ASAGA: An Adaptive Surrogate-Assisted Genetic Algorithm

Liang Shi
Computer Science Department, University of Georgia
538 Boyd Graduate Studies Research Center,
Athens, GA 30602
706-583-0828
shi@cs.uga.edu

Khaled Rasheed
Computer Science Department, University of Georgia
219B Boyd Graduate Studies Research Center,
Athens, GA 30602
706-542-3444
Khaled@uga.edu

ABSTRACT
Genetic algorithms (GAs) used in complex optimization domains usually need to perform a large number of fitness function evaluations in order to get near-optimal solutions. In real world application domains such as the engineering design problems, such evaluations might be extremely expensive computationally. It is therefore common to estimate or approximate the fitness using certain methods. A popular method is to construct a so called surrogate or meta-model to approximate the original fitness function, which can simulate the behavior of the original fitness function but can be evaluated much faster. It is usually difficult to determine which approximate model should be used and/or what the frequency of usage should be. The answer also varies depending on the individual problem. To solve this problem, an adaptive fitness approximation GA (ASAGA) is presented. ASAGA adaptively chooses the appropriate model type; adaptively adjusts the model complexity and the frequency of model usage according to time spent and model accuracy. ASAGA also introduces a stochastic penalty function method to handle constraints. Experiments show that ASAGA outperforms non-adaptive surrogate-assisted GAs with statistical significance.

Categories and Subject Descriptors: G.1.6
[**Optimization**]: Constrained optimization, Global optimization; I.2.8 [**Artificial Intelligence**]: Miscellaneous-Evolutionary computing and genetic algorithms

General Terms: Algorithms, Performance, Experimentation.

Keywords
Genetic algorithms, Surrogate-assisted evolution, Adaptive Meta-modeling, Fitness approximation

1. INTRODUCTION
Genetic algorithms have proven repeatedly to be powerful for solving optimization problems and thus are used in a wide range of real-world applications such as engineering design domains. In such domains, it is not uncommon to see that fitness functions are discontinuous, non-differential, highly multi-modal, noisy and ambiguous [1, 2]. It is therefore not surprising that GAs usually perform better than conventional optimizers such as sequential quadratic programming and Simulated Annealing [1, 4]. Many challenges are still faced in the application of GAs to real-world domains. For engineering design problems, a large number of objective evaluations may be required in order to obtain near optimal solutions. Moreover, the search space can be complex with many constraints and a small feasible (physically realizable) region. However, determining the quality (fitness) of each point may involve the use of a simulator or an analysis code that takes an extremely long time. Therefore it is impossible to be cavalier about the number of objective evaluations used for an optimization [2].

For such problems surrogate-assisted evolution methods based on fitness approximation are preferable as they can simulate the exact fitness with much less computational cost. A good fitness approximation method can still lead the GA process to find optimal or near-optimal solutions and is also tolerant to noise. The reader can refer to Jin et al. [3, 6] for details. A problem with fitness approximation methods arises because it is difficult to determine which approximate model is appropriate and how often the model should be used in each domain. Even for one domain, the answer is not usually fixed but varies with the different stages of the optimization process. A fixed approximate model cannot possibly fit well in all kinds of optimization problems as well as all stages.

In this paper we present ASAGA (Adaptive Surrogate-Assisted Genetic Algorithm). ASAGA has a skeleton of GADO [1, 2], which is a GA that proved to be powerful for solving engineering design problems. ASAGA uses global and local approximate models by using a clustering technique with one local model for each cluster. The global model can adaptively evolve from a simple average of the fitness of all individuals in a population, all the way to a Support Vector Machine (SVM) model. The model evolution depends on time spent and the model accuracy. The local models follow similar rules except that the final model is the quadratic model (not the SVM) since each local model is based on a sample set which is not so large. The number of clusters is also adaptively controlled according to the type of model currently used, the model accuracy, and the size of the cluster.

In ASAGA, the frequency of model usage is also adaptively adjusted according to the time spent on each approximate as well

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

as actual fitness evaluation and model accuracy. Intuitively, if the approximate model formation and usage eat up a lot of the total optimization time, the frequency should be reduced. On the other hand, if the approximate model fits very well, the frequency should be increased. A formula that takes into account both factors and reaches a trade-off point is applied in ASAGA.

The remainder of this paper is organized as follows. In section 2, several kinds of fitness approximation methods and comparative studies of the different approximate models are given. In section 3, the proposed approach is described in detail. In section 4, experimental results are presented. The paper is concluded in Section 5 with a discussion of open questions and future work.

2. FITNESS APPROXIMATION METHODS AND COMPARATIVE STUDIES

2.1 Machine Learning and Statistical Learning Methods

So far many Machine Learning and statistical learning methods have been used for fitness approximation in GAs. In this paper, several of the most popular ones are briefly reviewed next.

2.1.1 Fitness Inheritance

One approach is to set the fitness of a child solution to the average fitness of its parents or a weighted average based on how similar the child is to each parent. A re-sampling method combined with a simple average fitness inheritance method is used [10]. Another approach is to divide the population into building blocks according to certain schemata. Then an individual gets its fitness from the average fitness of all the members in each building block which this individual belongs to [8].

2.1.2 Radial Basis Function Model (RBF)

The RBF model is popular for fitness approximation [4, 14]. An RBF network consists of an input layer with the same number of input units as the problem dimension, a single hidden layer of k nonlinear processing units and an output layer of linear weights w_i. The output $y(x)$ of the RBF network is given as a liner combination of a set of radial basis functions expressed as:

$$y(x) = w_0 + \sum_{i=1}^{k} w_i \phi_i(\| x - c_i \|) \tag{1}$$

where w_0 and w_i are the unknown coefficients to be learned. The term $\phi_i(\| x - c_i \|)$, also called the kernel, represents the ith radial basis function. A Gaussian kernel is the most commonly used in practice having the form:

$$\phi_i(\| x - c_i \|) = \exp\left(-\frac{\| x - c_i \|^2}{2\sigma^2}\right) \tag{2}$$

A detailed comprehensive description of RBF networks can be found in [26].

2.1.3 Clustering Techniques

Classic clustering algorithms include hierarchical clustering, partition clustering (such as k-means algorithm), and overlapping clustering. Among them, the k-means algorithm appears to be the most popular one applied in GAs due to its relative simplicity and low computational cost. In [11], the whole population is divided into many clusters. Only the center of each cluster is evaluated. Other individuals' fitness values in the clusters are computed using the distance from their centers. Another approach is to build an approximate model based on sample points composed of the cluster centers. Then every other individual's fitness is estimated by the approximate model [13]. One important clustering technique applied in EA is to divide the population into several clusters, and then build an approximate model for each cluster. Multiple approximate models are believed to gain more local information about the search space and fit better than a single model [21]. ASAGA also uses this method.

2.1.4 Multilayer Perceptron Neural Network (MLPNN)

An MLPNN model is used to accelerate the convergence by replacing the original fitness function [24]. In engineering design domains, MLPNN is used to reduce the complex fitness function evaluation times [20]. A simple feed-forward MLPNN with one input layer, one hidden layer and one output layer can be expressed as:

$$y(x) = \sum_{j=1}^{K} w_j f\left(\sum_{i=1}^{n} w_{ij} x_i + \theta_j\right) + \theta_0 \tag{3}$$

where n is the number of input neurons (which is usually equal to the problem dimension), K is the number of nodes of the hidden layer, and the function f is called the activation function which is most commonly the logistic function. W and θ are the unknown weights to be learned. The reader can refer to Bishop et al. [26] for a comprehensive study.

2.1.5 Polynomial Models

Polynomial models are also called Response Surface methods. Commonly used quadratic polynomial models have the form:

$$\hat{F}(\overline{X}) = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{i=1,j=1}^{n,n} a_{ij} x_i x_j \tag{4}$$

where a_0, a_i and a_{ij} are the unknown coefficients to be fitted, n is the dimension of the problem.

Usually the least-squares approximation method is used to fit the unknown coefficients. The main limitation of the least-squares method is that the number of sample points N must exceed (n+1)(n+2)/2 for a second-order polynomial model. Even if this condition is satisfied, the fitting cannot be guaranteed because the singularity problem will still happen if two sample points are too close to each other. Another drawback of the least-squares method is that its computational complexity grows fast with the problem dimension. The gradient method is introduced to address the problems of the least-squares method. More implementation details for these two methods for fitting a polynomial model can be found in [18].

2.1.6 Kriging Models

The Kriging model is used to build the global models [14] and it is used to accelerate the optimization by reducing the expensive computational cost of the original fitness function [25]. A Kriging model can be mathematically expressed as:

$$y(x) = \beta + Z(x) \tag{5}$$

where β is usually a constant representing a global model and $Z(x)$ is the realization of a stationary Gaussian random function that creates a localized deviation from the global model. The estimated model in equation (5) is given as:

$$\hat{y} = \hat{\beta} + r^T(x) R^{-1}(y - f\hat{\beta}) \tag{6}$$

where f is a column vector which is filled with ones, and R is the correlation matrix which can be obtained by computing the correlation function which is often a Gaussian exponential correlation function. The Kriging method can give the accuracy information about the fitting by showing the confidence intervals of the estimated values without extra computational cost. One disadvantage of this method is that it is sensitive to the problem's dimension.

2.1.7 Support Vector Machines (SVM)

Good features of SVM are that it is not sensitive to local optima and that the optimization process does not depend on the problem dimension and seldom goes overfitting. A detailed description of SVM can be found in [9]. Applications of SVM for fitness approximation can be found in [17]. The epsilon-SVM regression model is commonly used for fitness approximation, where the linear epsilon-insensitive loss function is defined by:

$$L^{\varepsilon}(x, y, f) = |y - f(x)| = \max(0, |y - f(x)| - \varepsilon) \tag{7}$$

The sum of the linear epsilon-insensitive losses must be minimized which is equivalent to a constrained minimization problem having the form:

$$\frac{1}{2} w^T w + C \sum_{i=1}^{N} \zeta_i + C \sum_{i=1}^{N} \zeta_i^* \tag{8}$$

subject to the following constraints:

$$w^T \phi(x_i) + b - y_i \le \varepsilon + \zeta_i^*, \quad y_i - w^T \phi(x_i) - b_i \le \varepsilon + \zeta_i$$

$$\zeta_i, \zeta_i^* \ge 0, i = 1, \ldots, N \tag{9}$$

where $\phi(x)$ is called the kernel function.

Some approximation methods are combined for constructing the fitness approximation models in some earlier applications. In [13] the MLPNN model is combined with clustering methods for constructing approximate models. In [14, 23], the Kriging method is used for constructing the global approximate model for pre-selection and then RBF models are built upon those pre-selected sample points for further fitness approximation.

2.2 Comparative Studies for Approximate Models

The neural network model and the polynomial model were compared in [19]. The author concluded that the performance of the two types of approximation was comparable in terms of the number of functional evaluations required to build the approximations and the number of undetermined parameters associated with the approximations. However, the polynomial model had much less construction cost. In [21], a quadratic polynomial model was found to be the best among the polynomial model, RBF network, and the Quickprop neural network when the models are built for regions created by clustering techniques. The authors were in favor of the polynomial model because they found that the polynomial model formed approximations more than an order of magnitude faster than the other methods and did not require any tuning of parameters. The authors also pointed out that the polynomial approximation was in a mathematical form which could be algebraically analyzed and manipulated as opposed to the black-box results that neural networks give.

The Kriging model and the neural network model were compared using benchmark problems [27]. However, no clear conclusion was drawn about which model is better. Instead, the author showed that the optimization with a meta-model could lead to degraded performance. Another comparison was presented in [22] between the polynomial model and the Kriging model. By testing these two models on a real-world engineering design problem, the author found that the polynomial and the Kriging approximations yielded comparable results with minimal difference in predictive capability. Comparisons between several approximate models were presented in [7], which compared the performance of the polynomial model, the multivariate adaptive splines, the RBF model, and the Kriging model using fourteen test problems with different scales and nonlinearities. The conclusion was that the polynomial model is the best for low-order nonlinear problems and the RBF model is the best for dealing with high-order nonlinear problems.

Model performance may depend on the problem to be addressed and more than one criterion needs to be considered. The model accuracy probably is the most important criterion since low accuracy approximate models may lead the whole optimization process to inferior optima. The model accuracy also should be based on new sample points instead of the training data set points. The reason for this is that for some models such as the neural network, an overfitting problem is common. An overfitting model works very well on training data yielding good model accuracy though it might work badly on new sample points. Other criteria are also important including robustness, efficiency, and the time for model construction and updating. A good comparison would consider the model accuracy as well as all of those criteria.

It is difficult to draw a clear conclusion on which model is the best for the reasons stated above, though the polynomial model seems to be the best choice for a local model when we are dealing with local regions or clusters and have enough sample points [21]. In such cases, the fitting problem usually has low-order nonlinearity and the polynomial model is the best candidate according to Jin et al. [7]. Because of the good features of SVM (see above), the SVM model becomes a very good choice for constructing the global model, especially when the problem has a high dimension, many local optima, and enough sample points.

3. THE PROPOSED METHOD

ASAGA uses a skeleton of GADO (Genetic Algorithm for Design Optimization), a GA that was designed with the goal of being suitable for use in engineering design. It uses new operators and search control strategies that target the domains that typically arise in such applications. GADO has been applied in a variety of optimization tasks that span many fields [1, 2]. It demonstrated a great deal of robustness and efficiency relative to competing methods.

In GADO, each individual in the GA population represents a parametric description of an artifact, such as an aircraft or a missile. All parameters take on values in known continuous ranges. The fitness of each individual is based on the sum of a proper measure of merit computed by a simulator or some analysis code, and a penalty function if relevant. The penalty function consists of an adaptive penalty coefficient multiplied by the sum of all constraint violations if any. A steady state GA model is used, in which operators are applied to two parents selected from the elements of the population via a rank based selection scheme, one offspring point is produced, and then an existing point in the population is replaced by the newly generated point via a crowding replacement strategy. Floating point representation is used. Several crossover and mutation operators are used, most of which were designed specifically for the target domain type. GADO also uses a search-control method [2] that saves time by avoiding the evaluation of points that are unlikely to correspond to good designs. GADO was further improved by adding a surrogate-assisted component which forms approximate models and uses them for speedup through informed genetic operators. These operators are described in detail in [16]. The main idea is to make the genetic operators such as mutation and crossover more informed using approximate models. In every place where a random choice is made, for example when a point is mutated, instead of generating just one random mutation we generate several, rank them using the approximate model, then take the best to be the result of the mutation. GADO includes informed mutation, crossover and initial population formation.

3.1 Adaptive Approximate Model Selection

ASAGA utilizes both the global and local model structure. It fits one global approximate model for the optimization function and many local models for different regions (clusters). One advantage of this structure is that local models can fit much better since the approximate model will be defined over a smaller, less complex search region. Thus more local details can appear in successive approximations. On the other hand, the global model catches more global information and may prevent the search from falling into certain local optima. Ideally all the points previously evaluated in the course of the optimization should be used to construct the approximations. However, if the number of iterations is very large or to save time, a large enough sample (the default value is 2000) of evaluated points can be used to construct the approximate models. A good feature of ASAGA is that all current individuals hold exact fitness and approximate models are used for informed operators and initialization and never directly replace the original fitness. Thus it prevents the searching process from converging to false optima.

Based on the discussion in section 2.3, we know that the polynomial model is a good candidate for low-order nonlinearity problems; thus it becomes a good choice for the local approximate models. Because of good features of SVM (described in 2.1.7), it is believed to be a good choice for the global optimization model. In this paper, the SVM model is implemented using the SVM^{light} package. The implementation details can be found in [15].

ASAGA provides an adaptive mechanism to switch between different approximate models. Here the switching is one way, upgrading from lower to higher complexity only. The Global model upgrade path is shown in Fig. 1. The local model upgrade path is similar to the global one except that a local model never reaches the SVM model. The upgrade is triggered by a threshold. A formula calculates a value taking into account the model accuracy and the time spent at the approximation phase. If this value is less than a certain threshold, an upgrade happens. The formula is:

$$v = R \times (1 + T_1 / T_2) \tag{10}$$

where R is the correlation coefficient between the exact fitness and the approximate fitness, T_1 is the time spent at the approximation phase and T_2 is the total optimization time.

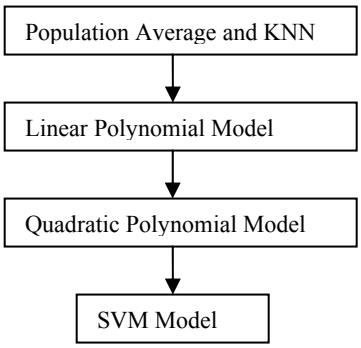

Figure 1. Approximate model upgrading path for global models.

The reason the correlation coefficient is used to measure the model accuracy is because of the way ASAGA uses approximate models. In ASAGA, informed operators (described above) are used as the approximate model incorporation mechanism. Because of the way informed operators work, the absolute difference between the approximate fitness and exact fitness is not important, whereas the good correlation between them is what really matters for informed initialization, crossover and mutation. The correlation coefficient equation is given by:

$$R = \frac{\sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \overline{x})^2 \sum_{i=1}^{n} (y_i - \overline{y})^2}} \tag{11}$$

where x is a vector of all exact fitness values for the sample points, and y is the corresponding estimated fitness values given by an approximate model. R is between -1 and +1. In our proposed algorithm, the R is usually between 0.5 and 1 for fitness models and between 0 and 1 for violation models, since approximate models fit very well.

According to the formula (10), T_1 / T_2 is between 0 and 1 but never reaches 1, in practice, T_1 / T_2 seldom exceeds 0.5. So the threshold is usually between 0.5 and 1.5 for fitness models and is between 0 and 1.5 for violation models. In order to choose a good threshold we run a set of experiments with different thresholds. We use a 5 dimension Ackley's function as the following:

$$f_{Ackley}(x) = -20\, e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)} + 20 + e$$
$$-30 \le x_i \le 30 \tag{12}$$

The average best fitness for 30 runs with different random starting populations is reported with the corresponding number of function evaluations in Fig. 2. From the figure we can see that all the lines have little difference (logarithm scale), whereas the line representing a threshold 0.9 stays at the bottom most of the time reaching the best final value. It was further found that any threshold larger than 0.95 performs the same as 0.95 and any threshold smaller than 0.65 performs the same as 0.65. Thus the threshold was set to 0.9 for objective function models. Following the same process, the threshold was set to 0.5 for the constraint violation function models in the experiments. Figure 3 shows the performance for three threshold values. The figure shows that a threshold of 0.5 yields the best result. The model update is triggered if the v value in (10) is smaller than the pre-set threshold.

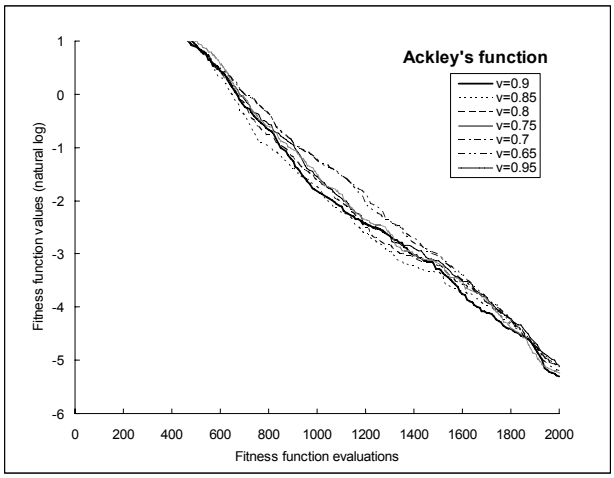

Figure 2. ASAGA performance with different thresholds tested by a 5D Ackley's function.

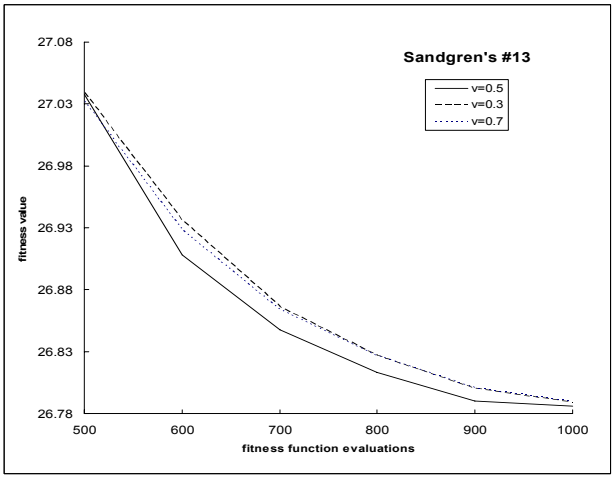

Figure 3. ASAGA performance with different thresholds for constraint violations

3.2 Adaptive Clustering Techniques

Since clusters are introduced in ASAGA, determining the number of clusters becomes a critical issue. Dividing the optimization trace (i.e. all the points evaluated using the actual fitness function so far) into too many clusters may lead to failure of the local model fit because too few sample points may restrict us to a model that does not capture the local space information well. ASAGA uses least-squares approximation to develop polynomial models for the local model construction. In addition, too few sample points may result in a singularity problem. On the other hand, too few clusters loose the benefit of the global and local model system. The large clusters are still complex and difficult to fit with any approximate model.

To solve this problem, ASAGA uses an adaptive clustering technique to adjust the number of clusters during the optimization. After 2 times the population size of fitness function evaluations, (i.e. when the number of sample points reaches double the population size), the first two clusters are created by splitting the current sample points to 2 clusters with the K-means algorithm. Then it will be periodically determined if the number of clusters needs to be increased based on two factors. The first factor is the requirements of fitting a polynomial model. The minimal number of sample points needed for fitting a linear polynomial model is (n+1). It is (n+1)(n+2)/2 for fitting a quadratic polynomial model. In practice, these minimal numbers still need to be increased to avoid the singularity problem. In ASAGA 1.5 times the minimal numbers are applied. To be eligible for splitting, a cluster should have 3.5 times the minimal required number of sample points to ensure that the resulting sub-clusters still have enough points to fit models. The second factor is the correlation coefficient R introduced in formula (11) above.

The number of clusters is increased when a cluster satisfies two conditions:

1. The cluster has sample points more than 3.5 times the minimal requirement for its current model.
2. The R of this cluster is lower than certain threshold, in ASAGA, 0.9 for an objective function and 0.5 for a violation function.

The reason for using the number "3.5" is that we found it to be the minimal number for which the singularity problem does not happen. The splitting is recursively performed (i.e., the resulting sub-clusters can be further split) as long as they satisfy the two conditions. These two conditions are checked periodically at fixed intervals of actual fitness evaluations. The interval is usually set to the population size.

3.3 Adaptive Approximate Model Usage

ASAGA uses informed operators guided by approximate models in the same way as GADO, its ancestor uses them. In informed mutation for example, a number of random mutations are generated as candidate children and ranked using the approximate model. Then their best according to that ranking is evaluated using the actual fitness function. However, GADO with surrogate -assisted uses a fixed number of such random mutations. A fixed number is not suitable for all problems. Too few mutations will not get the full benefit of the usage of the approximate model, whereas too many mutations will greatly increase the computational cost for little or no additional gain. The same argument holds for informed crossover and initialization. ASAGA uses an adaptive number of candidate children based on a formula considering the model accuracy and computational cost simultaneously. The formula is as follows:

$$N = Round \left(\frac{R}{C \times (T_1 / T_2)} \right) \quad (13)$$

where N is the number of the potential children with minimum 1. R, $T1$ *and* $T2$ are defined in the same manner as in (10). C is a small constant which was set to 0.2 based on experimental results. Fig. 4 shows ASAGA performances on Ackley's function (12) with different parameter C settings. The curves have little difference but the line C=0.2 stays at the bottom most of the time and reaches the minimum value among all three lines at the end.

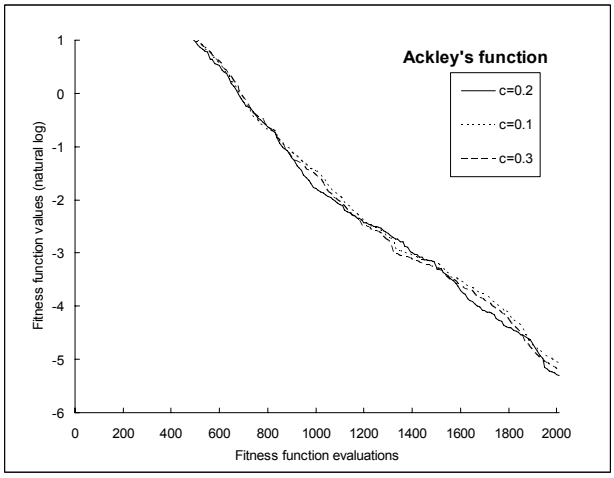

Figure 4. ASAGA performances with different parameter *C* tested by a 5D Ackley's function.

3.4 Stochastic Constraint Handling

ASAGA inherits the penalty function method used in GADO to deal with the constraints. In GADO, the fitness is the sum of the objective function value and a penalty coefficient multiplied by the violation function [1]. The violation function is the sum of the violations of all the violated constraints. The penalty coefficient is adaptively changed according to the feasible points found so far. ASAGA makes a further improvement on this penalty function method by making this fitness assignment stochastic. ASAGA randomly decides, with a small probability, to switch from the normal method above to a method in which only the objective function value or only the violation function value is considered instead of both. When it chooses to switch, then it checks if any feasible points have been found. If so, then the objective function alone is considered the fitness. Thus there is more pressure on objective improvement. On the other hand, if feasible points have not been found, ASAGA uses only the violation function as fitness. Thus there is more pressure on finding feasible points.

4. EXPERIMENTS AND RESULTS

In this section, ASAGA is tested on benchmark functions with constraints. In every experiment the population size was set to its default value of 10 times the problem dimension. The first four domains were introduced by Eric Sandgren in his Ph.D. thesis [5] in which he presented 30 engineering design optimization problems. The fifth experimental domain comes from a real engineering design problem called Welded Beam Design previously used as benchmark in several studies such as in [12]. We compared ASAGA to three methods: GADO, GADO with surrogate assist (GADO-R), and ASAGA without SVM model (ASAGA-NSVM).

GADO was used with no approximate model assist. GADO-R is based on GADO with a global and local model structure by clustering technique. It incorporates fixed approximation models which are quadratic polynomial models through informed operators [2, 16]. ASAGA-NSVM is similar to the proposed ASAGA except that its final model stops at Polynomial and will never be upgraded to the SVM model. All four methods ran 30 times with different random starting populations on each problem. The average best fitness values of the 30 runs with corresponding number of actual fitness evaluations are shown.

Figures 5 through 9 show the performance of these four methods. Notice that only feasible regions are shown. So the search traces in these figures do not start from the beginning since certain time is needed to find feasible solutions. In Sandgren's domain number 3, the ASAGA-NSVM and ASAGA have almost the same performance but both outperform GADO-R and GADO. In Sandgren's domains number 13, 21 and 22, ASAGA is obviously the winner among all four methods. Notice that the ASAGA-NSVM also outperforms the other two methods which have no adaptive fitness approximation or usage. In the Welded Beam design problem, ASAGA-NSVM performs the best where ASAGA is slightly worse than the ASAGA-NSVM but performs much better than the other two methods. Based on these experiments we can draw a conclusion: the surrogate-assisted GA performs better than the GA with no surrogate assist at all but the adaptive surrogate-assisted GA outperforms the non-adaptive surrogate-assisted GA.

Table 1 shows the statistical analysis and performance comparison of GADO-R which is a fixed surrogate-assisted GA and ASAGA which is an adaptive surrogate-assisted GA. A T-test is conducted in the five test domains. A T-test value less than 0.05 suggests a statistically significant difference between two objects. In this experiment, all five p-values are less than 0.05, some are even less than 0.01, which suggests ASAGA performs better than GADO-R with statistical significance.

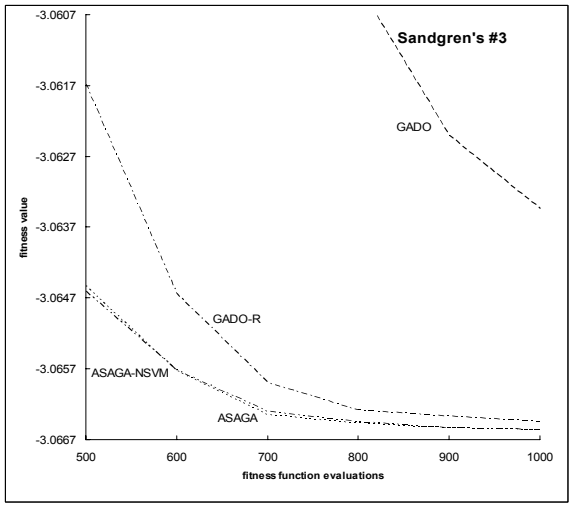

Figure 5. Sandgren #3 with global optima -3.0666.

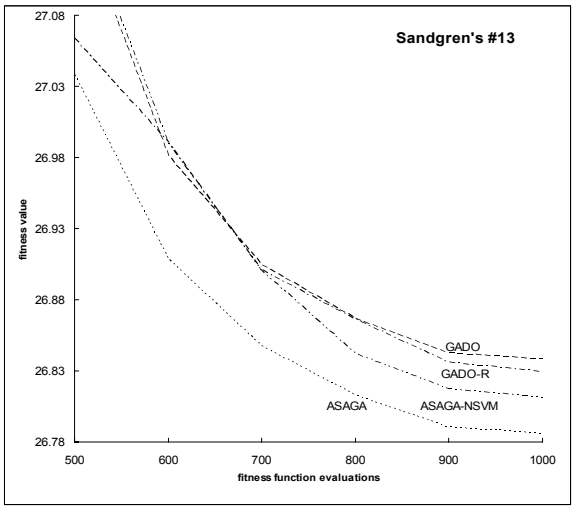

Figure 6. Sandgren #13 with global optima 26.78.

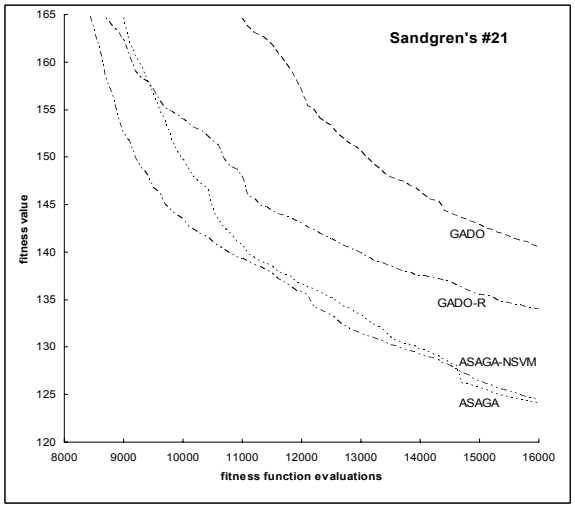

Figure 7. Sandgren #21 with global optima 97.5.

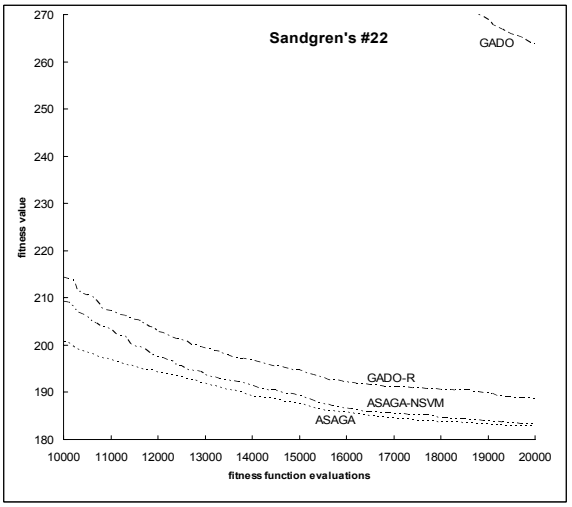

Figure 8. Sandgren #22 with global optima 174.7.

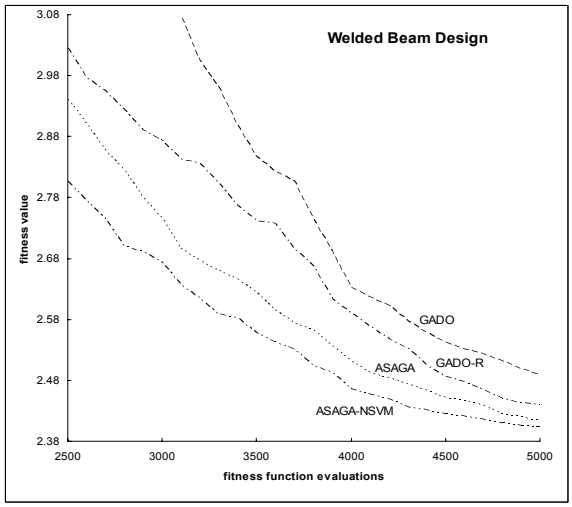

Figure 9. Welded Beam design with global optima 2.38116.

Sandgren's and Welded Beam design (WB)		Average of best fitness	T-test value
#3	GADO-R	-3.0665	0.0019
	ASAGA	-3.0666	
#13	GADO-R	26.83	0.0005
	ASAGA	26.79	
#21	GADO-R	135.42	0.0177
	ASAGA	124.11	
#22	GADO-R	188.64	0.0267
	ASAGA	182.74	
WB	GADO-R	2.4396	0.0019
	ASAGA	2.4139	

Table 1. Statistical analysis of GADO-R and ASAGA

5. FINAL REMARKS

Using fitness approximation methods to assist a GA or Evolutionary Algorithm (EA) have gained increasing popularity in recent years. An interesting question in this area is: what is the best model for fitness approximation? Though the answer depends on the problem and user requirements, we found an interesting generic solution which is to try the simplest model first. If the performance is not satisfactory or degrades with time, more sophisticated models are used.

So far many researchers use only one type of approximation model [25]. Some researches use multiple models for different levels of approximation but the approximate model itself is still fixed [13, 14]. ASAGA introduces an interesting research direction because it uses an adaptive modeling method. This adaptive method can provide the best trade-off between model performance and efficiency by adaptively adjusting the approximation. Experiments on benchmark domains and engineering design domains show that ASAGA outperforms good non-adaptive surrogate-assisted GAs.

There are still several open questions regarding this method that could be addressed in future work. One question would be if there is a better way to control model updates. The other question would be whether model updates should happen in the reverse direction as well, i.e. the more sophisticate model may be replaced by a simpler model at some point, perhaps when many more clusters are introduced.

6. REFERENCES

[1] K. Rasheed. GADO: A genetic algorithm for continuous design optimization. Technical Report DCS-TR-352, Department of Computer Science, Rutgers University, 1998. Ph.D. Thesis.

[2] Khaled Rasheed, Haym Hirsh: Learning to be selective in genetic-algorithm-based design optimization. *Artificial Intelligence in Engineering Design Analysis and Manufacturing* 13(3): 157-169, 1999.

[3] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments: A survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303-317, 2005.

[4] Y. S. Ong, P. B. Nair, A. J. Keane, and K. W. Wong. Surrogate-Assisted Evolutionary Optimization Frameworks for High-Fidelity Engineering Design Problems. In Y. Jin, editor, *Knowledge Incorporation in Evolutionary Computation*, Studies in Fuzziness and Soft Computing, pages 307--332. Springer, 2004.

[5] Eric Sandgren. The utility of nonlinear programming algorithms. Technical report, Purdue University. Ph.D. Thesis, 1977.

[6] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing Journal*, 9(1):3--12, 2005.

[7] R. Jin, W. Chen, and T.W. Simpson. Comparative studies of metamodeling techniques under miltiple modeling criteria. Technical report 2000-4801, AIAA, 2000.

[8] K. Sastry, D.E. Goldberg, and M. Pelikan. Don't evaluate, inherit. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 551-558, 2001. Morgan Kaufmann.

[9] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines.* Cambridge Press, 2000.

[10] LT Bui, HA Abbass, D Essam. Fitness inheritance for noisy evolutionary multi-objective optimization. *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages: 779-785. 2005.

[11] H.-S. Kim and S.-B. Cho. An efficient genetic algorithm with less fitness evaluation by clustering. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 887-894, 2001. IEEE.

[12] K Deb. An Efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 2000, Elsevier.

[13] Y. Jin and B. Sendhoff. Reducing fitness evaluations using clustering techniques and neural networks ensembles. In *Genetic and Evolutionary Computation Conference*, volume 3102 of *LNCS*, pages 688--699, 2004. Springer.

[14] Z. Z. Zhou, Y. S. Ong, P. B. Nair, A. J. Keane and K. Y. Lum. Combining Global and Local Surrogate Models to Accelerate Evolutionary Optimization. *IEEE Transactions on Systems, Man and Cybernetics - Part C*, Vol. 37, No. 1, pp. 66-76. 2007.

[15] T. Joachims. 11 in: *Making large-Scale SVM Learning Practical*. Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT Press, 1999.

[16] K. Rasheed and H. Hirsh. Informed operators: Speeding up genetic-algorithm-based design optimization using reduced models. *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2000)*, pp. 628--635, 2000.

[17] X Llorà, K Sastry, DE Goldberg, A Gupta, L Lakshmi. Combating User Fatigue in iGAs: Partial Ordering, Support Vector Machines, and Synthetic Fitness. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages: 1363-1370. 2005.

[18] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: the Art of Scientific Computing.* Cambridge University Press, Cambridge [England]; New York, 2nd edition, 1992.

[19] W. Carpenter and J.-F. Barthelemy. A comparison of polynomial approximation and artificial neural nets as response surface. Technical report 92-2247, AIAA, 1992.

[20] M. Hüscken, Y. Jin, and B. Sendhoff. Structure optimization of neural networks for aerodynamic optimization. *Soft Computing Journal*, 9(1):21--28, 2005.

[21] Khaled Rasheed, Xiao Ni, Swaroop Vattam. "Comparison of Methods for Developing Dynamic Reduced Models for Design Optimization". *Soft Computing Journal*, 9(1):29--37, 2005.

[22] T. Simpson, T. Mauery, J. Korte, and F. Mistree. Comparison of response surface and Kriging models for multidiscilinary design optimization. Technical report 98-4755, AIAA, 1998.

[23] Z. Zhou, Y.S. Ong, and P.B. Nair. Hierarchical surrogate-assisted evolutionary optimization framework. In *Congress on Evolutionary Computation*, pages 1586--1593, 2004. IEEE.

[24] Y. Jin, M. Hüsken, M. Olhofer, and B. Sendhoff. Neural networks for fitness approximation in evolutionary optimization. In Y. Jin, editor,*Knowledge Incorporation in Evolutionary Computation*, pages 281--305. Springer, Berlin, 2004.

[25] D. Bueche, N.N. Schraudolph, and P. Koumoutsakos. Accelerating evolutionary algorithms with Gaussian process fitness function models. In *IEEE Trans. on Systems, Man, and Cybernetics: Part C*, 35(2):183-194, 2005.

[26] C.M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[27] L. Willmes, T. Baeck, Y. Jin, and B. Sendhoff. Comparing neural networks and kriging for fitness approximation in evolutionary optimization. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 663--670, 2003.