multiple Pareto-optimal solutions, we need to repetitively choose different settings and solve the resulting single-objective problem. EAs are particularly suitable for multi-objective optimization problems (MOOPs) because EA's population approach can be exploited to emphasize all non-dominated solutions in a population equally and keep an archive of a diverse set of non-dominated solutions using a niche-preserving operator. This striking feature gives EAs favorable characteristic for MOOPs because they eliminate the need to convert a MOOP into multiple single-objective problems and the need to choose different settings of parameters favoring certain Pareto-optimal solutions. However,

EAs have some problems in practical applications. Unlike gradient-based methods, which use derivatives of the objective and constraint functions for a potential reducing direction, EAs use probabilistic recombination operators to control the step size and "searching direction" in which only objective and constraint functions are evaluated. Therefore, the convergence to the optimal solution is relatively slow, and this becomes particularly severe when a local optimum is approached. Furthermore, because EAs typically require considerable numbers of function evaluations to locate an optimal solution, the required CPU time may hinder any practical application in engineering fields, which is particulary true when high-fidelity simulations are needed, such as turbomachinery blade design.

Researchers have proposed numerous approaches to improve the efficiency of EAs by leveraging their convergence rate. One approach is to incorporate stochastic EAs with deterministic gradient-based methods. The basic idea is to resort to gradient-based methods whenever the EA convergence rate is slow. This strategy takes the advantage of the fast convergence of the gradient-based methods. Muyl et al. [11] coupled a GA with a BFGS (Broyden-Fletcher-Goldfarb-Shanno [16]) method to optimize an automobile shape. To further save CPU time, one can build a surrogate model to replace the computationally expensive exact model. Liang et al. [10] coupled EAs with response surface methods; Ong et al. [12] coupled an EA with a feasible sequential quadratic programming (SQP) solver. These proposed approaches typically decompose the original problem into a sequence of subproblems confined to a small region of the design space. In each subregion, a surrogate model is constructed and optimized with gradient-based methods. These works focus on single-objective optimization.

In this paper we propose a strategy to enhance the EA performance by coupling an EA with a gradient-based method for MOOPs. We implement a real-coded GA because it is straightforward in solving our real-parameter optimization problems. The gradient-based method is a SQP solver [6]. In our approach, the GA is first used to generate a population of data by evaluating real functions, and then, surrogate models are built. Thereafter, we conduct local search on surrogate models. The GA and local search are alternately used under a trust-region framework until an optimum is found. Alexandrov et al. [1] showed that under certain assumptions, solutions produced from surrogate models under a trust region framework converge to the optimum of the original problem. By hybridizing a GA and a gradient-based method under a trust-region framework, favorable characteristics of both local search and global search are maintained. The local search determines a faster convergence. The global search assures that solution produced by an optimization algorithm working with the surrogate models, started at an arbitrary initial value, will converge to a stationary point or local optimum for the original problem.

The building blocks of our hybrid approach include a real-coded GA, a SQP solver, and a surrogate model. The coupling is fulfilled with a trust region management. Our focus is on aerospace and aeronautical optimization problems, but this method can be applied to other engineering fields as well. To our best knowl-

edge, this is the first time that hybrid GA is used in multi-objective aeronautical optimization problem. This paper is organized as follows: we first present the surrogate model construction, then we present the trust region management. In the numerical analysis section we validate our method with two single-objective optimization problems and one multi-objective optimization problem.

2 Surrogate Model

A nonlinear inequality constraint problem in general has the following form:

$$\begin{aligned} &\text{Minimize:} \quad f(\boldsymbol{x}) \\ &\text{Subject to:} \ g_i(\boldsymbol{x}) \leq 0, \quad i = 1, 2, ..., p \\ &\qquad\qquad \boldsymbol{x}_l \leq \boldsymbol{x} \leq \boldsymbol{x}_u. \end{aligned} \tag{1}$$

This problem has a set of constraints $\{g_i\}_1^p$, $\boldsymbol{x} \in R^n$ is the design variable vector, and $\boldsymbol{x}_l$ and $\boldsymbol{x}_u$ are the lower and upper bounds of the design variables, respectively. In practice, when $f(\boldsymbol{x})$ and $g_i(\boldsymbol{x})$ are computationally expensive, instead of directly solving Eq. (1), we solve the following problem:

$$\begin{aligned} &\text{Minimize:} \quad \hat{f}(\boldsymbol{x}) \\ &\text{Subject to:} \ \hat{g}_i(\boldsymbol{x}) \leq 0, \quad i = 1, 2, ..., p \\ &\qquad\qquad \boldsymbol{x}_l \leq \boldsymbol{x} \leq \boldsymbol{x}_u, \end{aligned} \tag{2}$$

where $\hat{f}(\boldsymbol{x})$ and $\hat{g}_i(\boldsymbol{x})$ are the surrogate models of $f(\boldsymbol{x})$ and $g_i(\boldsymbol{x})$, respectively. Surrogate models are built to approximate computationally expensive functions; they are computationally orders of magnitude cheaper while still provide reasonably accurate approximation to the real functions. Popular techniques include response surface methods and Kriging methods.

In constructing surrogate models, because neighboring individuals have more impact than remote ones, this motivates the use of radial basis functions in our work. Thin-plate splines (TPS) interpolation [7] is chosen for this purpose. Suppose ($\boldsymbol{x}_0$,y_0) is an arbitrary individual, around which surrogate models will be constructed, and $\{\boldsymbol{x}_i,y_i\}_1^m$ are its closest neighboring points, then the surrogate model of $f(\boldsymbol{x})$ can be constructed as follows:

$$\hat{f}(\boldsymbol{x}) = \sum_{j=1}^{m} \alpha_j ||\boldsymbol{x} - \boldsymbol{x}_j||^2 \log||\boldsymbol{x} - \boldsymbol{x}_j||. \tag{3}$$

$\{\alpha_j\}_1^m$ denote the vector of weights, which can be determined by solving a system of linear equations. Substituting $\{\boldsymbol{x}_i,y_i\}_1^m$ into Eq. (3) we get a linear system of equations:

$$\sum_{j=1}^{m} \alpha_j ||\boldsymbol{x_i} - \boldsymbol{x}_j||^2 \log||\boldsymbol{x_i} - \boldsymbol{x}_j|| = y_i, \quad i = 1, 2, ..., m. \tag{4}$$

$\{\alpha_j\}_1^m$ can be determined by solving Eq.(4). Micchelli proved that Eq. (4) has a unique solution when the set of neighboring points $\{\boldsymbol{x_i}, y_i\}_1^m$ are distinct. Parameter m determines the number of points used to construct the surrogate model, it plays an important role in the overall performance of the hybrid method. We will discuss that in the numerical analysis part. We notice that the resulting linear system of equations are ill-conditioned when some neighboring points are close to each other. This scenario becomes more frequent when a local optimum is approached. To avoid unnecessary numerical error, double-precision computations are recommended to solve Eq. (4). Surrogate models for constraint functions can be formulated in the same way.

3 A Trust Region Management

The essential ingredients of our approach include a real-coded GA, a gradient-based optimizer, and a surrogate model. A trust-region framework is used to manage surrogate models to assure that the obtained solution from the surrogate model converges to the original problem. Classical trust region algorithm uses a quadratic model based on the Taylor series expansion, which produces a good approximation confined in a small neighborhood. In engineering practice, we might use other surrogate models which have better approximations in a larger neighborhood. Previous work in this field has exclusively concentrated on whether or not the optimization technique would converge to a solution of the surrogate model, rather than the original problem. Alexandrov et al. [1] presented an approach which inherits the convergence properties of classical trust region algorithms. Under simple conditions their approach assures that the solution produced by using the surrogate models converge to a local minimum of the original problem regardless of the initial condition. Rodriguez et al. [15] extended this analysis to nonlinear programming problems with general constraints by using augmented Lagrangian methods. Giunta and Eldred [8] and Ong et al. [12] retained separate objective and constraint functions in their surrogate model application, which we will follow here. Our strategy is outlined as follows:

1. Generate a database with a GA by evaluating computationally expensive exact models.
2. Proceed by building surrogate models and perform a sequence of K local searches based on the surrogate models under the trust region framework.
3. Return to Step 1 if further iteration is needed.

In step 2 we solve a series of problems with the following form:

$$\begin{aligned} &\text{Minimize:} \quad \hat{f}^k(\boldsymbol{x}_c^k + \boldsymbol{s}) \\ &\text{Subject to:} \ \hat{g}_i^k(\boldsymbol{x}_c^k + \boldsymbol{s}) \leq 0, \qquad i = 1, 2, ..., p, \\ &\qquad\qquad ||\boldsymbol{s}|| \leq \delta^k. \end{aligned} \tag{5}$$

In the above problem, $\boldsymbol{x}_c^k$ is the starting point for the k-th local search. Surrogate models $\hat{f}^k$ and $\hat{g}_i^k$ are constructed based on the m neighboring points around

$\boldsymbol{x}_c^k$; $\boldsymbol{s}$ denotes the prospective descending step size and direction; δ^k is the trust radius; the optimal solution at the k-th search is denoted as $\boldsymbol{x}_{lo}^k$. After performing optimization on the surrogate models, we have recourse to the real functions to recalibrate trust radius by comparing the actual and predicted improvements, and then we continue our local search with surrogate models. In principle, if the surrogate models are accurate enough to predict the improvement of the exact models, we enlarge the trust radius; or if the surrogate models are not accurate, either no improvement is made or the improvement is not as much as the predicted, then we decrease the trust radius; otherwise, if reasonable but not great improvement is made, we keep the trust radius[1]. The trust radius is updated in an dynamic way based on a measure indicating the accuracy of the surrogate models. For single-objective optimization problem, this measure of merit is designated ρ^k and is calculated as follows:

$$\rho^k = \min(\rho_f^k, \rho_{gi}^k), \quad \text{for} \quad i = 1, 2, ..., p \tag{6}$$

where

$$\rho_f^k = \frac{f(\boldsymbol{x}_c^k) - f(\boldsymbol{x}_{lo}^k)}{\hat{f}^k(\boldsymbol{x}_c^k) - \hat{f}^k(\boldsymbol{x}_{lo}^k)}, \qquad \rho_{gi}^k = \frac{g_i(\boldsymbol{x}_c^k) - g_i(\boldsymbol{x}_{lo}^k)}{\hat{g_i}^k(\boldsymbol{x}_c^k) - \hat{g_i}^k(\boldsymbol{x}_{lo}^k)}. \tag{7}$$

Intuitively we can see that the larger the ρ^k is, the more accurate the surrogate models are, and we can increase our trust radius; otherwise, we need to decrease it to increase accuracy. Mathematically, the trust region size is updated based on the value of ρ^k:

$$\begin{aligned} \delta^{k+1} &= 0.25\delta^k \text{ if } \rho^k \leq 0.25, \\ &= \delta^k \quad\quad \text{if } 0.25 < \rho^k < 0.75, \\ &= \xi\delta^k \quad\quad \text{if } \rho^k \geq 0.75, \end{aligned} \tag{8}$$

where $\xi = 2$ if $||\boldsymbol{x}_{lo}^k - \boldsymbol{x}_c^k||_\infty \geq \delta^k$, or $\xi = 1$ if $||\boldsymbol{x}_{lo}^k - \boldsymbol{x}_c^k||_\infty < \delta^k$. With this measure we can update the stating point of the next iteration as follows:

$$\begin{aligned} \boldsymbol{x}_c^{k+1} &= x_{lo}^k \text{ if } \rho^k > 0, \\ &= x_c^k \text{ if } \rho^k \leq 0. \end{aligned} \tag{9}$$

In practice, we only need to solve Eq. (5) approximately. The step size is acceptable if $||s|| \leq \alpha\delta_k$ for $\alpha > 1$ independent of k [1]. In our work, we set δ as the maximal distance between the starting point $\boldsymbol{x}_c^k$ and the neighboring points, with which the surrogate model is constructed.

For multi-objective optimization problems, to construct the surrogate model, we reformulate the MOOP as a single objective problem by maintaining only one objective function while putting the others as constraints with user-specified values. This approach in the literature is called the ϵ-constraint method. We notice that for MOOPs Eq. (6) is no long a very good measure of merit for surrogate models, so we propose another approach. Basically we compare the local search solution $f(\boldsymbol{x}_{lo}^k)$ with the existing solutions, if it is located on the Pareto-optimal front, we claim improvement is made.

For unconstraint problems, to ensure convergence, we enforce consistency conditions between the original functions and the surrogates. Specifically, the following conditions must hold for both the objective and constraint functions:

$$y(\boldsymbol{x}_c^k) = \hat{y}(\boldsymbol{x}_c^k) \tag{10}$$

$$\nabla y(\boldsymbol{x}_c^k) = \nabla \hat{y}(\boldsymbol{x}_c^k) \tag{11}$$

The first condition assure surrogate model has a good approximation near $\boldsymbol{x}_c^k$, which is guaranteed when we construct the surrogate model with TPS interpolation. However, the second condition is not easy to satisfy because it is computationally prohibitive to evaluate the gradient with finite difference methods if the real function is not known.

4 Numerical Tests

We use a real-coded GA in the proposed approach. A blend crossover (BLX-α) operator is used with a value of α=0.5. In our computations, all design variables are scaled to [0 1]. We choose uniform mutation operator which adds a uniform random number to the parent solution at a probability of p_c:

$$y_i = x_i + (r_i - 0.5)\Delta_i, \tag{12}$$

where r_i is a random number, Δ_i is the user-defined maximum perturbation allowed in the i-th decision variable. We set p_c=0.1 and $\Delta_i = 0.1$ in our computations. To ensure a monotonic improvement for the GA, we adopt the elitist strategy [4] in which some of the best individuals are copied directly into the next generation without applying any evolutionary operators. For single objective optimization, sorting is based on the value of the objective function; for multi-objective optimization, we rank solutions based on Goldberg's non-dominated sorting procedure [17]. To maintain a uniform distribution on the Pareto-optimal front, we use fitness sharing [9] in the multi-objective optimization. Local search is performed with an SQP solver available in commercial optimizer DOT [6].

The first problem we test is the Rosenbrock function with two variables,

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2, \qquad 2 \le x_1, x_2 \le 2. \tag{13}$$

This function has a minimum zero locating at $(x_1, x_2) = (1, 1)$. To provide a basic comparison, we first minimize the Rosenbrock function by applying the pure GA. We set the population size as 20. The convergence history is plotted in Fig. 1. The function evaluation records the number that real function is computed in pure GA. After 800 function evaluations, GA converges to a minimum of $1.0e{-}6$. Further iterations do not significantly improve the result.

Before we discuss the computational results with the hybrid method, we study how many local searches do we need to perform on each surrogate model. Generally speaking, to maintain the advantages of both GAs and gradient-based methods in the hybrid method, we need to keep a balance between the local

search and global search. On the one hand, gradient-based methods have a fast convergence but usually are stuck in a local optimum; on the other hand, GAs have a slow convergence but enable to identify a global optimum. In general, we do not construct surrogate model around each points but only around certain selected points. We explain the reason with a 1D cartoon problem shown in Fig. 2. Four individuals are initially sampled. If we constructed surrogated models around each of these four points, we properly obtain the same function as plotted in Fig. 2. Performing local searches on the surrogate model starting from the four points will converge to the same solution locating between 8 and 10. The global minimal optimal will never be found. To avoid this, we usually divide the whole data points into groups, with each group containing ISKIP individuals. In each group, only one surrogate model is built among the ISKIP points. We compare the effects of ISKIP in Fig. 1. For the hybrid approach, the function evaluations include those computed in the pure GA and in the evaluation of Eq. (6). By ISKIP = 1 we perform local search on each individual, and we get a premature convergence; ISKIP = 5 produces some improvement; ISKIP = 10 gives the best result, which not only has a faster convergence rate but gives a better solution than the pure GA. There is no explicit expression of parameter ISKIP, which depends on the function properties, dimensionality, and population size in GA. Our experiences with other optimization problems indicate that ISKIP = 10 is a good choice. We test this function with different initial conditions. In the other four trials with different initial conditions the hybrid GA outperforms the pure GA.

In constructing surrogate models, our interest is not to fit an exact representation of the known training data itself but rather to build a statistical model to make good predictions for new inputs. Parameter m is the number of neighboring points used to construct the surrogate model. From Eq. (3) we know that m determines the complexity, or flexibility of surrogate models. A small m means an inflexible surrogate model, while one with large m means a flexible model. An inflexible model usually have a large bias, while a flexible model have a large variance (oscillation). The best situation is obtained when we have the best compromise between the conflicting requirement of small bias and small variance [2]. This can be achieved by adjusting the value of m depending on the training data distribution. However, there is no comprehensive theory to choose such an optimal value. A heuristic approach is to determine m based on the number of design variables. We compare results with different values of m in Fig. 3. Among the three solutions, $m = 4$ gives the best result, while the global surrogate model, which is constructed with all the available points, has the worst. A global model generally has the least bias but may have the largest variance because of the overfitting. This observation is consistent with that of Ong et al. [12]. In our tests we set $m = 2n$, which produces good results.

As outlined in our trust region management, parameter K controls the number of local searches performed on a surrogate model before we have recourse to real models. Excessive local searches may result in a premature convergence to a local optimal. Fig. 4 illustrates the effect of K on the convergence history. We

find $K = 1$ gives the best result. In the following tests we set up the parameters as ISKIP $= 10$, $m = 2n$, and $K = 1$.

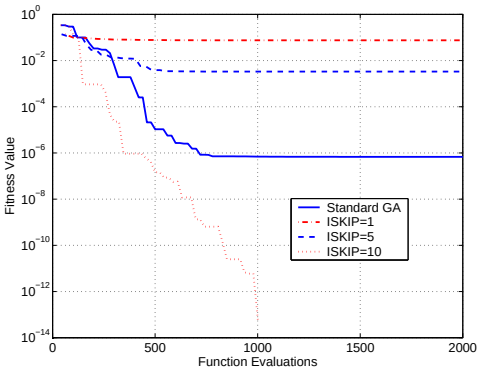

Fig. 1. Effects of cluster on convergence history

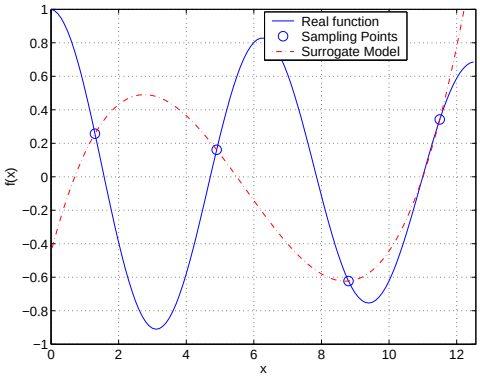

Fig. 2. Effects of cluster on convergence history

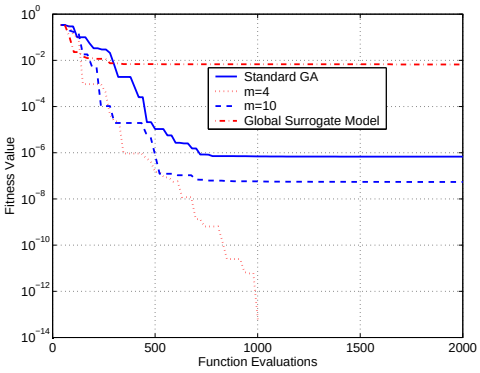

Fig. 3. Effect of number of points to construct the surrogate model

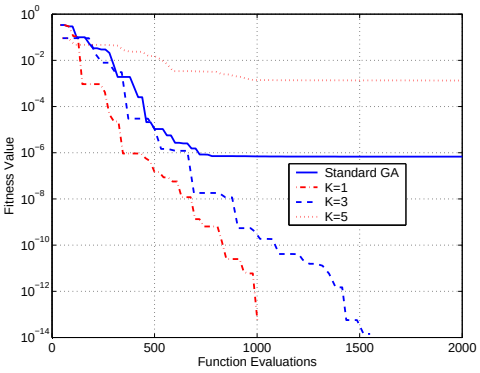

Fig. 4. Effects of number of local search on convergence history

In the second problem we minimize the Rastrigin function, which is commonly used in the global optimization literature. The Rastrigin function is defined as follows:

$$\begin{aligned} &\text{Minimize} \quad 10n + \textstyle\sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i)], \\ &\text{Subject to:} \; -5.12 \le x_i \le 5.12, \qquad i = 1, 2, ..., n. \end{aligned} \tag{14}$$

This function has a highly bumpy surface with many local optima. Fig. 5 shows the plot of a two-dimensional Rastrigin function. The function has a global minimal value of zero at $x_i = 0$. We test our proposed scheme with $n = 20$. We set the population size as 100. Here we are interested to know what are the advantages of the hybrid GA for solving this problem. We limit the total

function evaluations to 10,000. To start the local search, we let GA run five generations to generate enough data in the pool before we switch to the local search. The standard GA approaches 16 after 10,000 function evaluations, while the hybrid method converges to 8 after about 4,000 function evaluations. Other trials with different initial conditions show similar results. This test demonstrates advantages of the hybrid method for high-dimensional problems.

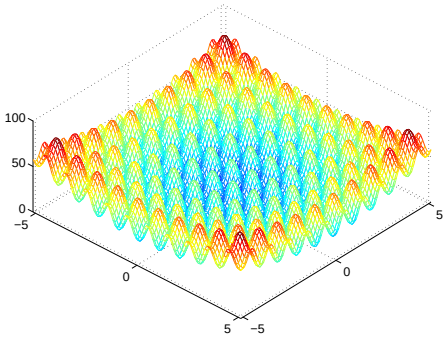

Fig. 5. Two-dimensional Rastrigin function

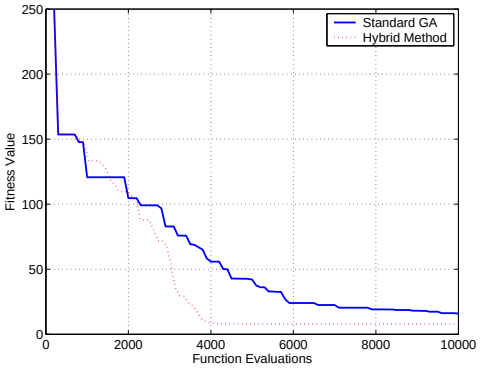

Fig. 6. Convergence history of Rastrigin function

In the third problem, we apply our approach to the design of a single-stage centrifugal pump. This problem was studied by Oyama and Liou [13] with a real-coded GA. The objectives are to maximize the total head and to minimize the input power at a design point. These objectives are competing with each other, and therefore, we have Pareto-optimal solutions. The baseline design has a shaft rotative speed of 5,416.7 rpm, total temperature of the fluid entering the pump of 29.6°C, total pressure of the fluid entering the pump of 35,200kgf/m^2, and a mass flow rate of 85.67 kg/s.

In the single-stage design, there are 11 design variables. They are the rotor leading-edge tip radius R_{tip1}, rotor trailing-edge radius R_2, volute tongue radius R_3, blade span at trailing edge B_2, blade span at volute tongue B_3, axial length of the blade at the rms diameter S, number of blades Z_n, blade thickness thk, blade trailing-edge angle at the hub, rms radius, and tip (β_{hub}, β_{mid}, β_{tip}). The design spaces are tabulated in Table 1.

We set the population size as 120 at each generation. The computation terminates after 10,000 function evaluations. The result is shown in Fig. 7 together with the original design. The comparison between a GA and a hybrid GA is not as straightforward as the single objective optimization because the final solution is a Pareto-optimal front. Suppose we obtain q solutions on the Pareto-optimal front in our final result, we can evaluate the performance of our hybrid method

Table 1. Design parameter spaces for the centrifugal pump design problem

Design Variables	$R_{1,tip}$ in.	R_2, in.	R_3, in.	B_2, in.	B_3, in.	
Lower bound	3.40	5.00	5.60	0.70	0.85	
Upper bound	4.00	5.60	6.20	0.85	1.00	
Design Variables	S, in.	β_{hub}, Deg.	β_{rms}, Deg.	β_{tip}, Deg.	thk, in	Z_{n2}
Lower bound	3.70	25.0	25.0	25.0	0.03	4
Upper bound	4.30	45.0	45.0	45.0	0.10	30

based on the generational distance G [18]:

$$G = \frac{\sqrt{\sum_{i=1}^{q} d_i^2}}{q} \tag{15}$$

where d_i is the distance between a Pareto-optimal solution and its nearest point at the true Pareto-optimal front. a true Pareto-optimal front is not available, we approximate it by using a GA with a population size of 400 and generation size of 300.

Fig 8 depicts the generational distance of the standard GA and the hybrid GA. The surrogate model starts from the sixth generations. The standard GA has a quicker convergence at the early stage and then it slows down after 4,000 function evaluations. The hybrid scheme catches up with the pure GA after 4,500 function evaluations. The superiority continues until the end. The convergence rate also slows down after about 6,000 function evaluations. Other computations with different initial condition also confirm the advantages of our hybrid approach.

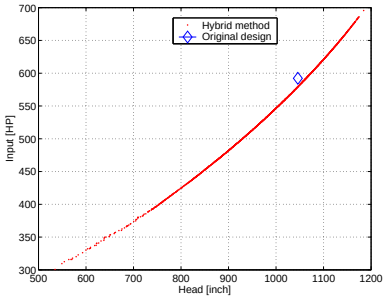

Fig. 7. Objective function values of the centrifugal pump designs.

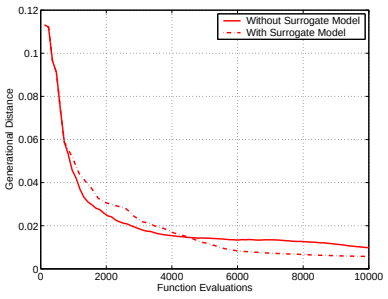

Fig. 8. Generational distance histories.

Our proposed method aims at solving computationally expensive real problems in aerospace and aeronautical fields. In such problems, the CPU time used to construct surrogate models and perform local searches are much less than

that used for real function evaluations. Therefore, we can assume that the total CPU time is exclusively determined by the number of function evaluations. By that token, the hybrid approach requires less CPU time than the pure GA.

5 Conclusions

An enhanced evolutionary algorithm was proposed to solve computationally expensive optimization problems. This algorithm hybridizes a GA with a local search method. Numerical tests demonstrated that this algorithm maintains the advantages of both local search and global search algorithms, i.e., a fast convergence rate and a global optimal property. Detailed studies were also conducted on the impacts of surrogate model accuracy, local search frequency, and local search number.

6 Acknowledgements

The authors thank the Computational Thermo-Fluids Laboratory at the University of Florida by providing the computational resources. This work is supported by NASA research grand NAG3-2869, under the Ultra Efficient Engine Technology Program.

References

1. Alexandrov, N. M., Dennis, Jr., J. E., Lewis, R. M., and Torczon, V.: A Trust-region Framework for Managing the Use of Approximation Models in Optimization. Structural Optimization. **15** (1998) 16–23
2. Bishop, C. M.: Neural Network for Pattern Recognition. Oxford University Press (2003)
3. Booker, A., J., Dennis, Jr., J. E., Frank, P. D., Serafini, D. B., Torczon, V., and Trosset, M. W.: A Rigorous Framework for Optimization of Expensive Functions by Surrogates. Structural Optimization. **17** 1–13
4. De Jong, K. A.: An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Ph.D dissertation, University of Michigan, Ann Arbor (1975)
5. Deb, K.: Milti-Objective Optimization using Evolutionary Algorithm. John Wiley & Sons, Chichester (2001)
6. DOT User's Manual. Version 4.20. Vanderplaats Research & Development, Inc., Colorado Springs, CO (1995)
7. Duchon, J. P.: Splines Minimizing Rotation-Invariant Semi-Norms in Sobolev Spaces. In: Schempp, W., Zeller, K. (eds): Constructive Theory of Functions of Several Variables. Springer-Verlag, Berlin (1977) 85–100
8. Giunta, A. A., and Eldred, M. S.: Implementation of a Trust Region Model Management Strategy in the DAKOTA Optimization Toolkit. AIAA Paper 2000-4935
9. Goldberg, D. E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA (1989)
10. Liang, K. H., Yao, X., and Newton, C.: Evolutionary Search of Approximated N-dimensional Landscapes. International Journal of Knowledge-based Intelligent Engineering Systems. **4** (2000) 172–183

11. Muyl, F., Dumas, L., and Herbert, V.: Hybrid Method for Aerodynamic Shape Optimization in Automotive Industry. Computers & Fluids. **33** (2004) 849–858
12. Ong, Y. S., Nair, P. B., and Keane, A.: Evolutionary Optimization of COmputationally Expensive Problems via Surrogate Modeling. AIAA Journal. **41** (2003) 687–696
13. Oyama, A., and Liou, M. S.: Multiobjective Optimization of Rocket Engine Pumps Using Evolution Algorithm. Journal of Propulsion and Power. **18** (2002) 528–535
14. Ratle, A.: Kriging as a Surrogate Fitness Landscape in Evolutionary Optimization. Artificial Intelligence for Engineering Design Analysis and Manufacturing. **15** (2001) 37–49
15. Rodríguez, J., Renaud, J. E., and Watson, L. T.: Convergence of Trust Region Augmented Lagrangian Methods Using Variable Fidelity Approximation Data. Structural Optimization. **15** (1998) 1–7
16. Shanno, D. F.: Conditioning of quasinewton methods for function minimization. Math. Comput. **24** (1970) 647-664.
17. Srinivas, N., and Deb, K.: Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Evolutionary Computation. **12** (1994) 221-248
18. Van Veldhuizen, D. A., and Gary, B. L.: Evolutionary Computation and Convergence to a pareto Front. In: Koza, J. R. (eds): Late Breaking Papers at the Genetic Programming 1998 Conference. Stanford University, Stanford (1998) 221–228