ADAPTIVE SURROGATE-ASSISTED EVOLUTION

by

LIANG SHI

(Under the Direction of Khaled Rasheed)

ABSTRACT

Evolutionary Algorithms (EAs) used in complex optimization domains usually need to perform a large number of fitness function evaluations in order to find near-optimal solutions. In real world application domains such as engineering design problems, such evaluations can be extremely computationally expensive. It is therefore common to estimate or approximate the fitness. A popular method is to construct a so-called surrogate or meta-model, which can simulate the behavior of the original fitness function, but can be evaluated much faster. An up-to-date survey of fitness approximation applied to EA is presented in this dissertation. The main focuses are the methods of fitness approximation, the working styles of the fitness approximation, and management of the fitness approximation during the optimization process. Working with fitness approximations creates a problem in that it is difficult to determine which approximate model is appropriate for each domain and how often the model should be used. Even for one domain the answer is not usually fixed, but varies within the different stages of the optimization process. To solve this problem, an adaptive fitness approximation GA (ASAGA) is presented. ASAGA adaptively chooses the appropriate model type by adjusting the model complexity and the frequency of model usage according to time elapsed and model accuracy. ASAGA also introduces a stochastic penalty function method to handle constraints. Experiments show that ASAGA performs better than or comparable to several state-of-art surrogate-assisted EAs in benchmark domains and engineering design domains. Statistical analysis suggests that ASAGA outperforms a fixed surrogate-assisted GA in constrained function domains with statistical significance.

INDEX WORDS: Evolutionary algorithms, Surrogate-assisted evolution, Adaptive Meta-modeling, Fitness approximation

ADAPTIVE SURROGATE- ASSISTED EVOLUTION

by

LIANG SHI

B.S., Nanjing University, P.R.China, 1994

M.S., University of Georgia, 2006

A Dissertation Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

ATHENS, GEORGIA

2008

© 2008

Liang Shi

All Rights Reserved

ADAPTIVE SURROGATE-ASSISTED EVOLUTION

by

LIANG SHI

Major Professor: Khaled Rasheed

Committee: Walter D. Potter
Suchendra M. Bhandarkar
Jeongyoun Ahn

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
December 2008

DEDICATION

This dissertation is dedicated to my wife and my son. Without them, it would not have been possible for me to finish this dissertation. This dissertation is also dedicated to my parents, for their continuous encouragement and support.

ACKNOWLEDGEMENTS

The author would like to thank Dr. Khaled Rasheed for his continuous guidance and advice through my academic career at UGA. The author also would like to thank Dr. Walter D. Potter, Dr. Suchendra M. Bhandarkar, Dr. Jeongyoun Ahn and Dr. Mary Meyer. They gave the author many valuable comments and ideas for writing this dissertation. In addition, the author would like to thank Dr. Hamid Arabnia and Dr. John A. Miller, for their valuable suggestions and advice for the author's course work and research. Finally, the author would like to thank the authors who were kind enough to provide their data for the comparison of methods. In particular the author would like to thank Dr. J. Branke, Dr. Y.S. Ong, Dr. R. Regis and Dr. Z. Zhou for their professional courtesy.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

Chapter 1

Introduction

Background:

Evolutionary Algorithms (EAs) have repeatedly proven to be a powerful method for solving optimization problems, and are consequently used in a wide range of real-world applications such as engineering design domains. In such domains, it is not uncommon to see fitness functions that are discontinuous, non-differential, highly multi-modal, noisy and ambiguous [1, 58]. It is therefore not surprising that EAs usually perform better than conventional optimizers such as sequential quadratic programming and Simulated Annealing [1, 2].

Problem Statement:

Many challenges still arise in the application of EAs 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 region. However, determining the fitness of each point may involve the use of a simulator or analysis code that takes an extremely long time to execute. Therefore it would be difficult to be cavalier about the number of objective evaluations used for an optimization [58]. Another challenge is that the environment of an Evolutionary Algorithm can be noisy, which means that the exact fitness cannot be determined, and an approximate fitness must be assigned to each individual. An average fitness solution to the noise problem requires even more evaluations [51].

For such problems surrogate-assisted evolution methods based on fitness approximation are preferable, as they can simulate the exact fitness at a much lower computational cost. A good fitness approximation method can still lead an EA process to find optimal or near-optimal solutions and is also tolerant to noise (refer to Schwefel et al. [3] and Jin et al. [6] for details). However, working with fitness approximations creates a problem in that 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, or in all stages of optimization.

Proposed Solution:

In this dissertation we present ASAGA (Adaptive Surrogate-Assisted Genetic Algorithm). ASAGA's backbone consists of GADO (Genetic Algorithm for Design Optimization) [1, 58], a GA that has proved to be powerful for solving engineering design problems. ASAGA makes use of global and local approximate models, using a clustering technique with one local model for each cluster. The global model can adaptively evolve from a quadratic polynomial model to a Support Vector Machine (SVM) model. The model evolution depends on time spent and model accuracy. The local models follow similar rules, except that they can evolve from a simple average of the fitness of all individuals in a cluster to a final quadratic model (not SVM), since each local model is based on a sample set which is not as large. The number of clusters is also adaptively controlled according to the type of model currently being used, model accuracy, and the size of the cluster.

In ASAGA, the frequency of model usage is also adaptively adjusted according to a formula that takes into account the percentage of runtime spent on fitness approximation, as well as model accuracy. Intuitively, if the approximate model formation and usage make up a significant amount 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 to reach a trade-off point is applied in ASAGA.

A stochastic penalty function method is introduced in ASAGA to further strengthen its performance when dealing with constrained problem domains.

Dissertation Structure:

The remainder of this dissertation is organized as follows.

Chapter 2: Several kinds of fitness approximation methods and their incorporation mechanisms are discussed. Multiple surrogate usage in current EA research is introduced. Comparative studies of different surrogate-assisted EA methods are given.

Chapter 3: The proposed approach (ASAGA) is described in detail.

Chapter 4: Experimental results are presented, including comparisons to state-of-art surrogate-assisted EA methods in unconstrained benchmark function domains, constrained benchmark function domains, and a real engineering design problem.

Chapter 5: Parameter tuning for ASAGA via performance analysis is discussed

Chapter 6: The dissertation is concluded in this chapter with a summary and a discussion of open questions and future work.

Chapter 2

Fitness Approximation Methods and Comparative Studies

The optimization problem usually deals with non-linear functions. There are some classic approximation methods, such as transforming the original function to a simpler one. This method entails transforming the original functions to linear ones, and then using a linear programming technique, such as the Frank-Wolfe method [61] or Powell's quadratic approximation [62]. Other classical methods like the Fourier approximation and Walsh function approximation use a set of basis functions and find a weighted sum of such basis functions to use as an approximation. These techniques have been used to first transform the original problem into an easy one, and then apply EA to find the optima of the easier version of the original fitness function [63, 64].

Another type of method determines a function's approximation using a chosen set of evaluated points extracted from the whole design space or from the evaluation history. This class includes instance learning methods, machine learning methods and statistical learning methods. The relationships between the different types of fitness approximation methods are shown in Fig. 2.1.

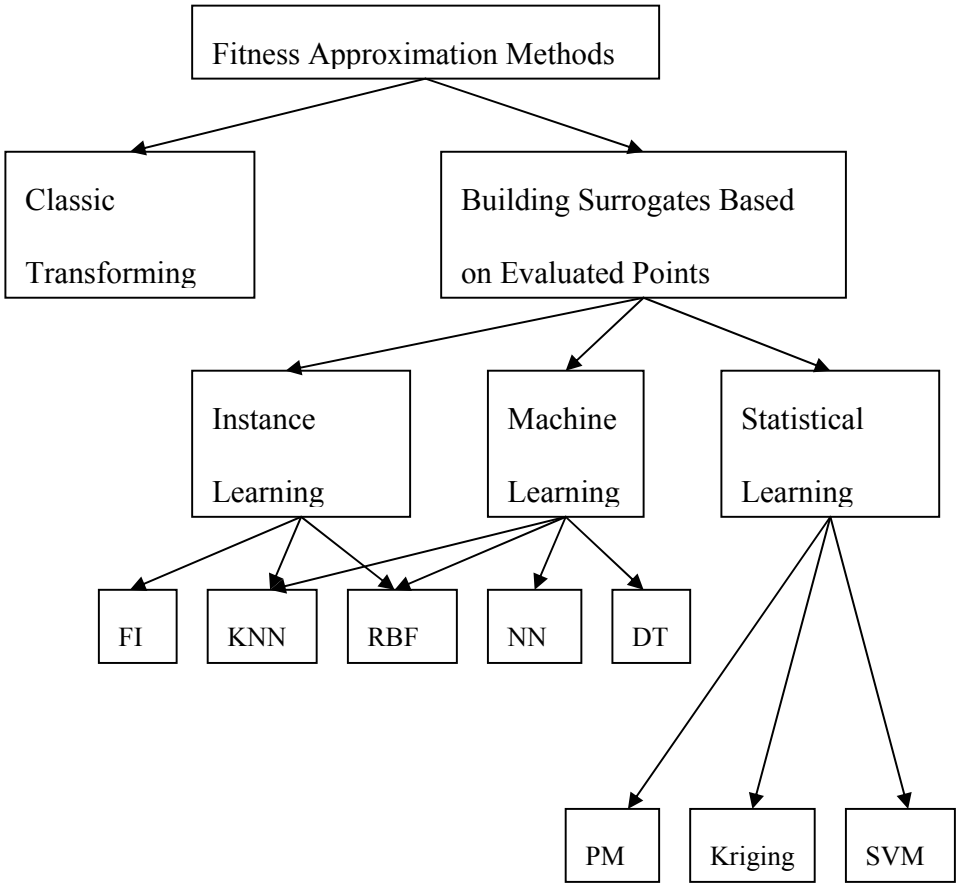

Fig. 2.1 Fitness approximation methods, FI: Fitness Inheritance; KNN: K-Nearest Neighbors; RBF: Radial Basis Functions; NN: Neural Networks; DT: Decision Tree; PM: Polynomial Model; SVM: Support Vector Machines

Instance-based Learning and Machine Learning Methods:

Many Instance-based Learning and other Machine Learning methods have been used for fitness approximation in GAs. In this section, several of the most popular of these methods are reviewed.

1. Fitness Inheritance

Fitness inheritance techniques are one of the main subclasses of fitness approximation techniques. One such technique consists of setting the fitness of a new solution (child) based on the average fitness of its parents or a weighted average based on how similar the child is to each parent [7]. To deal with a noisy fitness function, a re-sampling method combined with a simple average fitness inheritance method is used to reduce the computational cost [10]. Another approach is to divide the population into building blocks according to certain schemata. Under this approach, an individual obtains its fitness from the average fitness of all the members in its building block [8]. More sophisticated methods such as conditional probability tables and decision trees are used in [9] for fitness inheritance.

2. Radial Basis Functions Model

The Radial Basis Functions (RBF) model is another instance learning method. RBF networks are actually a type of neural network. Since it is a very popular technique for fitness approximation in EA [2, 14, 26], it is worthy of being introduced independently from normal multilayer perceptron neural networks. 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. Fig. 2.2 shows the structure of an RBF network. The size of the hidden layer (k) can be equal to the problem dimension if the sample size is small. In the case of a larger sample size, k is usually smaller than the problem dimension to avoid expensive implementation. This type of RBF network is called a generalized RBF network. The output y(x) of the RBF network is given as a linear combination of a set of radial basis functions expressed in the following way:

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

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. It evaluates the distance between the input x and the center c i. For the case of the generalized RBF network, the center c i is also unknown and has to be learned by other methods such as the K-means method. Typical choices for the kernel include linear splines, cubic splines, multi-quadrics, thin-plate splines, and Gaussian kernels. 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 [27].

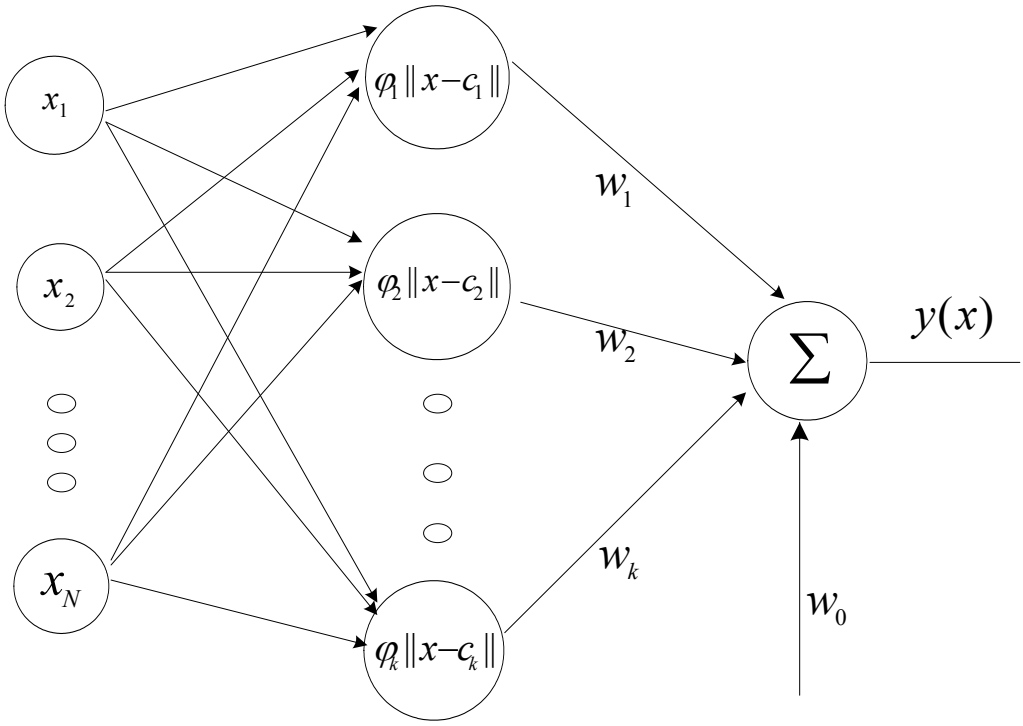

Fig. 2.2 Structure of RBF models

3. Clustering Techniques

Clustering algorithms include those based on hierarchy clustering (such as single-linkage, complete-linkage, average-linkage and Ward's method), partition clustering (such as Hard C-Means and K-Means algorithm), and overlapping clustering (such as Fuzzy C-Means and B-Clump algorithm). Among them, the K-Means algorithm appears to be the most popular one for application to GAs due to its relative simplicity and low computational cost. In [11, 12], the entire population is divided into many clusters, and only the center of each cluster is evaluated. Other individuals' fitness values in the clusters are computed using their distance from these centers. Another approach is to build an approximate model based on sample points composed of the clusters' centers. Every other individual's fitness is estimated by this approximate model,

which may be a neural network model [13] or an RBF model [15]. 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 utilize more local information about the search space and fit the original fitness function better than a single model [4, 15, 16]. ASAGA, the method proposed in this dissertation, also uses this method.

4. Multilayer Perceptron Neural Network

Multilayer Perceptron Neural Networks (MLPNNs) usually utilize the back-propagation algorithm. MLPNNs have been proven to be powerful tools for fitness approximation. An MLPNN model is generally used to accelerate the convergence by replacing the original fitness function [29, 31]. In engineering design domains and drug design, MLPNNs have been used to reduce the evaluation times of complex fitness functions [30, 32, 33].

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. The structure of a feed-forward MLPNN is shown in Fig. 2.3. W and θ are the unknown weights to be learned. The most commonly used activation function is the logistic function, which has the form:

$$f(x) = \frac{1}{1 + e^{-cx}} \tag{4}$$

where c is a constant. The function is plotted in Fig. 2.4. Readers can refer to Bishop et al. [27] for a comprehensive study.

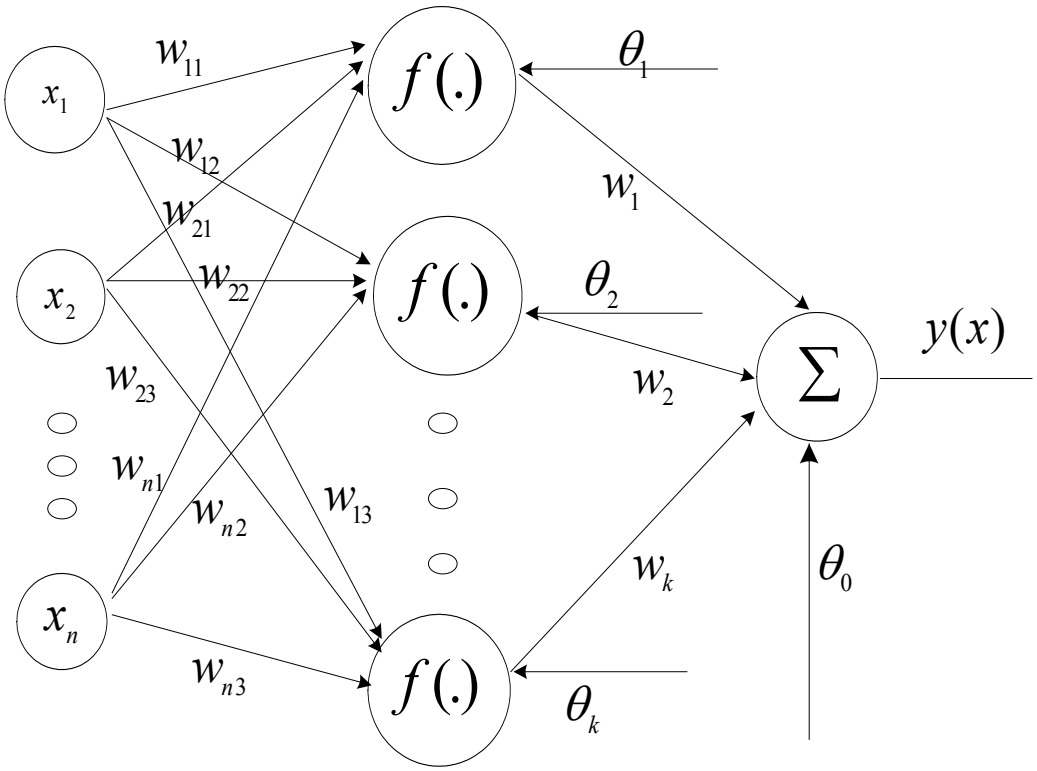

Fig. 2.3 Structure of the feed-forward MLPNN model

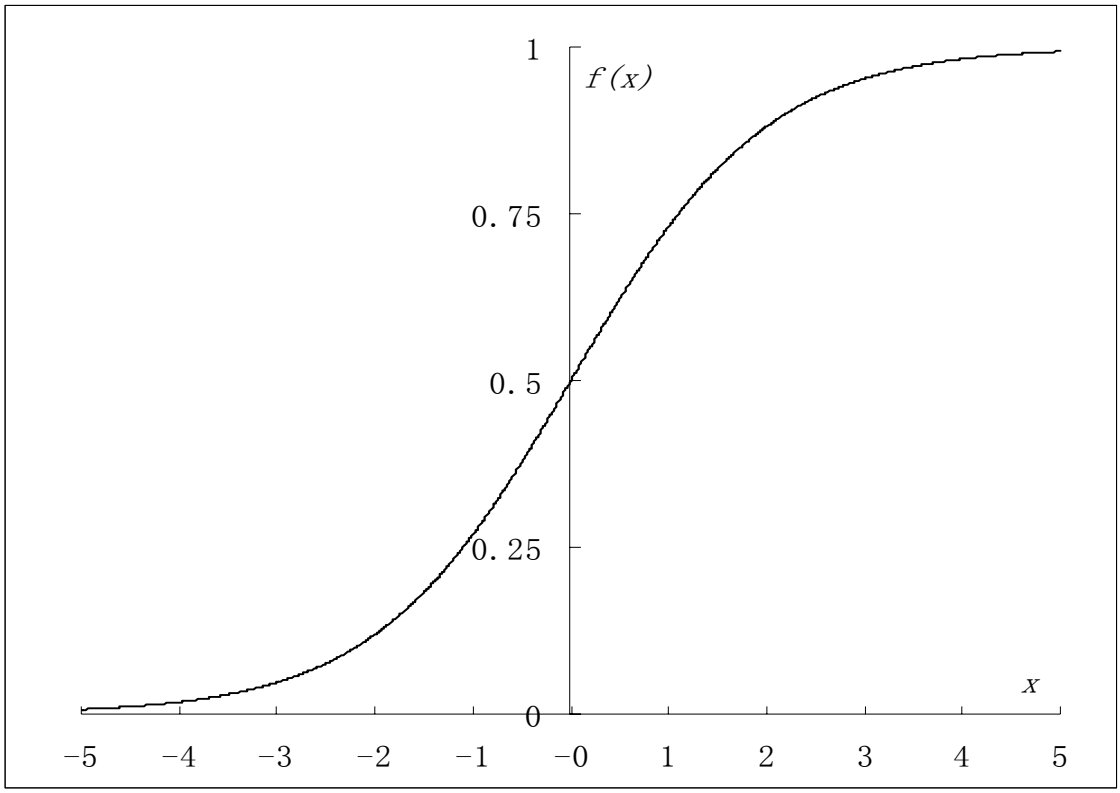

Fig. 2.4 Logistic function with C = 1

Other machine learning techniques are also applied for fitness approximation in EA. An individual's fitness can be estimated by its neighbors using K-nearest-neighbors algorithm [17]. The screening model technique has been used for pre-selection [1, 58]. Decision Tree (DT) is another machine learning technique which has been used in [9].

Statistical Learning Methods:

Statistical learning methods for fitness approximation (basically statistical learning models) as applied to EA have gained much interest among researchers, and have been used in several

successful GA packages. In these methods, single or multiple models are built during the optimization process to approximate the original fitness function. These models are also referred to as approximate models, surrogates or meta-models. Among these models, Polynomial Models, Kriging Models, and Support Vector Machines (SVM) Models are the most commonly used.

1. Polynomial Models

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

$$\hat{F}\left(\overline{X}\right) = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{i=1,\, j=1}^{n,n} a_{ij} x_i x_j \tag{5}$$

where a0, ai and aij are the coefficients to be fitted, n is the dimension of the problem, and xi is the ith design variable.

Usually the least-squares approximation method is used to fit the unknown coefficients a_0, a_i and a_{ij}. 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 arise if two sample points are too close to each other. Another drawback of the least-squares method is that its computational complexity grows quickly with the problem's dimension. This growth can be unacceptable when the problem's dimension is high. The gradient method is introduced to address the problems of the least-squares method. More implementation details for using these two methods to fit a polynomial model can be found in [18].

2. Kriging Model

The Kriging model consists of two component models which can be mathematically expressed as:

$$y(x) = f(x) + Z(x) \tag{6}$$

where f(x) represents a global model and Z(x) is the realization of a stationary Gaussian random function that creates a localized deviation from the global model. f(x) is a polynomial and can be considered to be an underlying constant β in many cases, so that equation (6) becomes:

$$y(x) = \beta + Z(x) \tag{7}$$

The estimated model in equation (3) is given as:

$$\hat{y} = \hat{\beta} + r^T(x) R^{-1} (y - f\hat{\beta}) \tag{8}$$

where y is a vector of length N as defined in equation (7), $\hat{y}$ is the estimated value of y given the current input x, 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 between any two sampled data points. The form of the correlation function is specified by the user. Gaussian exponential correlation functions are commonly used, which is why the Kriging model is also called a Gaussian process:

$$R(x^i, x^j) = \exp[-\sum_{k=1}^{n} \theta_k | x_k^i - x_k^j |^2] \tag{9}$$

The correlation vector between x and the sampled data points is expressed as:

$$r^T(x) = [R(x, x^1), R(x, x^2), \ldots, R(x, x^n)]^T \quad (10)$$

Estimation of the parameters is often carried out using the generalized least squares method or the maximum likelihood method. Detailed implementations can be found in [19, 20].

In addition to the approximate values, the Kriging method can also give accuracy information about the fitting in the form of confidence intervals of the estimated values without extra computational cost. In [5, 23], a Kriging model is used to build the global models because it is believed to be a good solution for fitting complex surfaces. A Kriging model is used to pre-select the most promising solutions in [24]. In [21, 22, 25], a Kriging model is used to accelerate the optimization or reduce the expensive computational cost of the original fitness function. One disadvantage of this method is that it is sensitive to the problem's dimension. The computational cost is unacceptable when the dimension of the problem is high.

3. Support Vector Machines (SVM)

The SVM model is primarily a classifier method that performs classification tasks by constructing hyper-planes in a multidimensional space to separate cases of different class labels. SVM models support both regression and classification tasks and can handle multiple continuous and categorical variables. A detailed description of SVM can be found in [35, 36]. Beneficial features of SVM compared to other approximate models are that it is not sensitive to local optima, the optimization process does not depend on the problem dimensions, and overfitting is seldom an issue. Applications of SVM for fitness approximation can be found in [37]. The regression SVM is used for constructing approximate models. There are two types of regression

SVMs: epsilon-SVM regression and nu-SVM regression. The epsilon-SVM regression model is more commonly used for fitness approximation, where the linear epsilon-insensitive loss function is defined by:

$$L^{\varepsilon}(x, y, f) = |y - f(x)| = \max(0, |y - f(x)| - \varepsilon) \tag{11}$$

The sum of the linear epsilon-insensitive losses must be minimized:

$$\frac{1}{2} w^T w + C \sum_{i=1}^{N} L^{\varepsilon}(x_i, y_i, f) \tag{12}$$

which is equivalent to a constrained minimization problem having the form:

$$\frac{1}{2} w^T w + C \sum_{i=1}^{N} \zeta_i + C \sum_{i=1}^{N} \zeta_i^*, \tag{13}$$

subject to the following constraints:

$$\begin{aligned} & w^T \phi(x_i) + b - y_i \le \varepsilon + \zeta_i^* \\ & y_i - w^T \phi(x_i) - b_i \le \varepsilon + \zeta_i \\ & \zeta_i, \zeta_i^* \ge 0, i = 1, \ldots, N \end{aligned} \tag{14}$$

where $\phi(x)$ is called the kernel function. It may have the forms of linear, polynomial, Gaussian, RBF and sigmoid functions. The RBF is by far the most popular choice of kernel type used in SVMs. This is mainly because of their localized and finite responses across the entire range of the real x-axis. This optimization problem can be solved by using quadratic programming techniques.

Existing research in Multi-surrogate Assisted EA:

For some real world applications, special approximation methods have been used. For example, a one dimensional approximation of the Kubelka Munk model is used to replace the expensive Monte Carlo method in an EA for analyzing colon tissue structure [50]. In [52], a classifier with confidence information is evolved to replace time consuming evaluations during tournament selection.

In some applications, several approximate methods have been combined to construct a type of fitness approximation model known as a Multi-surrogate. In [13, 28] the MLPNN model was combined with clustering methods for constructing approximate models (shown in Fig. 2.5). Fig. 2.6 shows another strategy using clustering techniques and polynomial models together [4]. A trained RBF model was used to generate sample points for the construction of polynomial models for fitness approximation in [34]. In [23, 46], the Kriging method was used to construct a global approximate model for pre-selection. Then RBF models were built using those pre-selected sample points for further fitness approximation. Fig. 2.7 shows the structure of this model. Multiple approximate models formed in a hierarchical structure have been used to assist the fitness evaluations in [47]. In [65, 66], multiple local approximate models are built for each individual, then these local models are aggregated by an average or weighted average of all approximate models. In [67, 68], multiple surrogates are built, then the best surrogate is used [63] or the weighted sum of all surrogates is used, where the weights associated with each surrogate are determined based on the errors of the surrogate [64]. Fig. 2.8 shows this model in detail.

Multi-surrogates are also used for multi-objective optimization problems. In [69], a NSGA-II algorithm multi-objective EA using PM and RBF surrogates together is presented. A local

surrogate-assisted evolution strategy using KNN and RBF models is introduced in [59]. For each new offspring in this strategy, a cubic RBF surrogate is built using the *k*-nearest previously evaluated points. This local RBF surrogate is then used to estimate the new offspring's fitness. Fig. 2.9 shows the model structure used in [59]. In [70], Polynomial Models are used to estimate the coefficients of the fitness surrogate. Thus the surrogate is made adaptive to characteristics of the specific optimization problems.

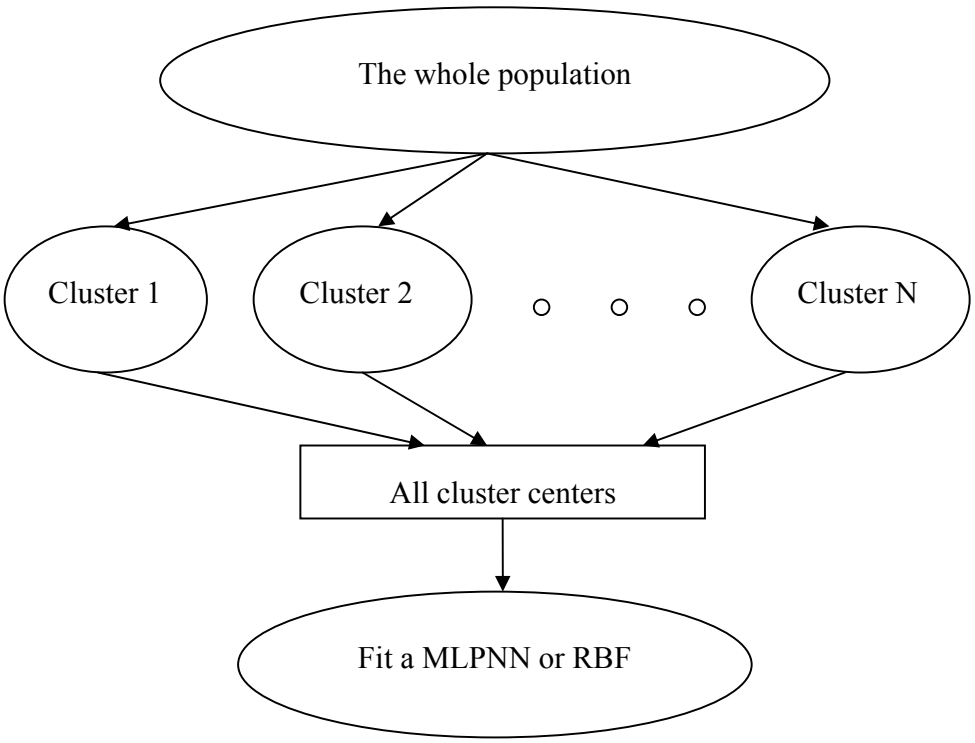

Fig. 2.5 A multiple surrogate structure used in [13, 28]

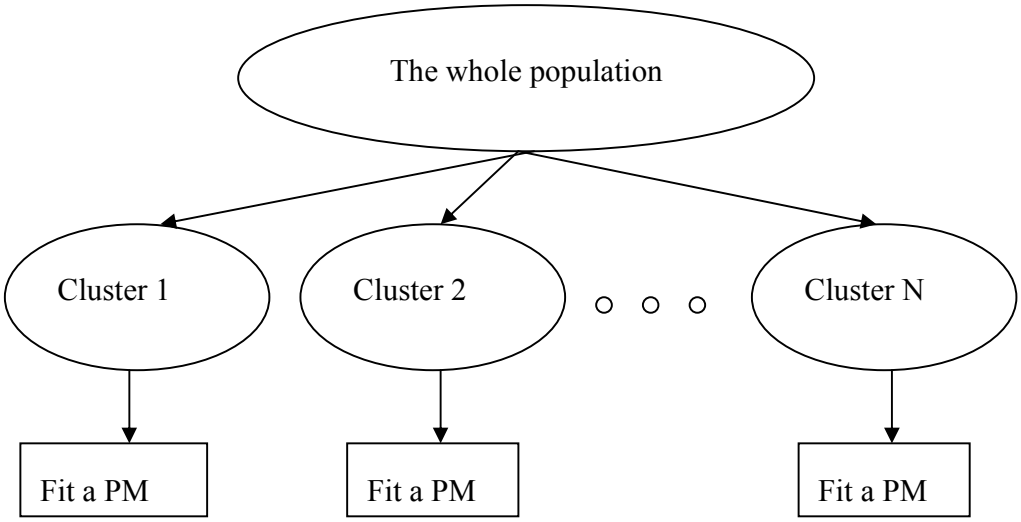

Fig. 2.6 A multiple surrogate structure used in [4]

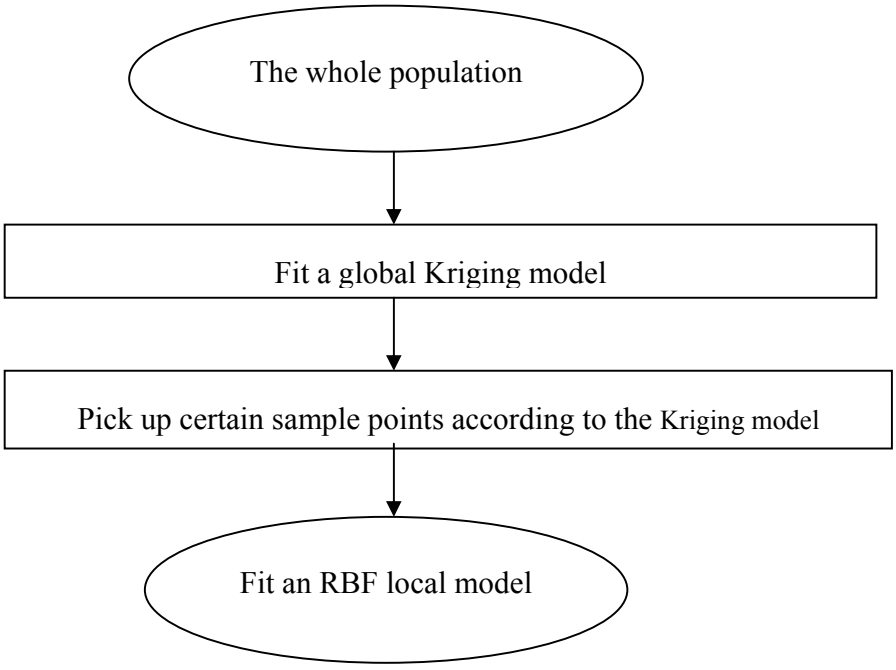

Fig. 2.7 A multiple surrogate structure used in [23, 46]

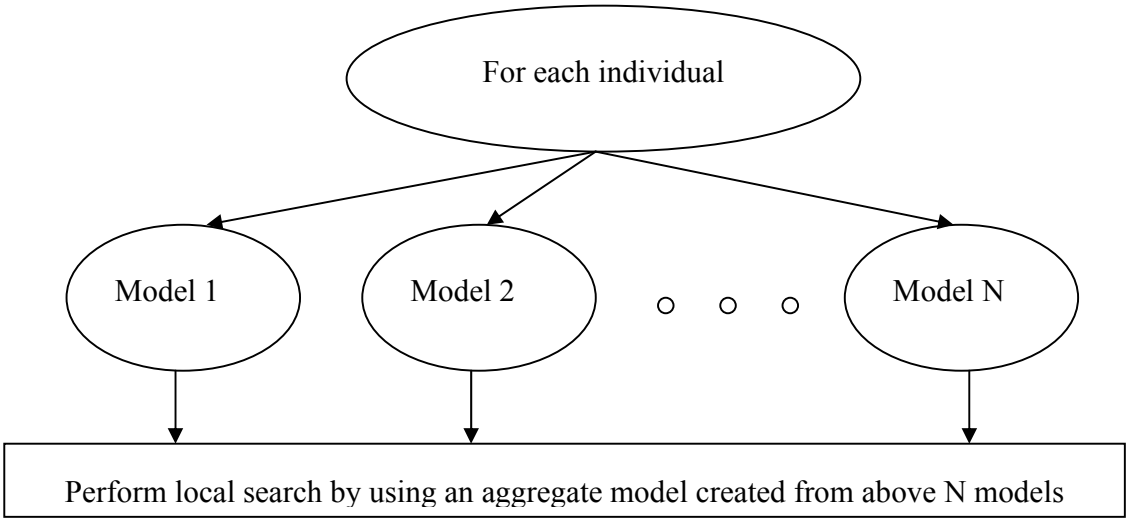

Fig. 2.8 A multiple surrogate structure used in [65]

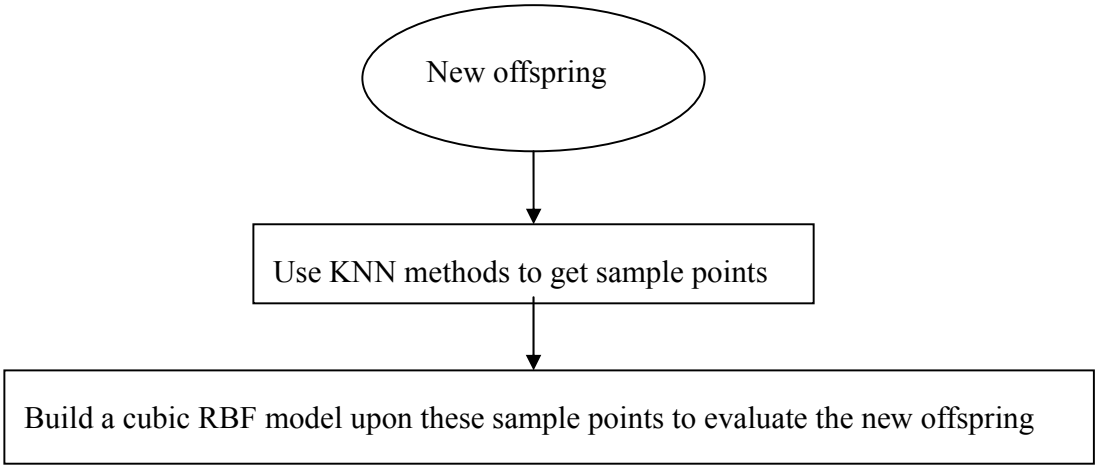

Fig. 2.9 A multiple surrogate structure used in [59]

Surrogate Incorporation Mechanisms:

There are two categories of surrogate incorporation mechanisms in EAs, as shown in Fig. 2.10. In one category the original fitness is directly replaced by the estimated fitness when the individual is evaluated throughout the optimization. Only a few individuals have their exact fitness calculated for control purposes. In the other category, the original fitness is kept for each individual and the approximate fitness is not used to directly replace the original fitness. These two methods are reviewed next.

1. Direct Fitness Replacement

Direct fitness replacement is straightforward. Individuals are evaluated by surrogates and then the estimated fitness is assigned to each individual. During the course of the EA process, the approximate fitness assumes the role of the original fitness. This method has been used in numerous research efforts [5, 11, 12, 13, 14, 21, 22, 23, 25, 26, 28, 29, 30, 31, 32, 34, 37, 41, 42, 43, 46, 52]. The obvious drawback is that the inaccuracy of the approximate fitness may lead the EA to inferior local optima. Consequently, the direct fitness replacement method needs a continuous calibration process called Evolution Control. Even with Evolution Control, convergence to true optima cannot be guaranteed.

For direct fitness replacement methods the management of fitness approximation is necessary to drive the EA to converge to global optima with the cost reduced as much as possible. There are several ways to conduct the model management, as shown in Fig. 2.11.

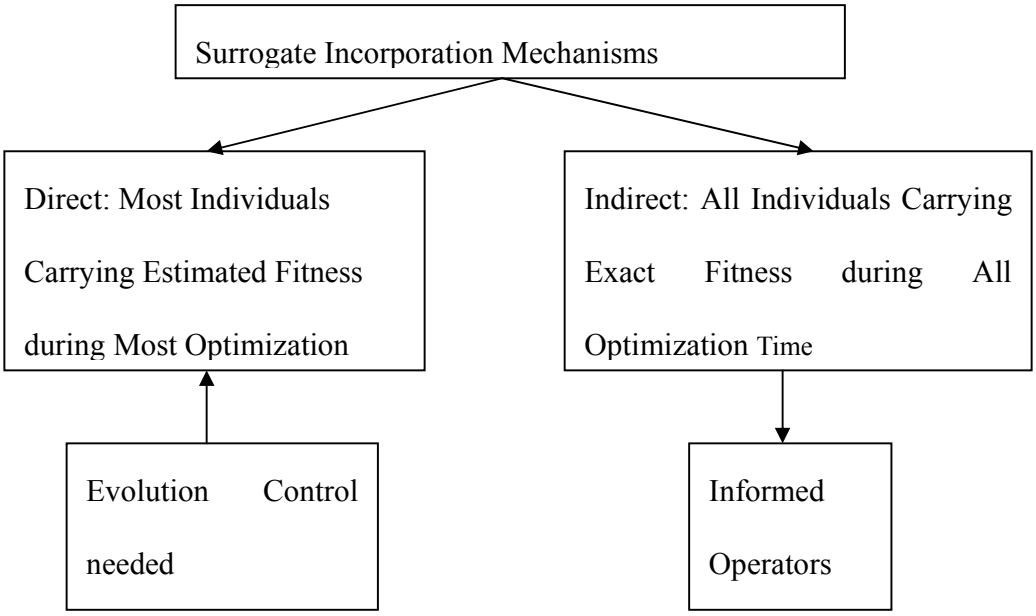

Fig. 2.10 Surrogate Incorporation Mechanisms

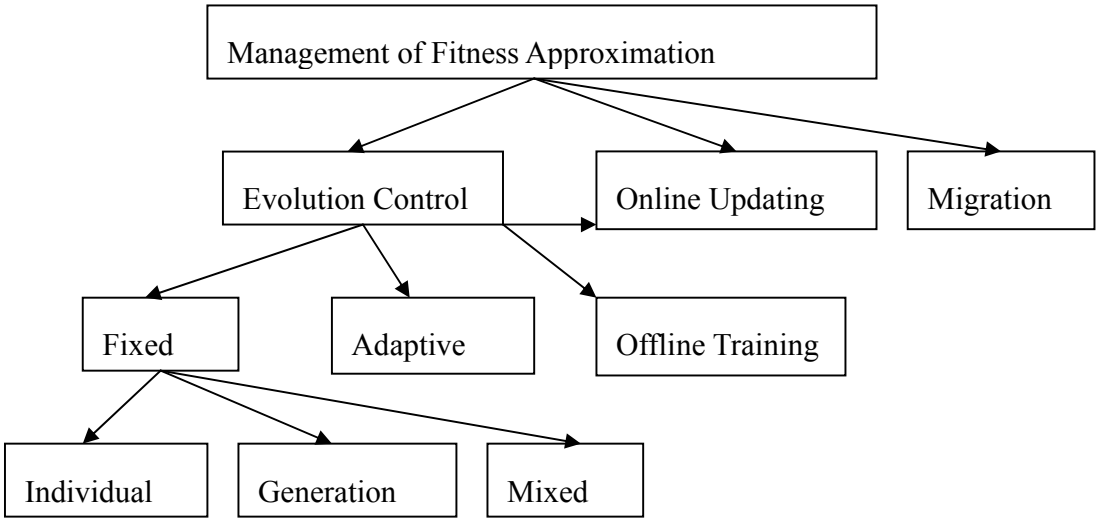

Fig. 2.11 Management of fitness approximation

Evolution Control uses surrogates together with original fitness functions in an EA process where the original fitness functions are used to evaluate some/all individuals in some/all generations. There are two categories of Evolution Control methods: fixed Evolution Control (including generation-based control and individual-based control) and adaptive Evolution Control. A detailed description of Evolution Control can be found in [6]. There are two ways to conduct the Evolution Control.

- Offline model training

Offline model training constructs surrogates based on human evaluation or previous optimization history data. In this case, either the approximate model is of high fidelity or the original fitness cannot be easily evaluated during an EA process, so the original fitness is never used. An example of this method can be found in [55].

- Online model updating

Fitness approximation may be constructed at an early stage of the EA process. Because of the limited sample points, a surrogate may concentrate on the region spanned by the existing sample points and not cover the rest of search space well. As the EA continues and new individuals enter into the population, the accuracy of the previously built surrogate model will decrease. Thus the surrogate model needs to be reformed using the old sample points together with the new sample points. This technique is known as online surrogate updating. There has been considerable research with this method [2, 4, 5, 12, 15, 22, 25, 26, 29, 31, 38, 43, 44, 46].

2. Indirect Surrogate Method

The indirect surrogate method computes the exact fitness for each individual during an EA process and the approximate fitness is used in other ways. For example, approximate fitness can be used for population pre-selection. In this method, instead of generating a random initial population, an individual for the initial population can be generated by selecting the best individual from a number of uniformly distributed random individuals in the design space according to the approximate fitness [4, 38, 44].

Approximate fitness can also be used for crossover or mutation in a similar manner, through a technique known as Informed Operators [4, 12, 38, 44]. Under this approach, the approximate models are used to evaluate candidates only during the crossover and/or mutation process. After the crossover and/or mutation process, the exact fitness is still computed for the newly created candidate solutions. Using the approximate fitness indirectly in the form of Informed Operators – rather than direct evaluation - is expected to keep the optimization moving toward the true global optima and to reduce the risk of convergence to suboptimal solutions because each individual in the population is still assigned its exact fitness [44]. On the other side, the direct fitness replacement methods need a correction process often called evolution control, and even then convergence to the true global optima still cannot be guaranteed. Experimental results have shown that a surrogate-assisted informed operator-based multi objective GA can outperform state-of-art multi objective GAs for several benchmark problems [4]. Informed Operators also make it easy to use surrogates adaptively, as the number of candidates can be adaptively determined.

3. Hierarchical Surrogates and Model Migration

The hierarchical surrogates method builds multiple models with a hierarchical structure during the course of an EA process [46, 56]. In [46], a Gaussian process model is built for the so-called global model. A user-specified percentage of the best individuals according to the global model are selected to form a local search space. Then Lamarckian evolution is performed involving a trust region-enabled gradient-based search strategy that employs RBF local approximate models to accelerate convergence in the local search space. In [56] the whole population is divided into several sub-populations. Each sub-population constructs its own surrogate with different accuracy. At a certain interval, the individuals in the different sub-populations can migrate into other sub-populations. This is called an island model. To gain a balance between the model performance and the population diversity, the selection of migration size and migration interval is important. It has been found that the migration interval plays a more dominant role than migration size [57].

Existing research into surrogate incorporation mechanisms and fitness management has been surveyed. So far none of these methods have used an adaptive strategy to incorporate surrogates into an EA process, and very few have used an adaptive technique for model construction or management.

Comparative Study for Fitness Approximate Methods:

Many approximation methods have been introduced for special problem domains. Even though these methods are claimed to save many function evaluations and to be nearly as good as the original fitness function, they are bound to their special domains, and thus no comparative

studies have been conducted on them. On the other hand, the performance of many general-purpose approximation methods has been compared in early papers, especially for popular methods such as statistical learning methods.

The neural network model and the polynomial model were compared in [41]. The study concluded that the performance of the two types of approximation was comparable in terms of the number of function evaluations required to build the approximations and the number of undetermined parameters associated with the approximations. However, the polynomial model had a much lower construction cost. In [38], a quadratic polynomial model was found to be the best method among the polynomial model, RBF network, and the Quickprop neural network when the models were built for regions created by clustering techniques. The authors were in favor of the polynomial model because they found that it 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 in [42]. However, no clear conclusion was drawn about which model is better. Instead, the author showed that optimization with a meta-model could lead to degraded performance. Another comparison was presented in [40] 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 Kriging approximations yielded comparable results with minimal difference in predictive capability. Comparisons between several approximate models were presented in [39],

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. Their 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 (details shown in Table 2.1).

	Low-order Nonlinearity	High-order Nonlinearity
Small Scale	Polynomial	RBF
Large Scale	Kriging	RBF
Overall	Polynomial	RBF

Table 2.1 Summary of best methods in [39]

In [65], four types of approximate models – Gaussian Process (Kriging), RBF, Polynomial model and Extreme Learning Machine Neural Network (ELMNN) – were compared on artificial unconstrained benchmark domains. Polynomial Models (PM) were found to be the best for final solution quality and RBF was found to be the best when considering correlation coefficients between the exact fitness and estimate fitness. Table 2.2 shows the performance ranks of these four models in terms of the quality of the final solution.

Benchmark Domains	Rank of Search Performance			
	Kriging	PM	RBF	ELMNN
Ackley	2	1	4	3
Griewank	3	1	2	4
Rosenbrock	1	3	2	4
Step	3	1	2	4

Table 2.2 Final quality measures for Kriging, PM. RBF and ELMNN models in [65]

So far different approximate models have been compared based on their performance, but the word performance itself has not been clearly defined. This is because the definition of performance may depend on the problem to be addressed, and multiple criteria need to be considered. Model accuracy is probably the most important criterion, since approximate models with a low accuracy may lead the optimization process to local optima. 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, overfitting is a common problem. In the case of overfitting, the model works very well on training data, yielding good model accuracy, but may perform poorly on new sample points. The optimization process could easily go in the wrong direction if it is assisted by a model suffering from overfitting. There are other important criteria to be considered, including robustness, efficiency, and time spent on model construction and updating. A fair comparison would consider the model accuracy as well as all of these 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 dealing with local regions or clusters and enough sample points are available [38]. In such cases, the fitting problem usually has low-order nonlinearity and the polynomial model is the best candidate according to [39]. The polynomial model is also believed to perform the best for problems with noise [39]. As for high-order nonlinear problems the RBF model is believed to be the best and it is the least sensitive to the sample size and has the most robustness [39]. So the RBF model is a good choice for a global model with or without plenty of samples.

The SVM model belongs to the class of kernel methods and is a very powerful model fitting tool. Because of the beneficial 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 dimensionality, many local optima, and a large amount of sample points.

Chapter 3

The Proposed Method - ASAGA

A General View of ASAGA:

ASAGA‘s backbone consists of GADO, 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, 58]. It has demonstrated a great deal of robustness and efficiency relative to competing methods.

In ASAGA, each individual in the GA population represents a parametric description of an artifact, such as an aircraft or 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 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. ASAGA has a surrogate assistance component which forms approximate models and uses them for speedup through informed genetic operators. These operators are described briefly here and

in more detail in [44]. The main idea is to make genetic operators such as mutation and crossover more informed by using approximate models. Whenever 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. The Informed Operators in ASAGA are used in the following operations:

- Informed initialization: Approximate fitness is used for population pre-selection. Instead of generating a random initial population, an individual for the initial population can be generated by selecting the best individuals from a number of uniformly distributed random individuals in the design space according to the approximate fitness.
- Informed mutation: To perform informed mutation, several random mutations of the base point are generated. Each random mutation is generated according to the regular method used in ASAGA by randomly choosing from among several different mutation operators and then randomly selecting the proper parameters for the mutation method. The mutation with the best approximate fitness value is returned as the result.
- Informed crossover: In crossover, two parents are selected at random according to the usual selection strategy. These two parents are not changed in the course of the informed crossover operation. Several crossovers are conducted by randomly selecting a crossover method, randomly selecting its internal parameters and applying it to the two parents to generate a potential child. The internal parameters depend on the crossover method selected; for example, to perform point crossover the cut-and-paste point has to be selected. The surrogate is used to evaluate every potential child, and the best child is selected as the outcome of the informed crossover.

- Informed guided-crossover: Guided crossover [49] is used in ASAGA to replace some of the regular crossover-mutation operations in order to improve convergence towards the end of the optimization. Guided crossover does not involve mutation, so we treat it differently. Informed guided crossover works as follows:
  - Several candidates are selected at random using the usual selection strategy of ASAGA to be the first parent (P1) for a guided crossover.
  - The second parent is selected from the population in a different way: for each point X in the population other than P1, a quantity Mutual_fitness(X, P1) is computed, where $Mutual\_fitness(A,B) = \frac{(fitness(A) - fitness(B))^2}{Euclidean\_dist(A,B)^2}$. A choice for X that maximizes Mutual_fitness(X, P1) is taken to be the second parent (49). Several random points are generated from the guided crossover of the two parents and ranked using the surrogate.
  - The best of the random points generated is taken to be the result of the guided crossover.

ASAGA adds many features to existing GADO and make it even more powerful. These features include:

- Adaptive surrogate selection
- Adaptive clustering technique
- Adaptive surrogate usage with informed operators
- Stochastic constraints handling

The rest of this chapter includes more details about these features.

Adaptive Surrogate Selection:

ASAGA utilizes both a global and local model structure. It fits one global approximate model for all the points evaluated so far during the course of the optimization and many local models for different regions (clusters). ASAGA first attempts to use the local model. However, if an individual belongs to a cluster for which no local model exists (e.g., when a new cluster is formed and does not have enough sample points to form a local model), this individual will be evaluated by the global model. One advantage of this structure is that local models can achieve a better fit because 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 takes into account more global information and may prevent the search from falling into 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 if runtime needs to be reduced, a large reservoir (the default size is 2000) of evaluated points can be used to construct the approximate models. When the reservoir is full, each newly evaluated point is added by replacing an old point selected randomly. ASAGA's global model and local models are formed based on the points in the reservoir. A beneficial feature of ASAGA is that all current individuals retain exact fitness values and approximate models are used for Informed Operators and initialization, never directly replacing the original fitness values. This prevents the search process from converging to false optima.

Based on the discussion in Chapter 2, 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 the beneficial features of SVM (described above), it is generally

considered to be a good choice for a global optimization model. In ASAGA, the SVM model is implemented using the SVM^{light} package. Implementation details about SVM^{light} can be found in [53, 54].

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 and the local model upgrade path are shown in Fig. 3.1 and Fig. 3.2 respectively.

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 occurs. The formula to determine the threshold is:

$$v = R \times (1 + T_1 / T_2) \tag{15}$$

where R is the correlation coefficient between the exact fitness and the approximate fitness, T_1 is the time spent at the surrogate phase which includes the time spent in forming surrogates and the time spent in using surrogates, and T_2 is the total running time. T_1 and T_2 are recorded during the program's run.

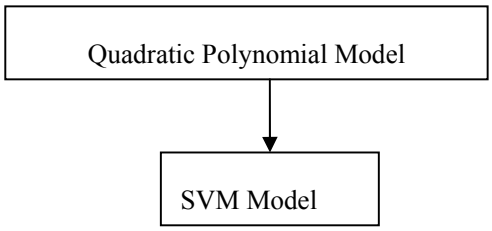

Fig. 3.1 Approximate model upgrading path for global models

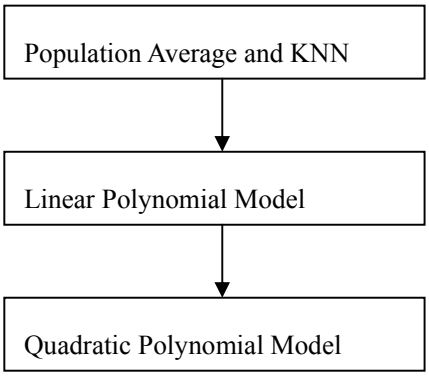

Fig. 3.2 Approximate model upgrading path for local models

The reason the correlation coefficient is used to measure the model accuracy is because of the way ASAGA utilizes 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; the correlation between them is what matters for informed initialization, crossover and mutation. The correlation coefficient equation is given by:

$$R = \frac{\sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \overline{x})^2 \sum_{i=1}^{n} (y_i - \overline{y})^2}} \tag{16}$$

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. The resulting value *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 (15), T_1 / T_2 is between 0 and 1 but never reaches 1. As a result, the formula (15) output is usually between 0.5 and 1.5 for fitness models and is between 0 and 1.5 for violation models. A good threshold can achieve a balance between optimization time and performance, using the least optimization time without hurting the final performance. In order to choose a good threshold we ran a set of experiments with different thresholds. For this purpose, we use a 5 dimension Ackley's function as the following:

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

$$-30 \le x_i \le 30 \qquad (17)$$

The average best fitness for 30 runs with different random starting populations is reported with the corresponding number of function evaluations in Table 3.1. From the table we can see that all threshold settings except for 0.2 have very little difference, and that the threshold 0.9 yields the best final value. It was further found that any threshold larger than 0.95 performs similarly to 0.95 and any threshold smaller than 0.65 performs worse than 0.65. Table 3.1 shows the statistics analysis which demonstrates that a small threshold 0.2 has worst performance, whereas other values have no statistically significant difference. Two T-tests are conducted to compare the performance with different thresholds. A T-test value less than 0.05 suggests a statistically significant difference between two objects. Table 3.2 shows an experiment in a 20 dimension Griewank's function domain with 6000 fitness function evaluations. Griewank's

function definition is as follows:

$$f_{Griewank}(x) = 1 + \sum_{i=1}^{n} x_i^2 / 4000 - \prod_{i=1}^{n} \cos(x_i - \sqrt{i})$$

$$-600 \leq x_i \leq 600 \quad (18)$$

It shows no significant difference between threshold values of 0.9 and 1.6, while the small threshold value of 0.2 performs much worse. Fig. 3.3 shows the best fitness found so far with the fitness function evaluations. Consequently, we believe that the threshold setting does not affect the final performance much as long as the threshold is within a reasonable range, whereas overly small threshold values will result in ASAGA retaining simple surrogate levels and damaging the optimization. Thus in ASAGA the threshold is 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 all experiments. Fig. 3.4 shows performance for three threshold values. The figure shows that a threshold of 0.5 yields the best result, while threshold settings 0.3 and 0.7 have little difference, showing that a threshold setting of 0.5 is within a reasonable range. Fig. 9 shows the performance of the three threshold settings. Table 3.3 shows the statistics analysis of this experiment. The test function used in this experiment is introduced in Chapter 4 and the violation function is explained later in this chapter.

Threshold	Mean ($\times 10^{-3}$)	SD ($\times 10^{-3}$)
0.95	5.223	3.162
0.9	4.938	3.878
0.85	6.051	4.153
0.8	5.874	3.289
0.75	6.124	3.689
0.7	5.548	2.901
0.65	5.858	3.665
0.2	13.161	10.631
T-test (0.9, 0.75)	0.1149	
T-test (0.9, 0.2)	0.0001	

Table 3.1 Statistical analysis of pre-threshold selection for objective function tested by a 5D Ackley function, where SD stands for Standard Deviation.

Threshold	Mean	SD
1.6	0.1667	0.0584
0.9	0.1579	0.0480
0.2	0.5253	0.2856
T-test (0.9, 0.2)	0.0000	
T-test (0.9, 1.6)	0.2634	

Table 3.2 Statistical analysis of pre-threshold selection for objective function tested by a 20D Griewank function

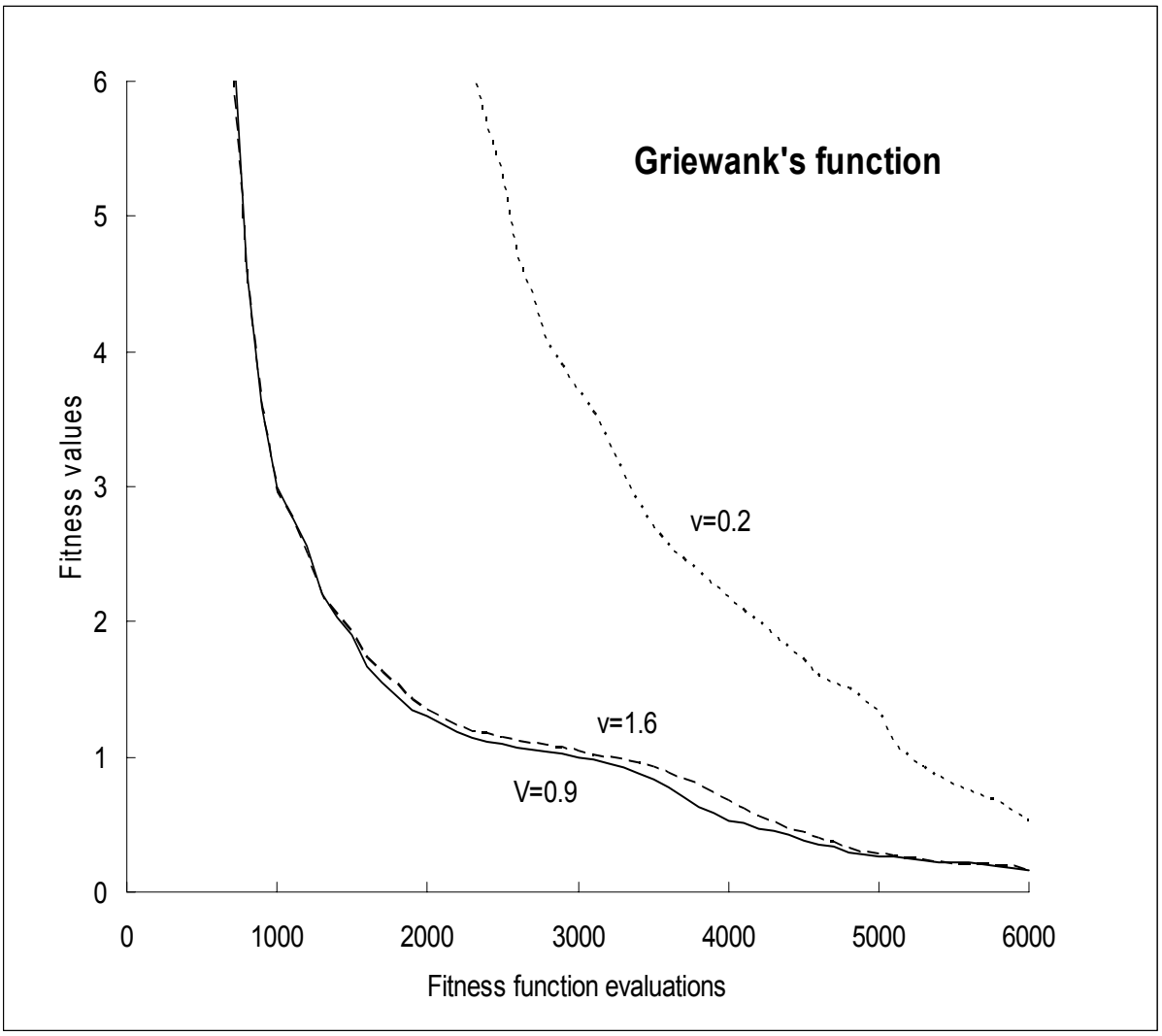

Fig. 3.3 ASAGA performances with different thresholds tested by a 5D Ackley's function

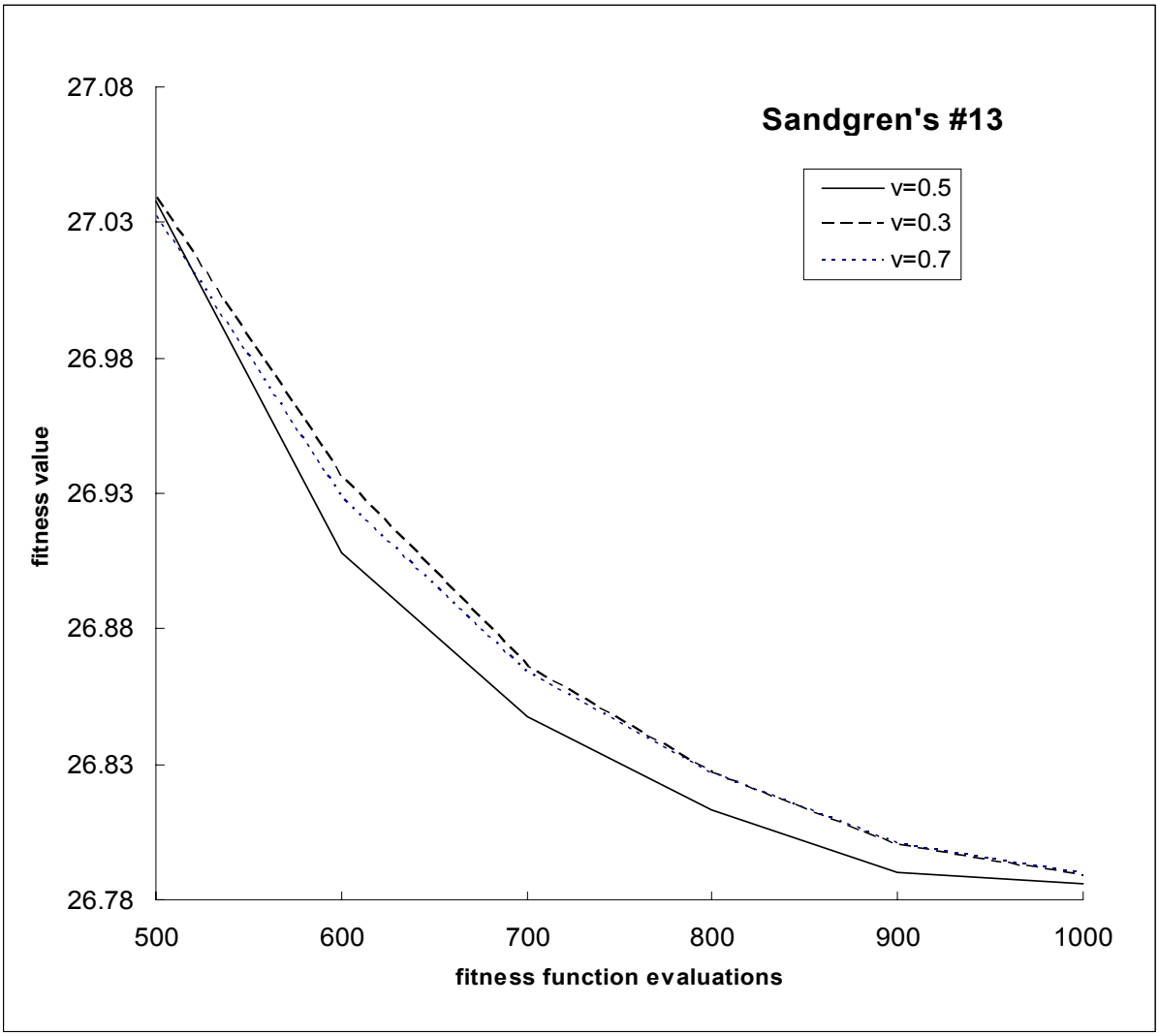

Fig. 3.4 ASAGA performance with different thresholds for constraint violations tested by the Sandgren's number 13 domain

Threshold	Mean	SD
0.7	26.789	0.0102
0.5	26.786	0.0107
0.3	26.789	0.0124
T-test (0.5, 0.3)	0.1602	

Table 3.3 Statistical analysis of pre-threshold selection for violation function tested by Sandgren's domain number 13

A model update is triggered if the v value in formula (15) is smaller than the pre-set threshold. In addition, the objective function threshold value of 0.9 suggests that as long as a surrogate has an R value equal to or greater than 0.9, ASAGA will always keep the surrogate in the current level, which is meaningful. ASAGA also provides a way for user to modify the threshold. If a user doesn't care about the surrogate phase time (for example, when the user is dealing with difficult optimization problems), the user can set a large threshold (if the user sets the threshold to 2 or larger, ASAGA will always go to the final models for both global and local models), which will make the surrogates evolve to higher levels faster. The downside is that a large threshold will lead to unnecessarily long optimization time in some cases.

Adaptive Clustering Technique:

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 to fit because a lack of 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, a lack of sample points may result in a singularity problem. On the other hand, a model with too few clusters loses the benefit of the global and local model system. 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 the number of sample points reaches twice the population size, the first two clusters are created by splitting the current sample points into two clusters via the K-means algorithm. Then it is periodically determined whether 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, more sample points than these minimal numbers are needed 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 (16) above.

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

1. The cluster has 3.5 times more sample points than the minimal requirement for its next level model.
2. The R of this cluster is lower than certain threshold. In ASAGA, this value is 0.9 for a fitness function and 0.5 for a violation function (in statistics, R=0.9 is considered a good correlation, while R=0.5 is considered a fairly good correlation).

The reason for using the specific value of 3.5 is that we found it to be the minimal number for which the singularity problem does not occur. 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 equal the population size.

Adaptive Surrogate 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. The best mutation according to that ranking is then evaluated using the actual fitness function. However, GADO with surrogate assistance uses a fixed number of such random mutations. A fixed number is not suitable for all problems. If not enough mutations are performed, the model will not gain 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:

$$N = Round \quad \left(\frac{R}{C \times (T_1 / T_2)}\right) \tag{19}$$

where N is the number of the potential children with a minimum of 1, R, *T1 and T2* are defined in the same manner as in (15), and C is a small constant which was set to 0.2 based on experimental results. Fig. 3.5 shows ASAGA performances on Ackley's function for different values of C. 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. Like we chose the threshold value in the previous section, the user can adjust this parameter value. Table 3.4 shows that a statistically significant difference does not exist among different values of C when C is within a reasonable range. If the user doesn't care about surrogate phase time (e.g., when the user deals with difficult optimization problems), the user can always set a small C value (in practice, C=0.02 is a small enough value to let ASAGA generate more than 20 offspring from two parents, though this requires more computational time). Table 3.5 shows a further experiment on a higher dimension Griewank's function, where we can see that C=0.2 has no statistical difference from C=0.02, whereas C=2 has much worse performance. Fig. 3.6 shows optimization trends. We keep a C value of 0.2 in ASAGA for the following experiments in this paper.

Parameter C	Mean ($\times 10^{-3}$)	Standard Deviation ($\times 10^{-3}$)
0.3	5.602	3.636
0.2	4.938	3.878
0.1	6.197	3.781
T-test between 0.2 and 0.1		0.1040

Table 3.4 Statistical analysis of parameter C selection tested by a 5D Ackley's function

Parameter C	Mean	Standard Deviation
0.2	0.1579	0.0480
0.02	0.1384	0.0510
2	0.2101	0.0735
T-test between 0.2 and 0.02		0.1327
T-test between 0.2 and 2		0.0021

Table 3.5 Statistical analysis of parameter C selection tested by a 20D Griewank's function

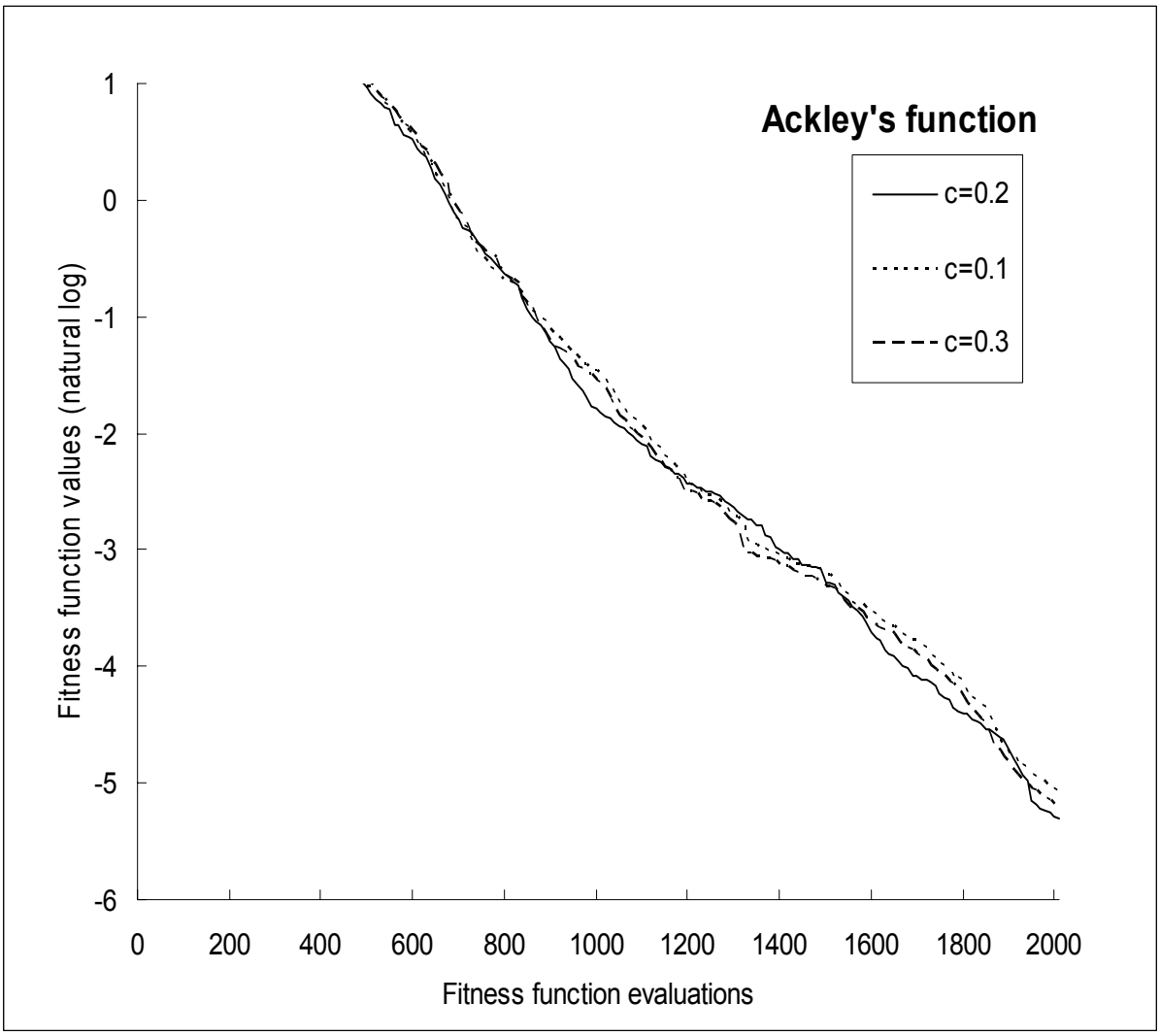

Fig. 3.5 ASAGA performance with different parameter C tested by a 5D Ackley's function

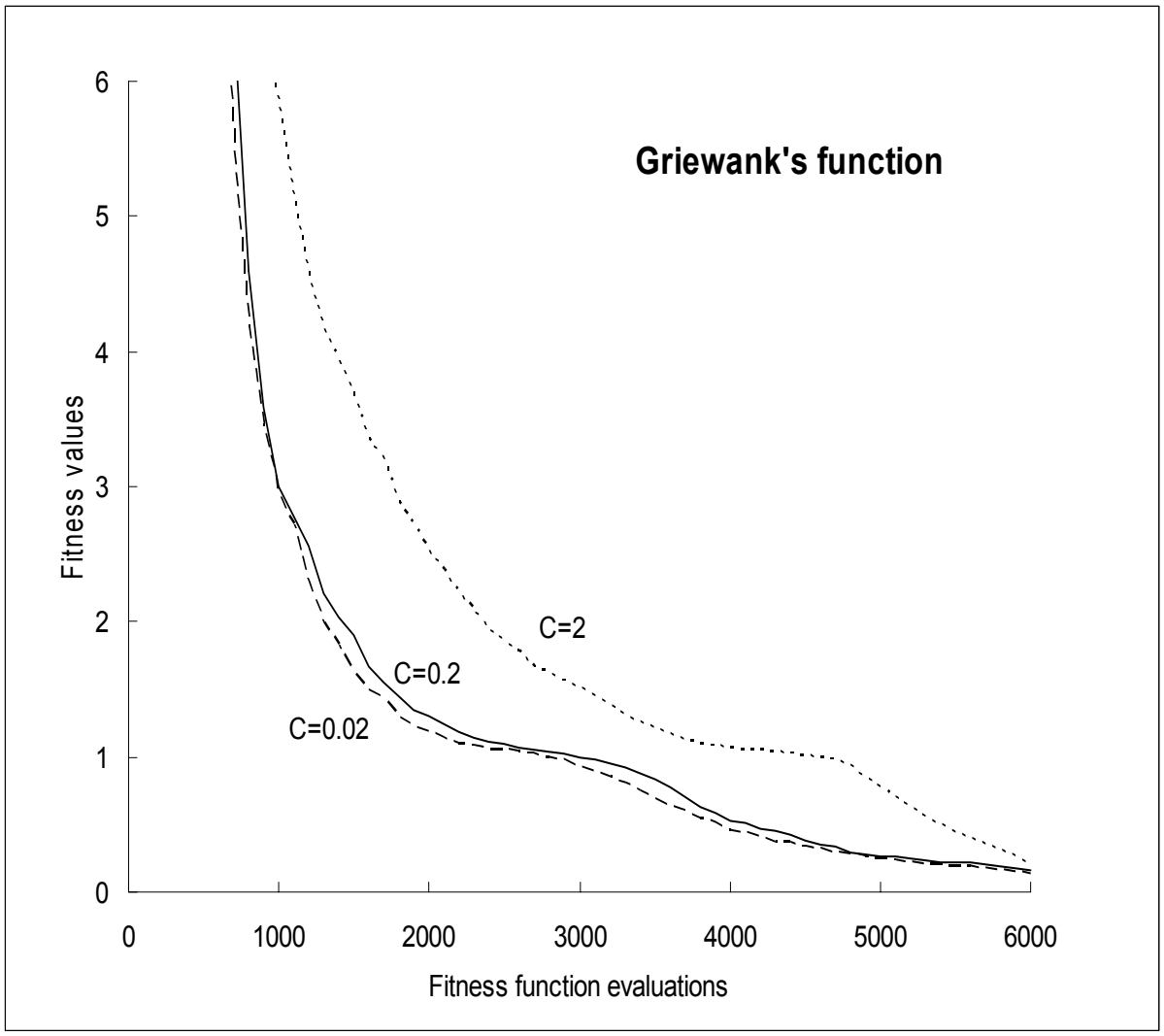

Fig. 3.6 ASAGA performance with different parameter C tested by a 20D Griewank's function

Stochastic Constraint Handling:

ASAGA inherits the penalty function method used in GADO to deal with 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 determined by the sum of the violations of

all 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, it then checks whether any feasible points have been found. If so, the objective function alone is considered in the fitness, placing more pressure on objective improvement. On the other hand, if feasible points have not been found, ASAGA uses only the violation function as fitness, placing more pressure on finding feasible points.

Chapter 4

Experimental Support

Benchmark Unconstrained Functions:

In this section ASAGA is compared with three surrogate-assisted EA methods published in [43], [46], and [59]. In this section, for every experiment the population size was set to its default value of 2.5 times the problem dimension (the population size can be increased by the user according to problem difficulty). The EA used in [43] is assisted by a one-type fixed surrogate whereas global and local model structure is used in [46]. In [59], local models are used to help the optimization. These are all state-of-the-art methods with demonstrated strength in several benchmark domains and/or real-world applications. However, their surrogates are fixed and no adaptive modeling or usage techniques have been applied in them. We used these methods for comparison because, as we mentioned earlier, no adaptive surrogate-assisted methods were found in literature.

1. Comparison with a Regression Model Assisted EA

In [43], both interpolation and regression models have been used to estimate the fitness. Only a part of the population is evaluated with the exact fitness function and the remaining individuals are evaluated with the approximate models. Experiments show that both approximate models speed up population convergence. The empirical results in [43] showed a slight performance advantage for the regression method. ASAGA is compared here with the better method which is the regression EA. Ackley's function and Fletcher-Powell's function have

been used as benchmark domains in [43] with 5-dimensions and 2000 true fitness function evaluations. ASAGA is tested in the same benchmark domains and under exactly the same conditions. The Ackley function is defined in Equation (17) above and the Fletcher-Powell function definitions are as follows.

$$f_{Fletcher\ -\ Powell}(x) = \sum_{i=1}^{n} (A_i - B_i)^2$$

Where

$$A_i = \sum_{j=1}^{n} (a_{ij} \sin \alpha_j + b_{ij} \cos \alpha_j)$$

$$B_i = \sum_{j=1}^{n} (a_{ij} \sin x_j + b_{ij} \cos x_j)$$

$$-100 \le a_{ii}, b_{ij} \le 100$$

$$-\pi \le x_j, \alpha_j \le \pi \qquad (20)$$

The Ackley's function corresponds to a simple function overlaid with high frequency noise and it reaches its global minimum of zero when all x_i are equal to zero. The Fletcher-Powell function is smoother, but not symmetric and has randomly distributed zero minima when every x_j equals α_j .

In the experiment, ASAGA ran 30 times using random starting populations. The fitness of the best individual according to the exact fitness function was recorded at certain function evaluations for each run. The average best fitness values of the 30 runs with corresponding

number of actual fitness evaluations are shown. The data for the regression EA were provided to us through the courtesy of the authors of [43].

Figure 4.1 and 4.2 show ASAGA and the regression EA running in the two benchmark domains, respectively. ASAGA global model goes to the SVM model for the Ackley's function and stays at quadratic PM for the Fletcher-Powell function. ASAGA local models go to the quadratic PM for both functions. In the Ackley domain we can see that ASAGA performs slightly better than the Regression EA at the early stages and obviously outperforms the Regression EA from the middle stage to the end. In the Fletcher-Powell domain, ASAGA performs much better throughout all stages of the optimization.

2. Comparison with a Global and Local Model Assisted GA: SAGA-GLS

SAGA-GLS is introduced in [46]. It is a surrogate-assisted evolutionary optimization framework which combines both global and local surrogate models for solving computationally expensive problems. The first level of the optimization framework involves a strategy that employs a Data Parallel Gaussian Process (DPGP) surrogate model to identify the promising individuals in the EA population. The DPGP approach was devised to reduce the high computational costs associated with Gaussian Process modeling. Subsequently, the promising individuals undergo Lamarckian learning based on a trust-region enabled gradient-based search strategy that accelerates local search using computationally cheap RBF surrogate models. Lamarckian learning forces the genotype to reflect the results of improvement by replacing the locally improved individual back into the population to compete for reproductive opportunities. Numerical results suggest that SAGA-GLS converges to good designs on a limited

computational budget with significant savings in computational cost compared to traditional evolutionary algorithms and other surrogate-assisted optimization frameworks [46].

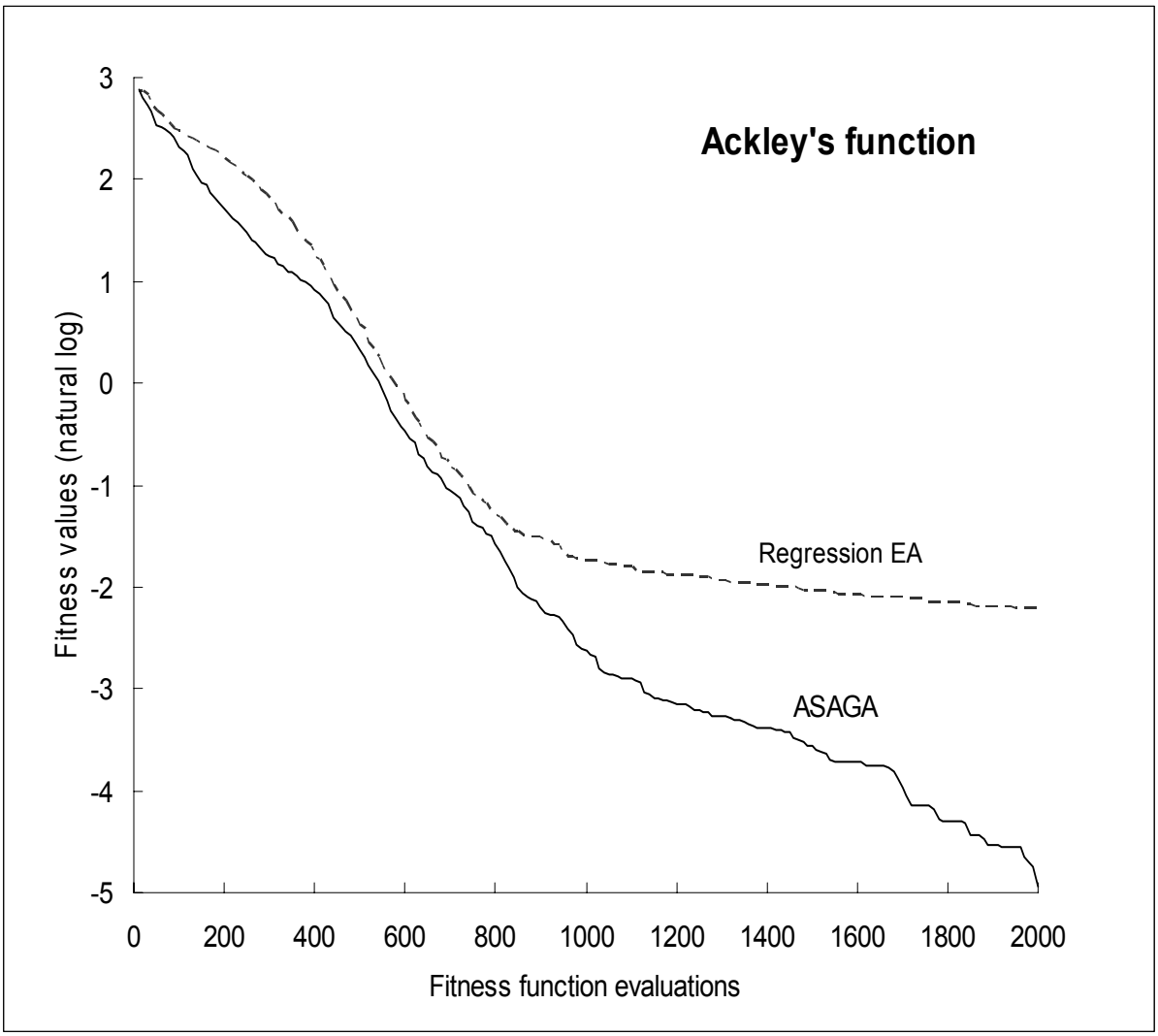

Fig. 4.1 Comparison of ASAGA and Regression EA for a 5D Ackley's function with 2000 function evaluations

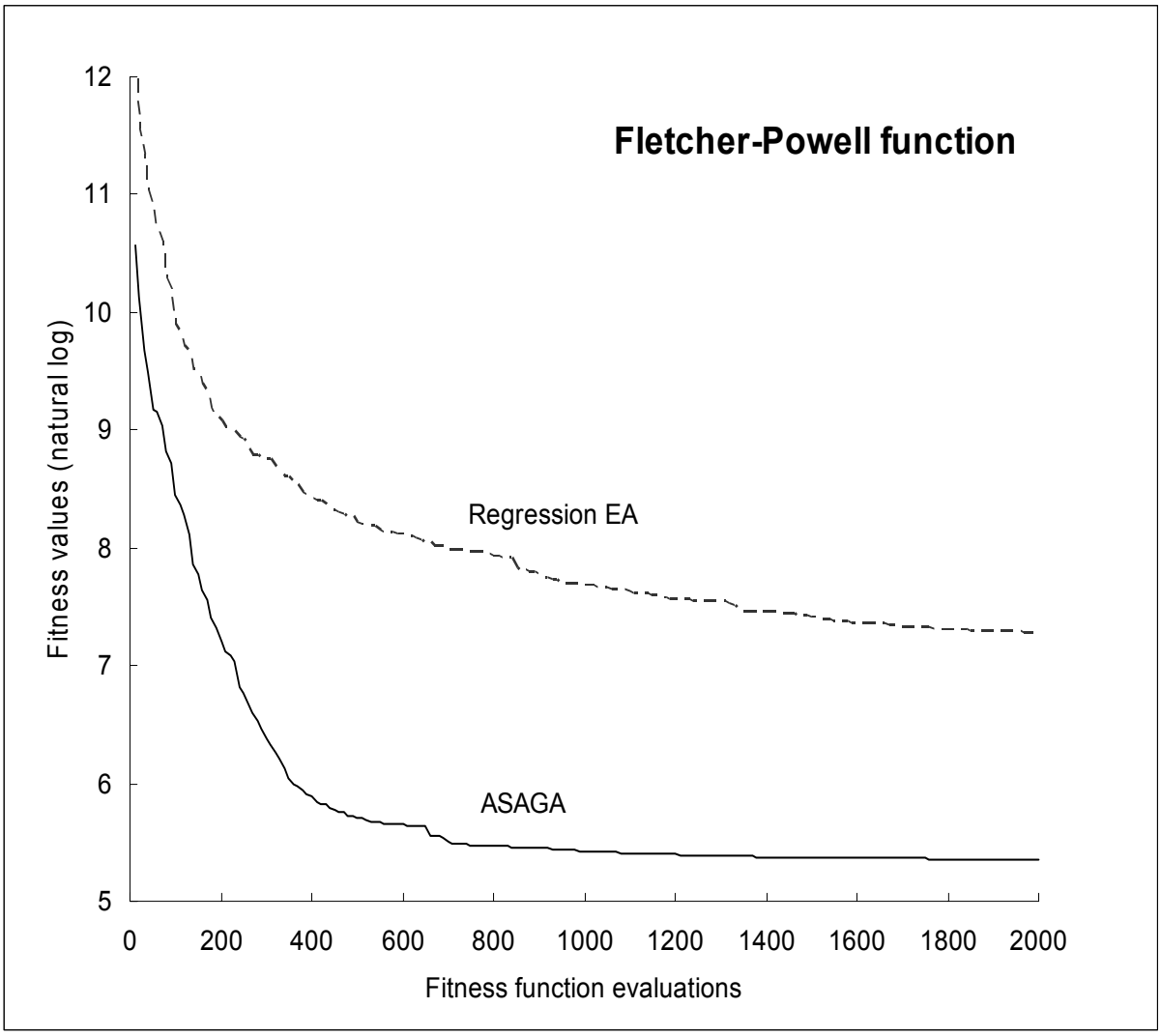

Fig. 4.2 Comparison of ASAGA and Regression EA for a 5D Fletcher-Powell's function with 2000 function evaluations

In this work ASAGA is compared to SAGA-GLS in five function domains: Ackley's, Griewank's, Rastrigin's, Rosenbrock's and Sphere functions – the benchmark domains used in [46]. ASAGA is tested with the number of true fitness function evaluations set to 6000 and all the dimensions set to 20, again exactly the same as in [46]. In this experiment, ASAGA ran 30

times and the average fitness was reported as described previously in the previous section. The data for SAGA-GLS were provided to us through the courtesy of the authors of [46]. The Ackley's function is defined the same as in Formula (17) except that the range of the design variable x is [-32.678, 32.678]. The Griewank's function is defined the same as in Formula (18). The other three benchmark functions are defined as formula (21) through (23).

For Griewank's, Rastrigin's and Sphere functions, the global minimum of zero is reached when each x_i equals zero. For Rosenbrock's function, the global minimum of zero is reached when each x_i equals one. The ASAGA global model progresses to the final SVM model for Ackley, Griewank and Rastrigin's functions, but stays at quadric PM in Rosenbrock and Sphere function domains. ASAGA local models go to the final quadratic PM for all test functions.

Figures 4.3 through 4.7 show the comparison between the ASAGA and SAGA-GLS. According to these figures, we can see that ASAGA performs better than SAGA-GLS in Griewank's and Rosenbrock's function domains in all optimization stages except for early periods in which it is slightly worse. In Rastrigin's and Sphere function domains, ASAGA performs worse in most stages but outperforms SAGA-GLS at the end. However, in Ackley's function domain, SAGA-GLS beats ASAGA in all stages except at the beginning. In most of these tests, ASAGA performs worse at the early stages. The reasons are that ASAGA always uses simple surrogates during the early stages of a run and that ASAGA needs enough sample points before it can form large clusters with good approximations. At later stages, when enough clusters are constructed, ASAGA performs much better in terms of convergence speed and the ability to find the global optima. Thus we can conclude that ASAGA's performance is very competitive with SAGA-GLS in these benchmark function domains with better average final

performance in four out of five of them. In the following section, ASAGA will be further compared to an ESRBF algorithm [59] in Ackley's and Rastrigin's function domains, which happen to be the weak part of ASAGA's performance compared to SAGA-GLS.

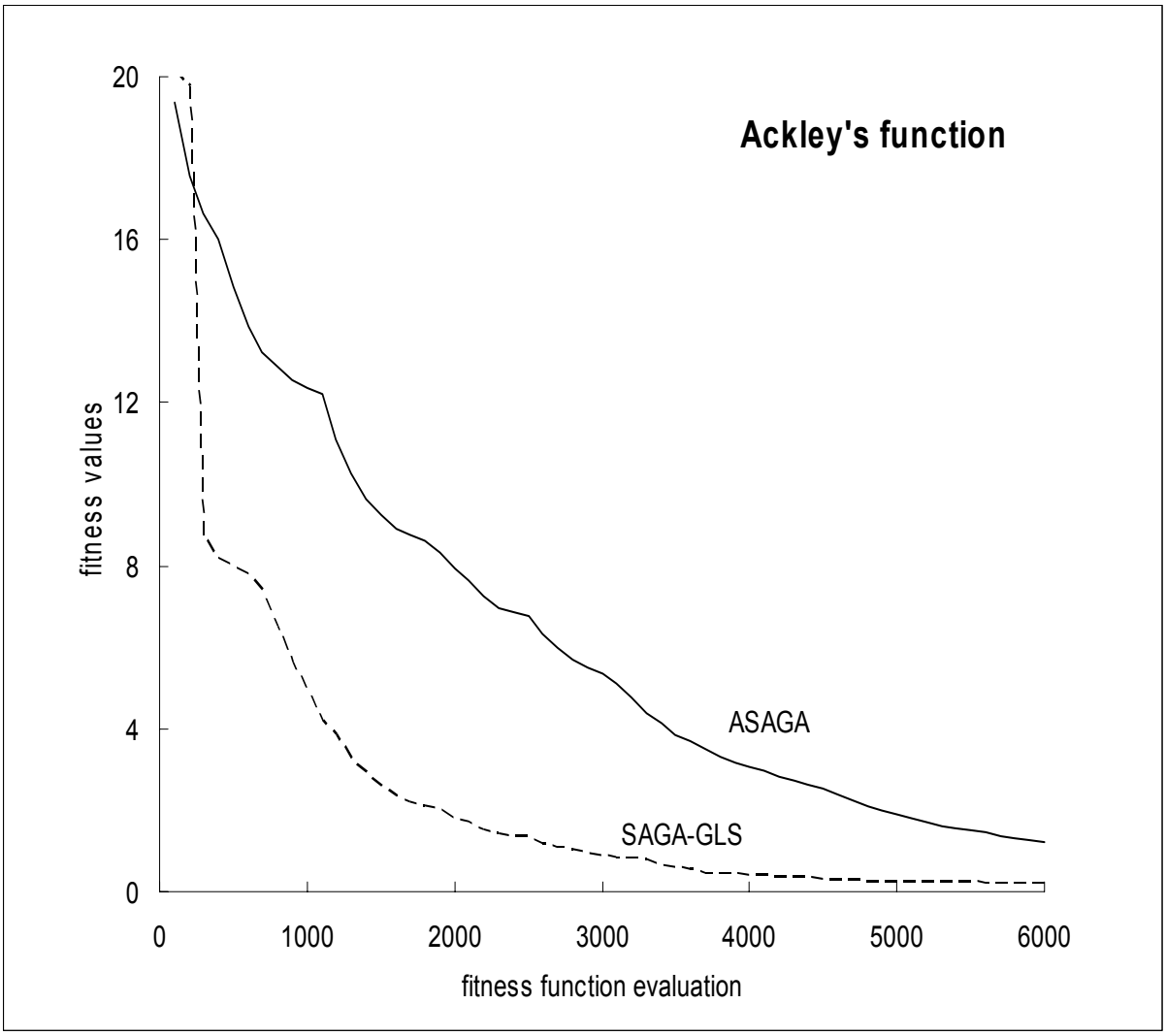

Fig. 4.3 Comparison on a 20D Ackley's function domain

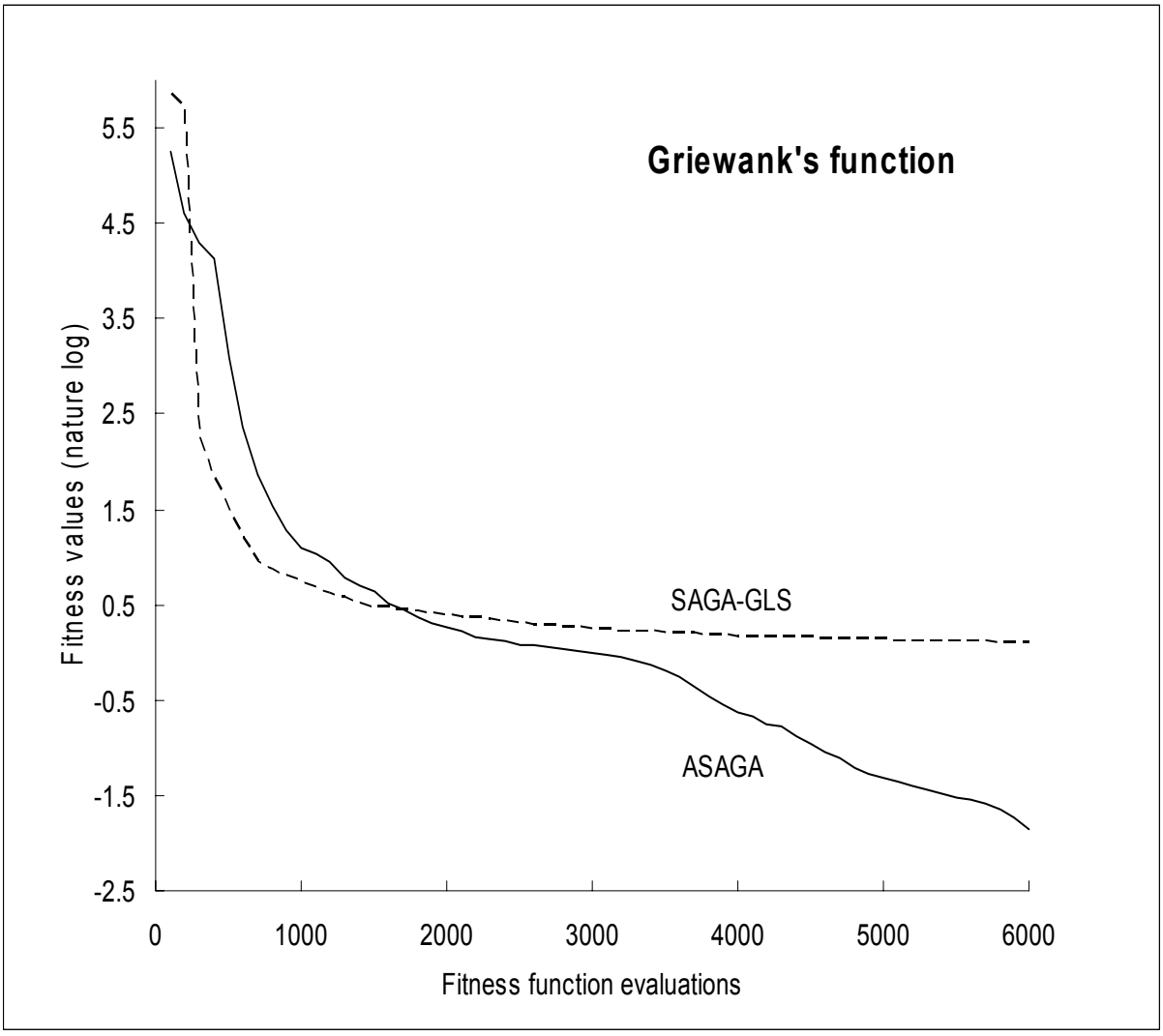

Fig. 4.4 Comparison on a 20D Griewank's function

The Griewank's function is defined as the same as in formula (18).

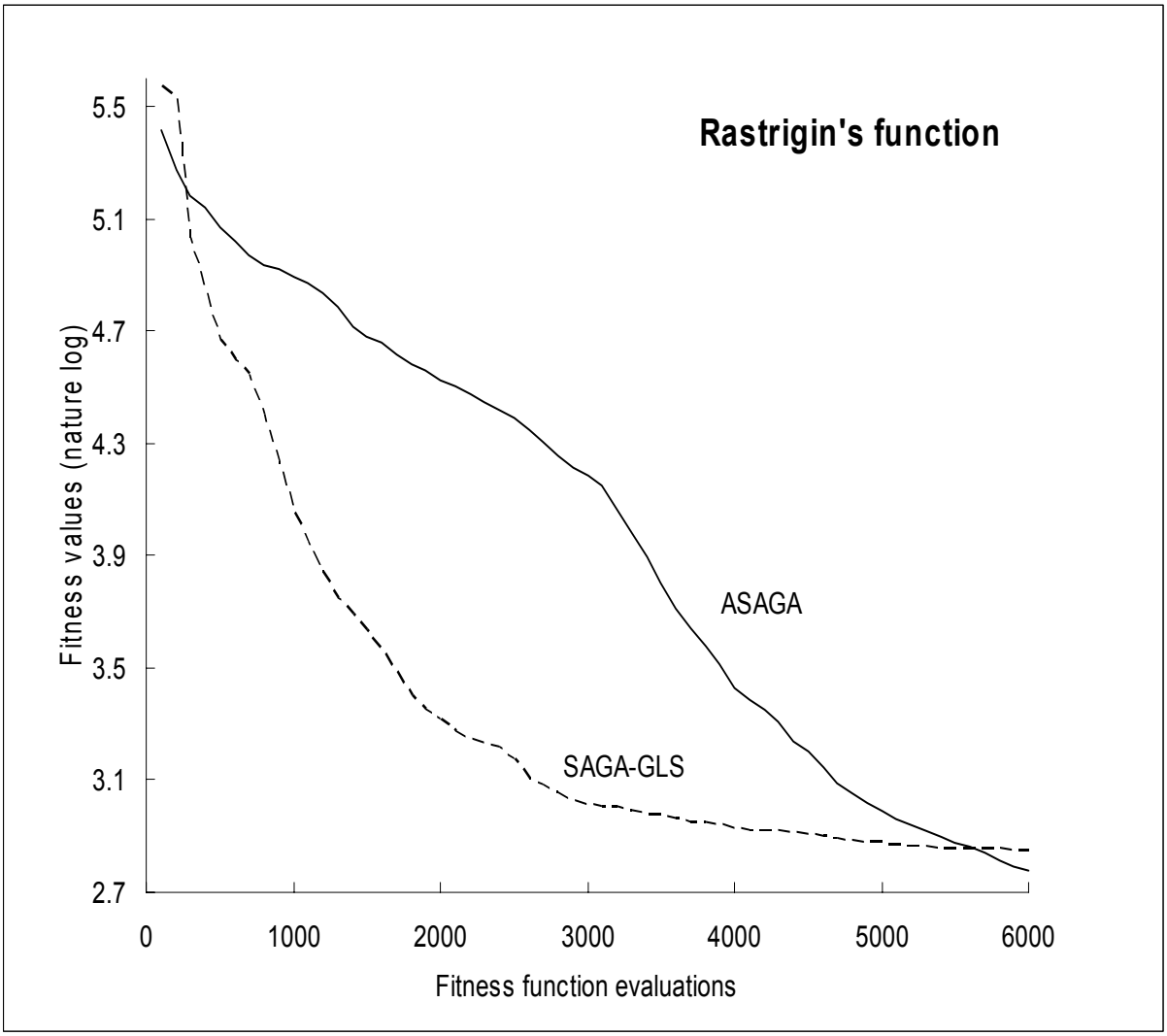

Fig. 4.5 Comparison on a 20D Rastrigin's function

The Rastrigin's function is defined as follow:

$$f_{Rastrigin}(x) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$$

$$-5.12 \le x_i \le 5.12 \qquad (21)$$

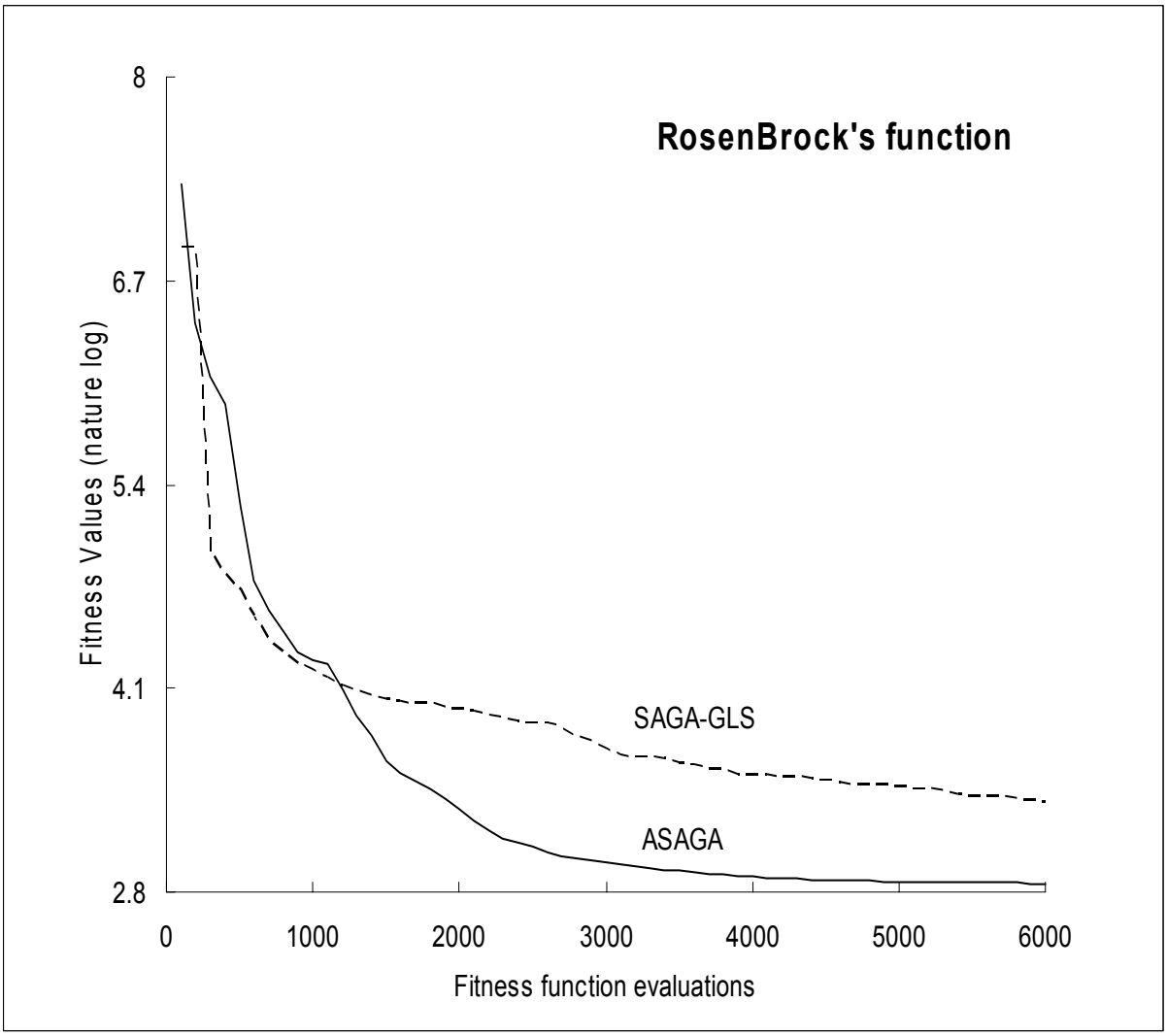

Fig. 4.6 Comparison on a 20D Rosenbrock's function

The Rosenbrock's function is defined as follow:

$$f_{Rosenbrock}(x) = \sum_{i=1}^{n} (100 \times (x_{i+1} - x_i^2)^2 + (1 - x_i^2))$$

$$-2.048 \le x_i \le 2.048 \qquad (22)$$

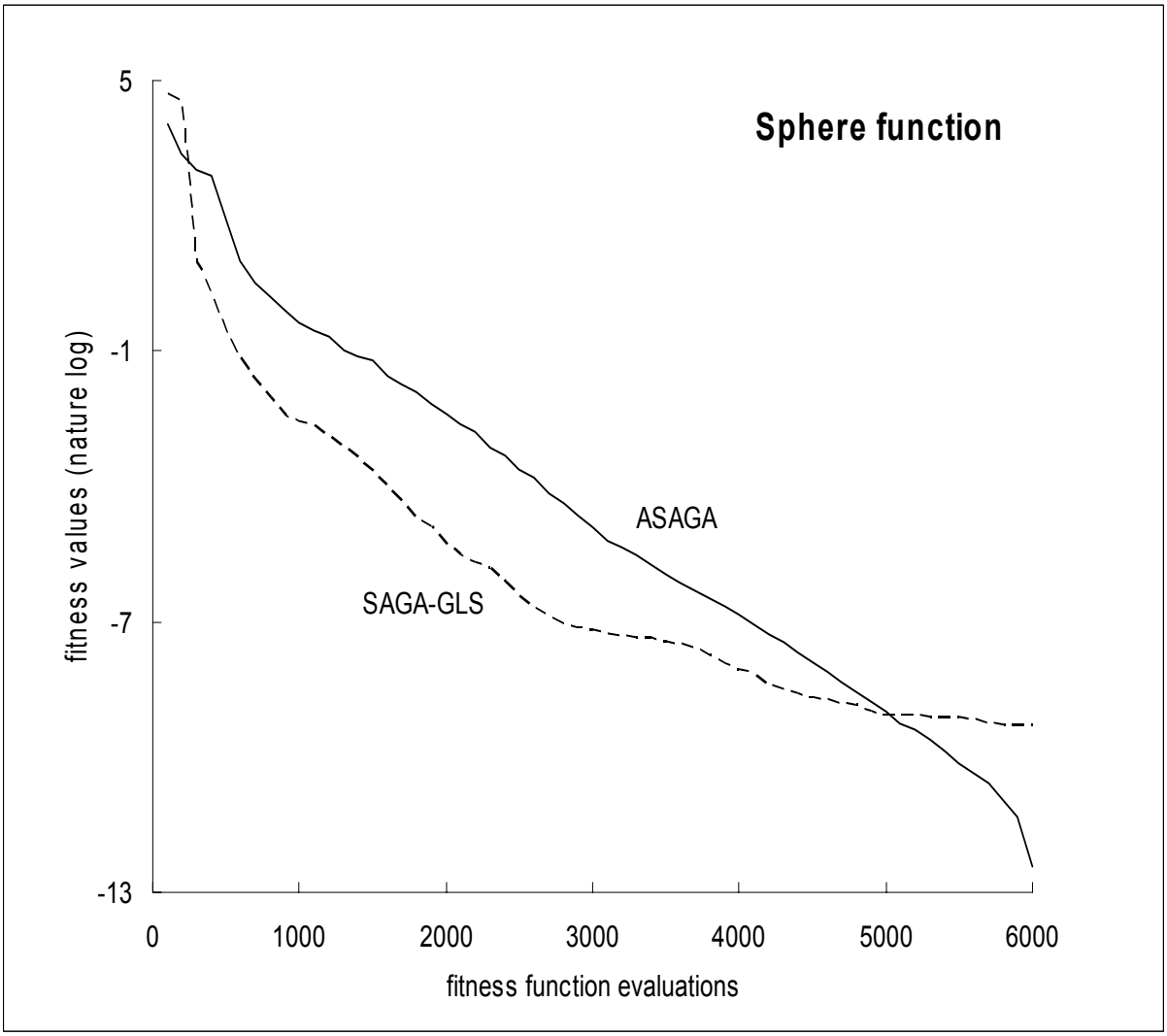

Fig. 4.7 Comparison on a 20D Sphere function

The Sphere's function is defined as follow:

$$f_{Sphere}(x) = \sum_{i=1}^{n} x_i^2$$

$$-5.12 \leq x_i \leq 5.12 \qquad (23)$$

3. Comparison with a Local Model Assisted EA: ESRBF

A local surrogate-assisted evolution strategy (ESRBF) is introduced in [59]. In this method, for a new offspring, a cubic RBF surrogate is built using the *k*-nearest previously evaluated points, where $k = (d+1)(d+2)/2$ and *d* is the dimension of the problem. Then, this local RBF surrogate is used to estimate the new offspring's fitness. In addition, a symmetric Latin hypercube design (SLHD) is used to determine the initial points for function evaluation. In [59], experimental results showed that ESRBF was significantly better than conventional evolution strategies (ES), ES with SLHD and quadratic model assisted ES (ESQR) in a real world domain and two benchmark domains. The author pointed out that only ESRBF always succeeded in improving the performance of the ES and was superior to ESQR on all test functions. In this study, we therefore compare ASAGA with ESRBF in Ackley's and Rastrigin's functions, which were used as the test functions in [59]. The definition of these two functions in [59] was slightly different from what we used in section 4.1.2. They were defined as follows.

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

$$-1 \le x_i \le 3 \tag{24}$$

$$f_{Rastrigin}(x) = \sum_{i=1}^{n} (x_i^2 - \cos(2\pi x_i))$$

$$-1 \le x_i \le 3 \tag{25}$$

Notice that the only difference is that the constant part is taken out in formula (24) and (25). By this definition, Ackley's function has a global minimum of -20-e and Rastrigin's function has a global minimum value equal to the negative of the problem dimension. ASAGA was tested on a 10 dimensional version on these two functions and the total evaluations were set to 2000, which are the same values used in [59]. ASAGA ran 100 times with random starting populations and we reported average best fitness for different stages, exactly the same as in [59]. The data for ESRBF were provided to us through the courtesy of the authors of [59]. Figures 4.8 and 4.9 show the performance traces of ASAGA and ESRBF. These figures show that ASAGA outperforms ESRBF in all optimization stages except the beginning of optimization for Ackley's function.

4. Comparison with Gaussian Process Assisted Evolution Strategies:

A Gaussian Process pre-selection Evolution Strategy (ES) is introduced in [24]. In this ES method more than λ offspring are generated from previous generation, then the Gaussian Process model is used to pre-select λ offspring. Only these λ offspring are evaluated by the exact fitness function. The author introduced two selection criteria, Mean of Model Prediction (MMP) and Probability of Improvement (POI). The MMP criterion selects based on the mean of the predictions. The POI improves on this by additionally considering the confidence intervals. The author showed that either MMP or POI method is better than the standard ES, and the POI method is even better than the traditional MMP method. In the following section, ASAGA is compared to the MMP and POI methods in three multimodal benchmark domains, which are used in [24]. The experimental setting is exactly the same as what presented in [24]. The figures

for MMP and POI were provided to us through the courtesy of the authors of [24].

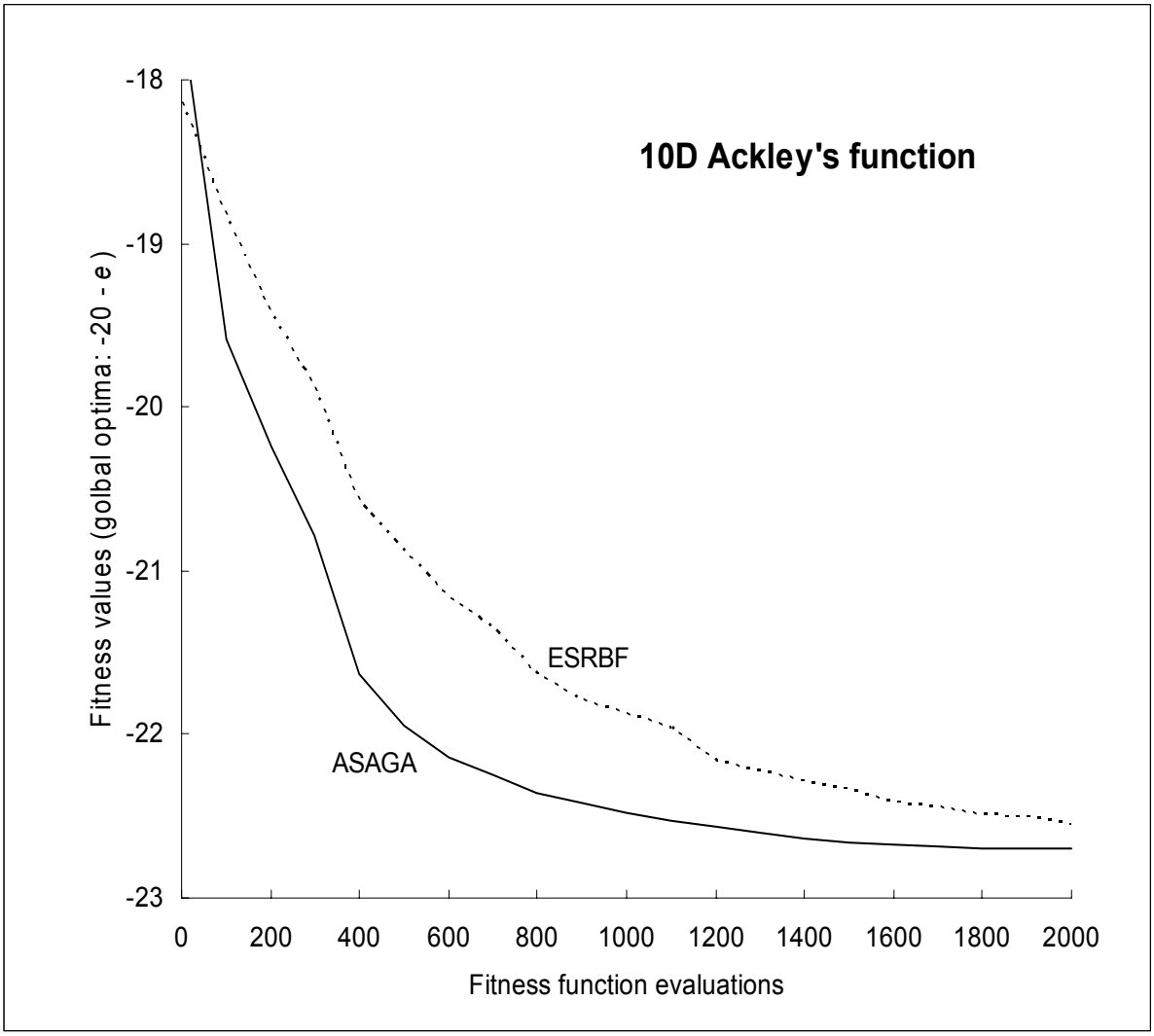

Fig. 4.8 Comparison on a 10D Ackley's function

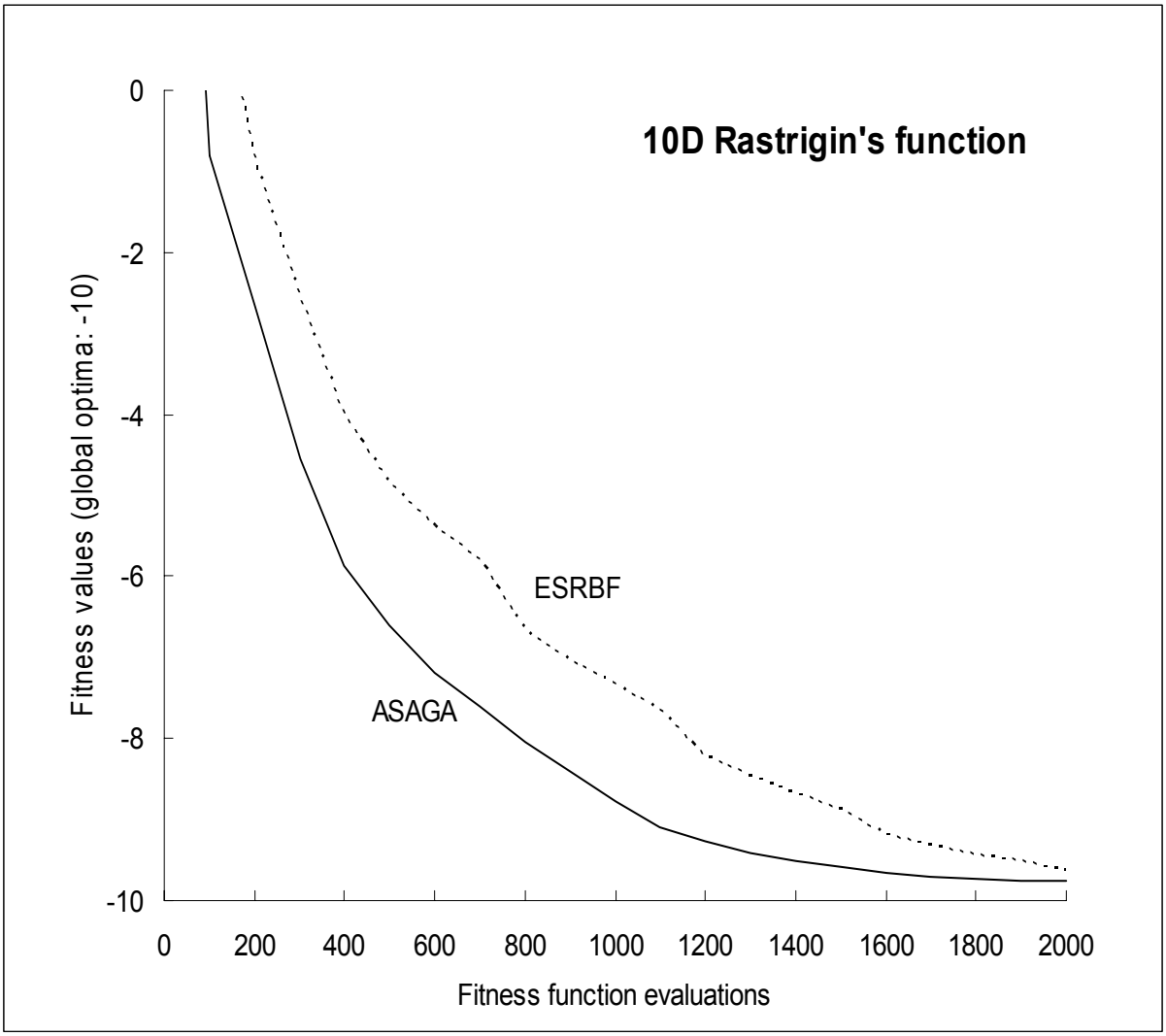

Fig. 4.9 Comparison on a 10D Rastrigin's function

Fig. 4.10 shows ASAGA, MMP (GP MAES) and POI performances on a 20 dimension Step function. We can see that ASAGA finds the global optima before 1000 fitness evaluations whereas POI finds the global optima after 2500 fitness evaluations. The Step function is defined as follows:

$$f_{Step}(x) = \sum_{i=1}^{n} (floor(x_i - 0.5))^2$$

$$-5.12 \leq x_i \leq 5.12 \tag{26}$$

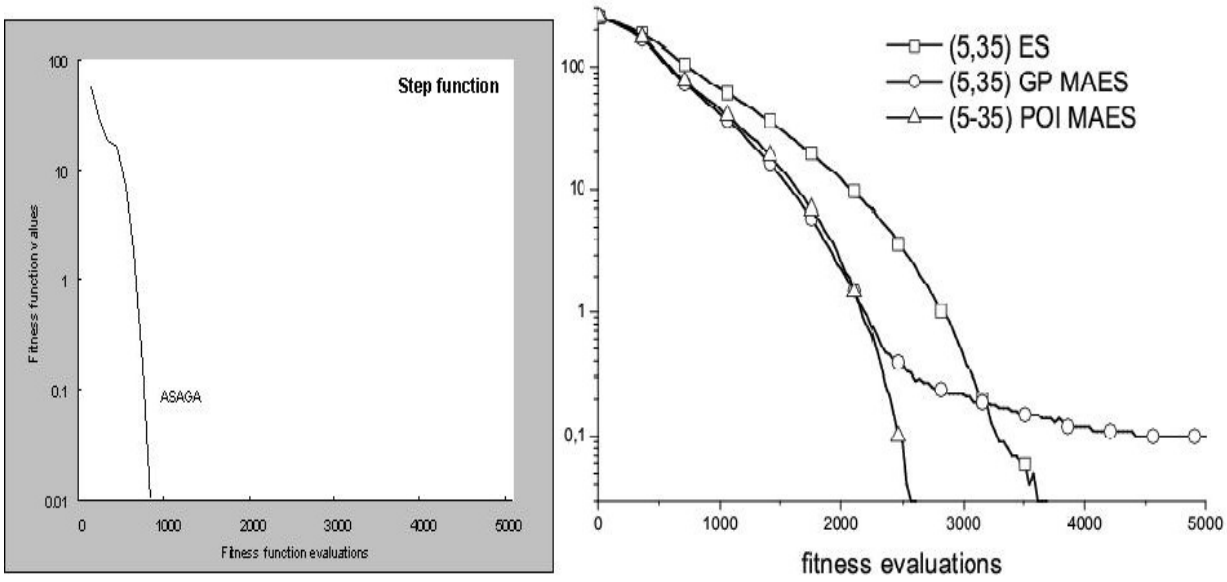

Fig. 4.10 Comparison on a 20D Step function

Fig. 4.11 shows ASAGA, MMP (GP MAES) and POI performances on a 5 dimension Griewank's function. It shows that ASAGA outperforms MMP, but performs a little bit worse than POI. Fig. 4.12 shows the three methods' performances on a 20 dimension Ackley's function. It can be seen that ASAGA outperforms MMP and POI in this domain. ASAGA reached a solution of less than 5 whereas the other two methods did not find any solution less than 5.

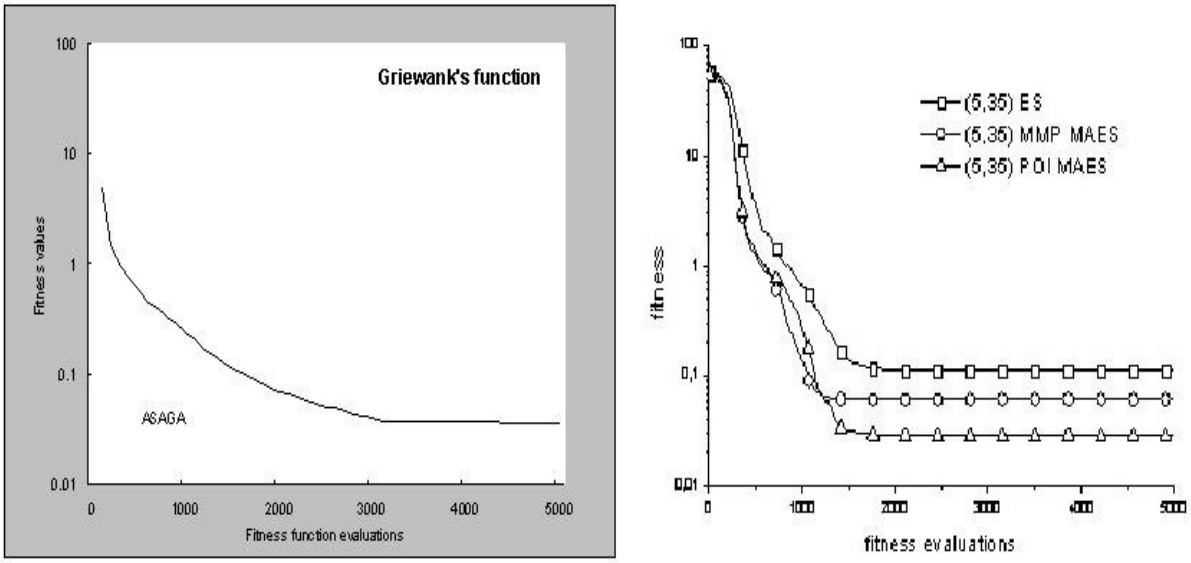

Fig. 4.11 Comparison on a 5D Griewank's function, the Griewank's function is defined previously as Formula (20), except that x ranges in [-100, 100]

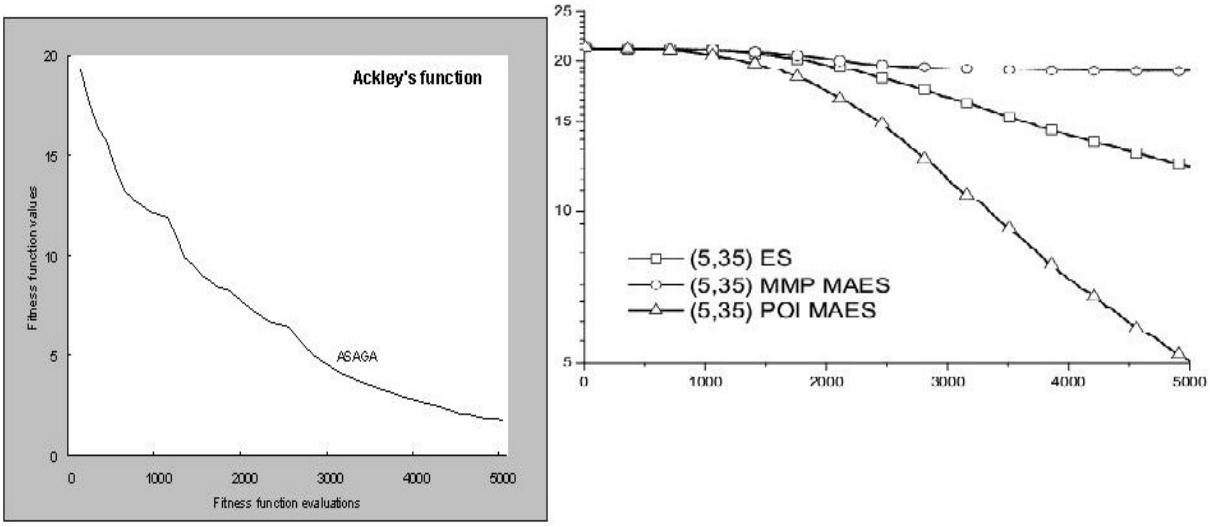

Fig. 4.12 Comparison on a 20D Ackley's function, the Ackley's function is defined previously as Formula (17), except that x ranges in [-32.768, 32.768]

Benchmark Constrained Functions and Engineering Design Domains:

In this section, ASAGA is tested on benchmark functions with constraints and an engineering optimization problem. In section 4.2, for every experiment the population size was set to its default value of 10 times the problem dimension (For constrained functions ASAGA increases its default population size by a factor of 4 compared to the default for unconstrained functions). The first four test domains were introduced by Eric Sandgren in his Ph.D. thesis [48] in which he presented 30 engineering design optimization problems. These domains have since been used as benchmarks for engineering design optimization [71]. Sandgren's domains number 3, number 13, number 21, and number 22 are used to test ASAGA because they are most difficult ones among the 30 benchmark functions and these four functions all have inequity constraints. Readers can refer to Sandgren et al. [48] for details. The fifth experimental domain comes from a real engineering design problem called Welded Beam Design previously used as benchmark in several studies such as [45]. The Welded Beam Design problem is illustrated in Fig. 4.13. The goal for this design is to use the least material to sustain a certain weight. It has four design variables $\vec{x} = (h, j, t, b)$; the definition of Welded Beam Design problem is given below. The known optimal solution is 2.38116 with h = 0.2444, l = 6.2187, t = 8.2915, and b = 0.2444.

Minimize:

$$f_{Welded\ -Beam}(\vec{x}) = 1.10471\ h^2 l + 0.04811\ tb(14 + l)$$

Subject to:

$$13600 - \tau(\vec{x}) \geq 0 \;,\; 30000 - \sigma(\vec{x}) \geq 0 \;,\; b - h \geq 0$$

$$P_c(\vec{x}) - 6000 \geq 0 \;,\; 0.25 - \delta(\vec{x}) \geq 0$$

$$0.125 \leq h \leq 10 \;,\; 0.1 \leq l, t, b \leq 10$$

The terms $\tau(\vec{x})$, $\sigma(\vec{x})$, $P_c(\vec{x})$, and $\delta(\vec{x})$ are given below:

$$\tau(\vec{x}) = \sqrt{\tau'(\vec{x})^2 + \tau''(\vec{x})^2 + l\tau'(\vec{x})\tau''(\vec{x}) / \sqrt{0.25\,(l^2 + (h+t)^2)}}$$

$$\sigma(\vec{x}) = 504000 \;/(t^2 b)$$

$$P_c(\vec{x}) = 64746.022(1 - 0.0282346\,t)tb^3$$

$$\delta(\vec{x}) = 2.1952 \;/(t^3 b) \qquad (27)$$

where

$$\tau'(\vec{x}) = 6000 \;/(\sqrt{2}hl)$$

$$\tau''(\vec{x}) = \frac{6000\;(14 + 0.5l)\sqrt{0.25\,(l^2 + (h+t)^2)}}{2(0.707\;hl\,(l^2/12 + 0.25\,(h+t)^2))}$$

We compared ASAGA to three methods: GADO, GADO with surrogate assistance (GADO-R), and ASAGA without the SVM model (ASAGA-NSVM). GADO was used with no approximate model assistance. GADO-R is based on GADO, and also includes a global and local model structured by clustering techniques. It incorporates fixed approximation models which are quadratic polynomial models through Informed Operators [44, 58]. ASAGA-NSVM is similar to the proposed ASAGA except that its final global model stops at Polynomial and is never 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.

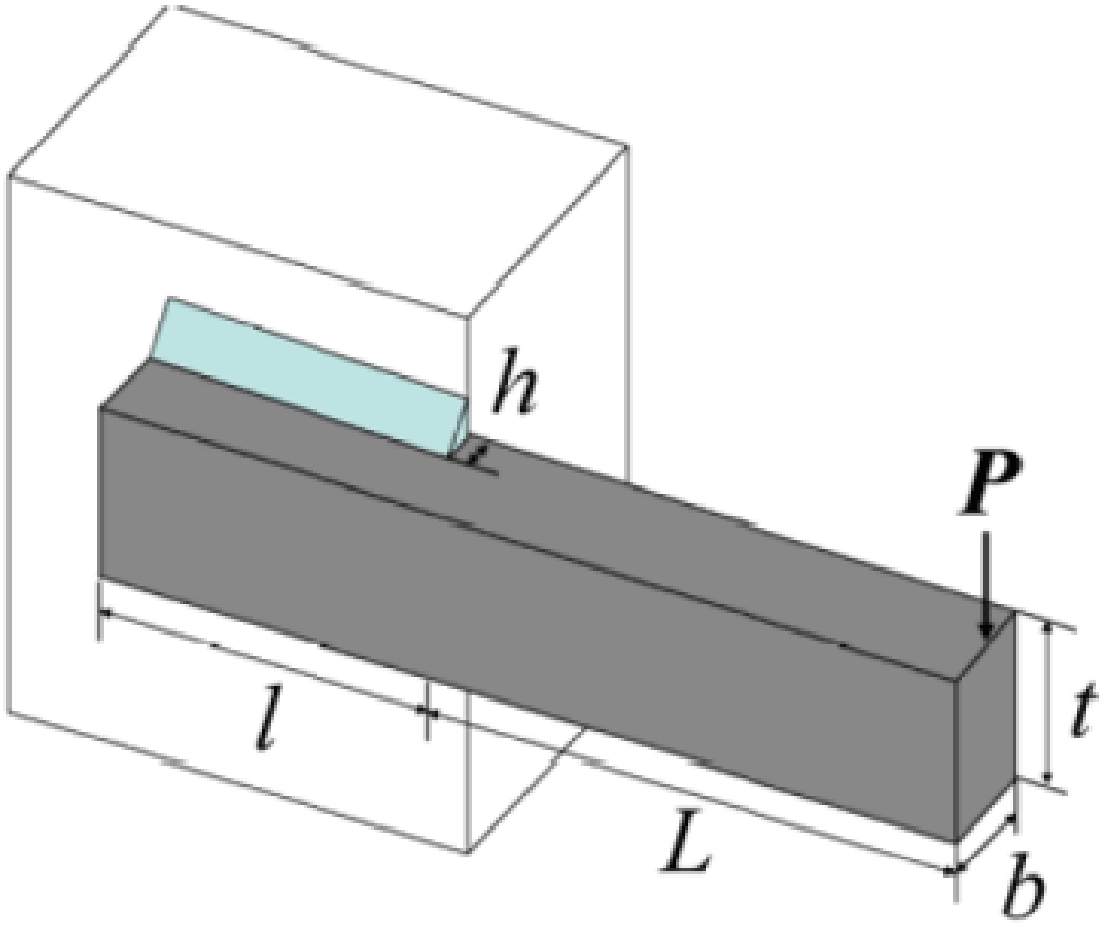

Fig. 4.13 Welded Beam Structure (Source: http://mikilab.doshisha.ac.jp)

Figures 4.14 through 4.18 show the performance of these four methods. Notice that only feasible regions are shown. Thus, the search traces in these figures do not start from the beginning since a certain amount of 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, and ASAGA is slightly worse than the ASAGA-NSVM, though it 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 assistance at all, but the adaptive surrogate-assisted GA outperforms the non-adaptive surrogate-assisted GA.

Table 4.1 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, and some are even less than 0.01, suggesting that ASAGA performs better than GADO-R with statistical significance. Also notice that ASAGA has smaller standard deviations in all domains, which suggests that ASAGA has more stable performance than GADO-R. ASAGA reached the global optima in almost every run in Sandgren's domain number 3 and number 13, which results in very small standard deviations in these two domains.

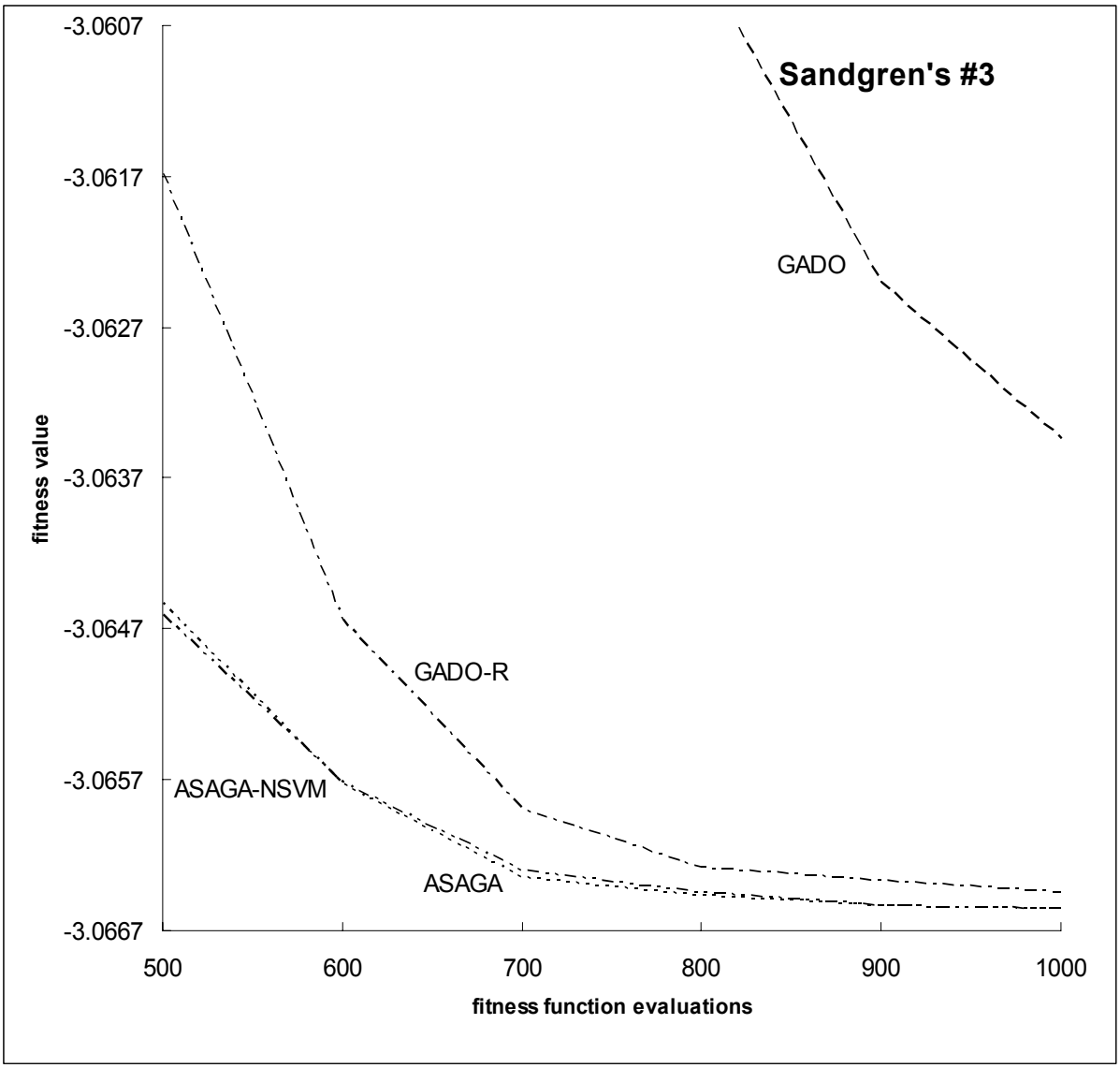

Fig. 4.14 Sandgren function #3 with global optima -3.0666

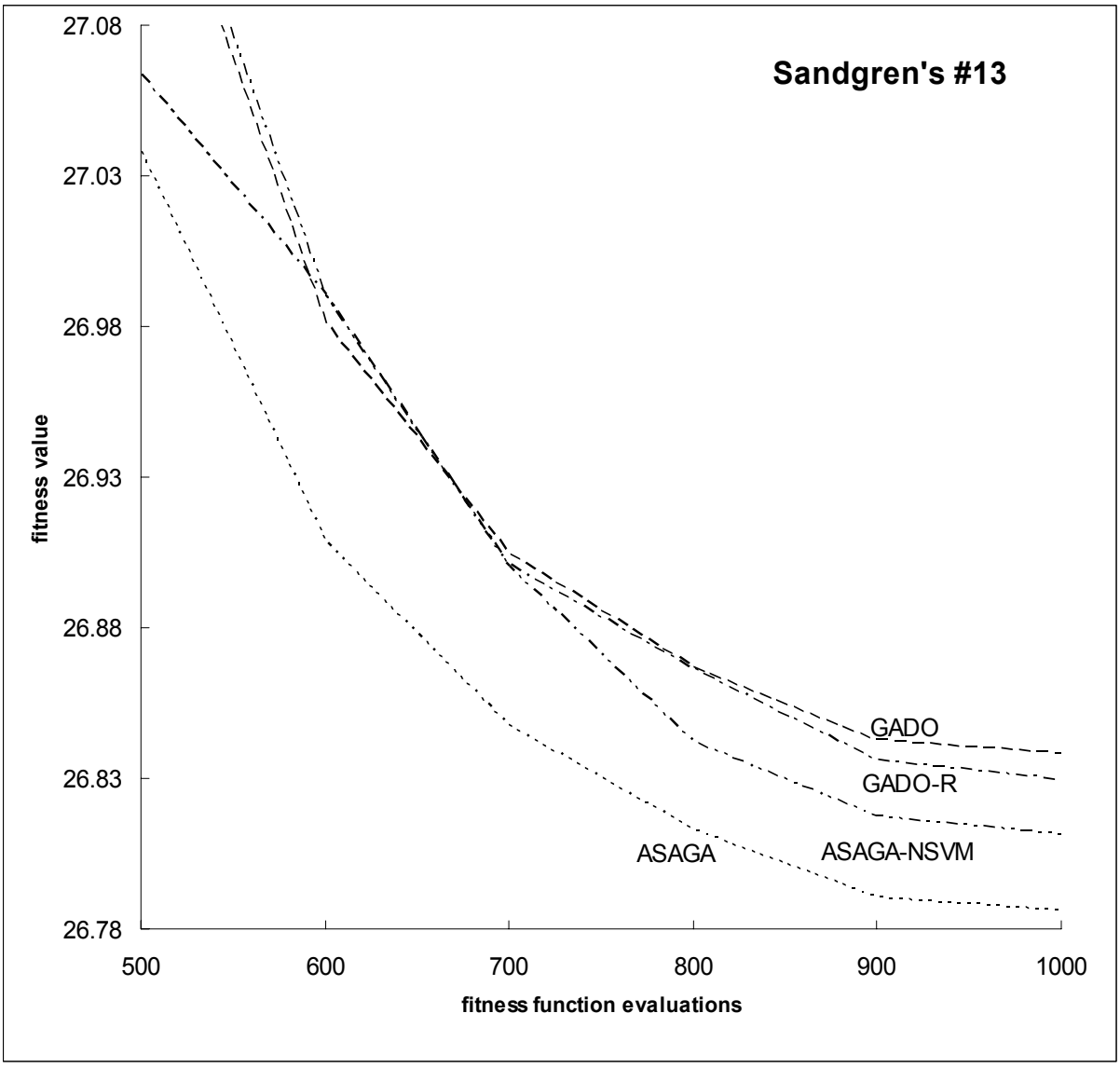

Fig. 4.15 Sandgren function #13 with global optima 26.78

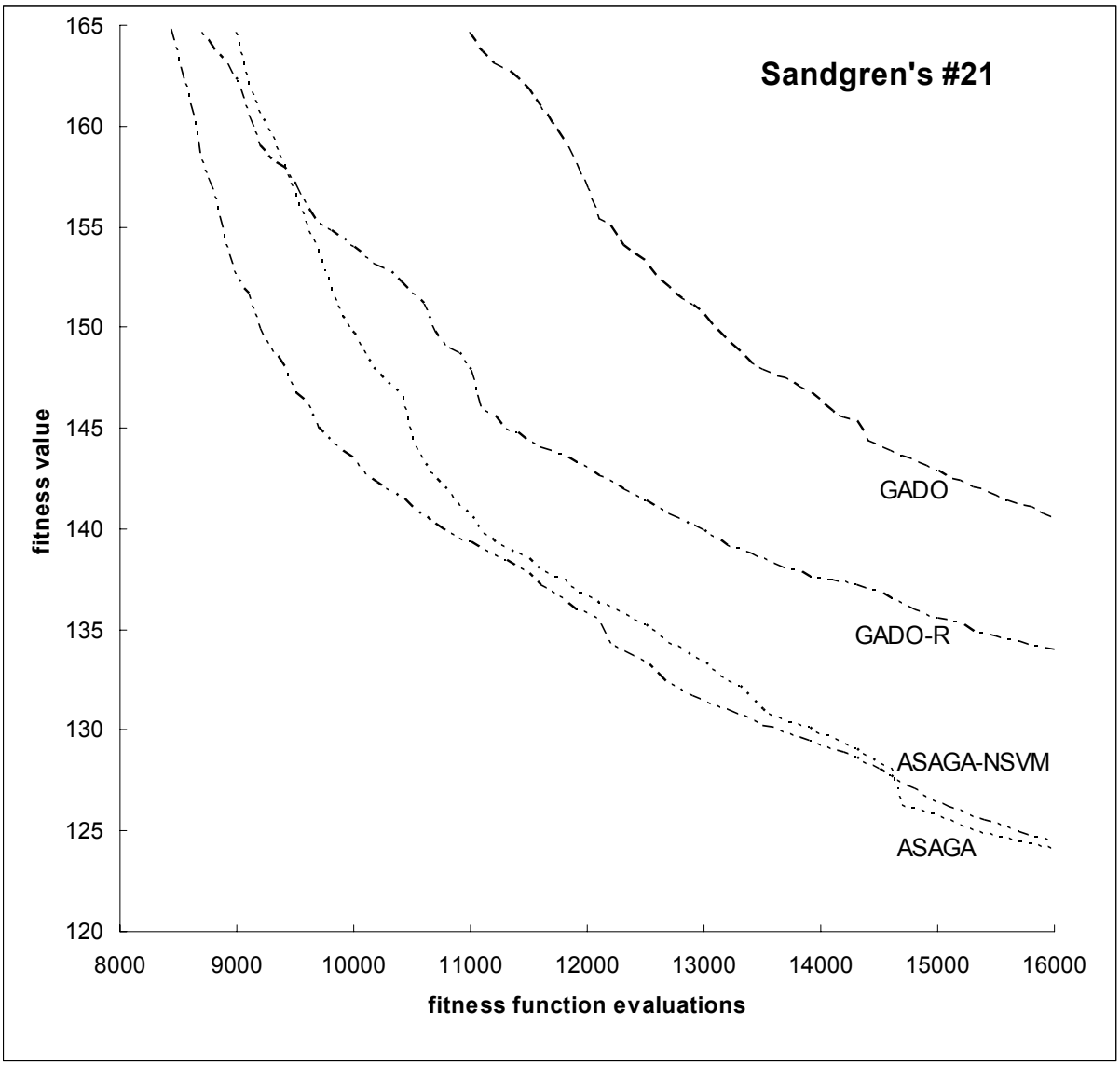

Fig. 4.16 Sandgren function #21 with global optima 97.5

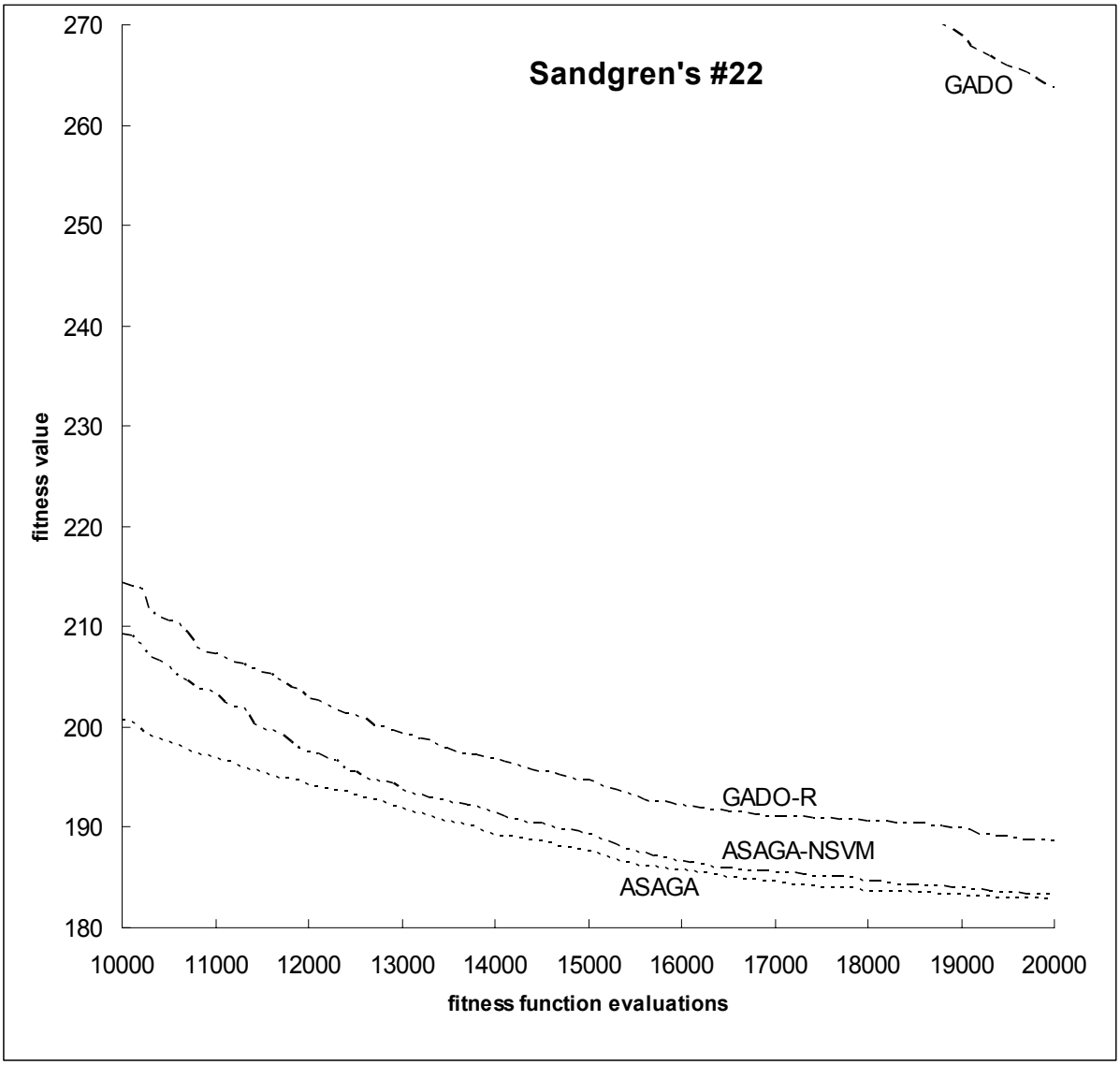

Fig.4.17 Sandgren function #22 with global optima 174.7

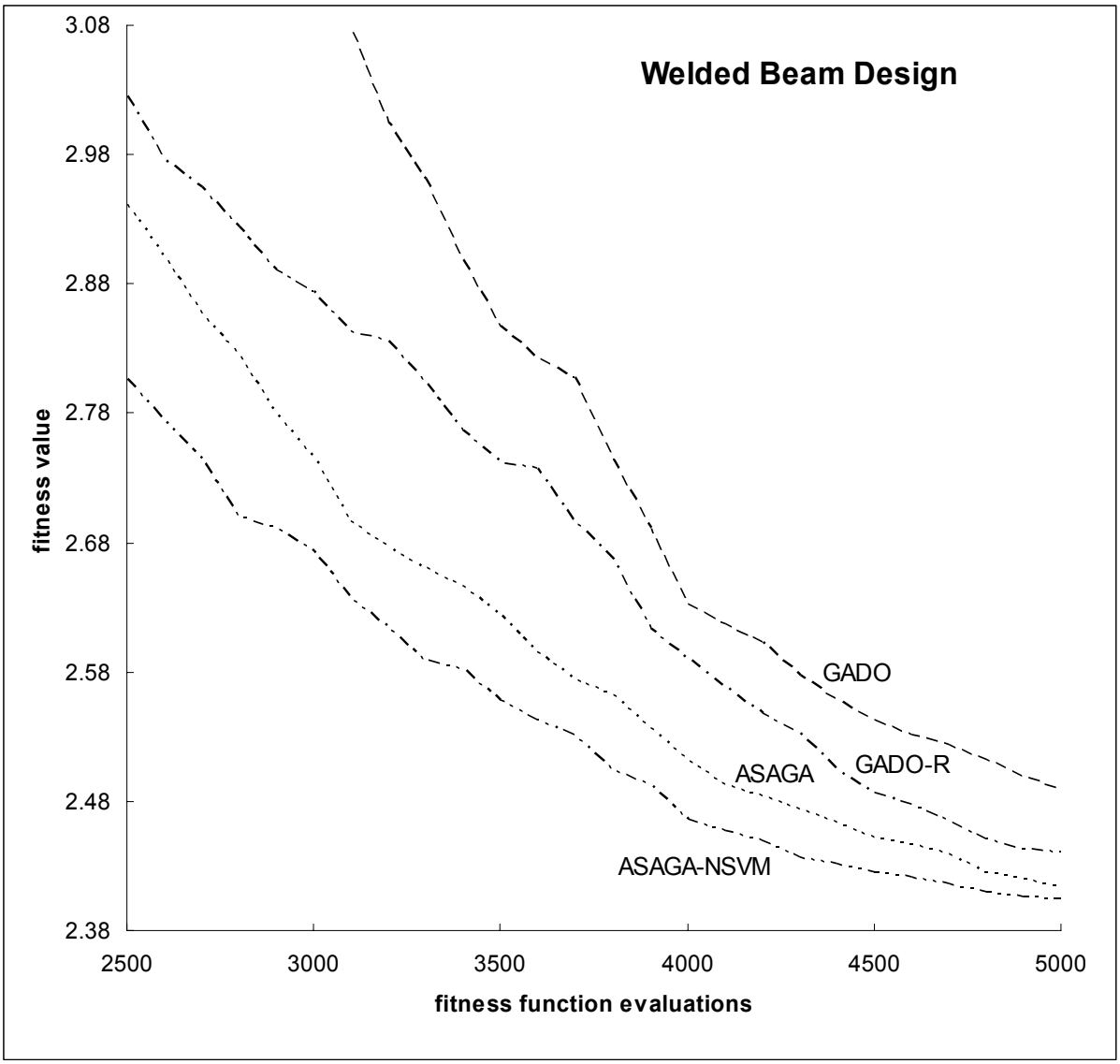

Fig. 4.18 Welded Beam design with global optimum 2.38116

Benchmark		Mean	SD	T-test
#3	GADO-R	-3.0665	0.0002	0.0019
	ASAGA	-3.0666	0.0001	
#13	GADO-R	26.83	0.0637	0.0005
	ASAGA	26.79	0.0107	
#21	GADO-R	135.42	22.72	0.0177
	ASAGA	124.11	17.02	
#22	GADO-R	188.64	13.33	0.0267
	ASAGA	182.74	8.19	
WB	GADO-R	2.4396	0.0439	0.0019
	ASAGA	2.4139	0.0440	

Table 4.1 Statistical analysis of GADO-R and ASAGA in constrained benchmark function domains

Chapter 5

Parameter Tuning by Performance Analysis

Adjusting population size in ASAGA:

Like other GAs, ASAGA's performance depends on its parameter settings, which include population size, surrogate model settings, model build interval etc. However, finding a best setting for all kinds of problems is impossible, since any given parameter setting may work very well in certain problems but poorly in others. In this dissertation, ASAGA is tested with different benchmark functions using different population sizes, population size being the parameter which has the most impact on ASAGA's performance.

First ASAGA was tested in unconstrained benchmark domains. In the experiment, ASAGA was run 30 times using random starting populations. The fitness of the best individual according to the exact fitness function was recorded at certain function evaluations for each run. The average best fitness values of the 30 runs with the corresponding number of actual fitness evaluations are shown. The Ackley's function used in this experiment is the same as formula (17), except that the range of the design variable X is [-32.678, 32.678].

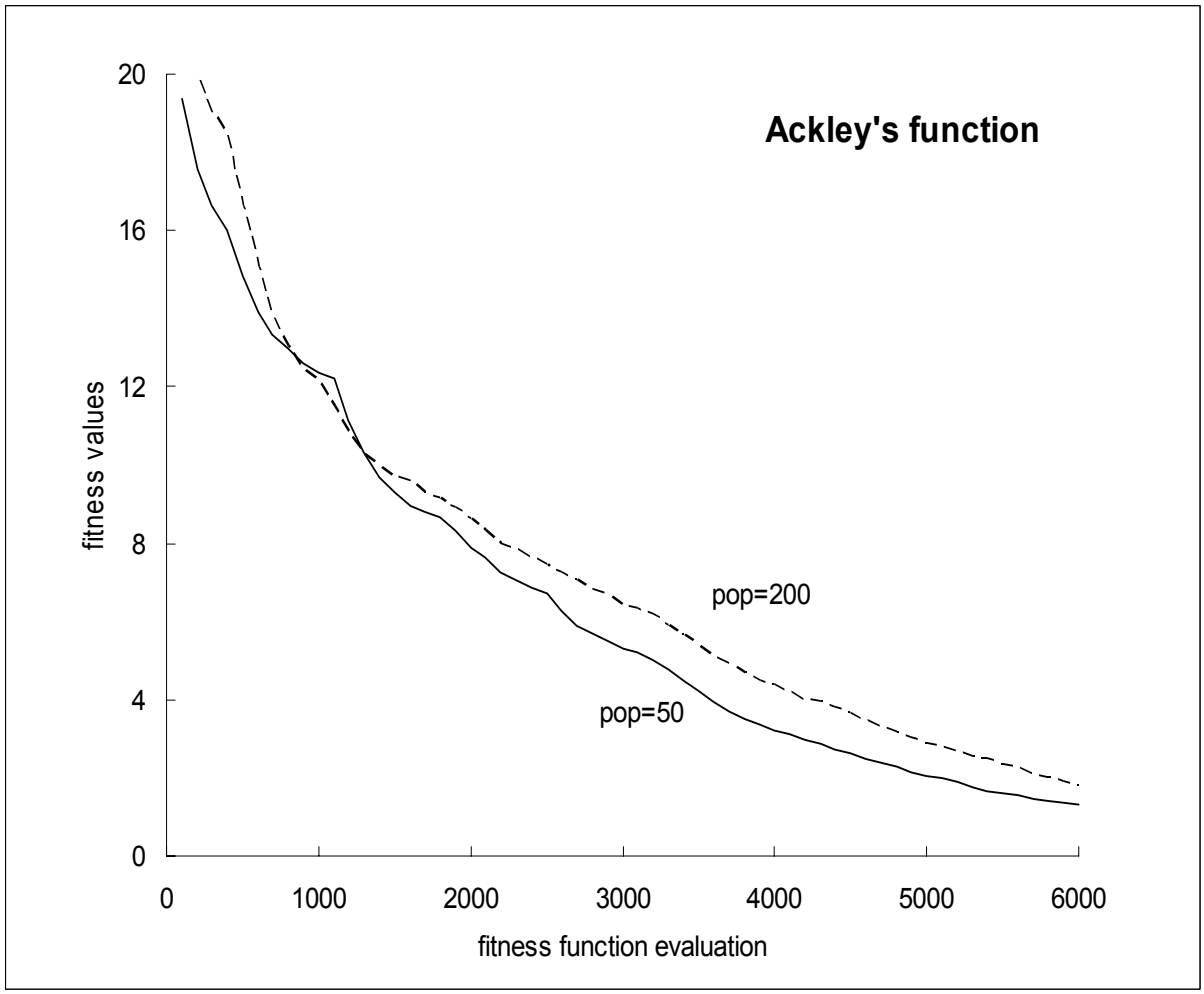

Fig. 5.1 ASAGA performance with different population sizes tested by a 20D Ackley's function

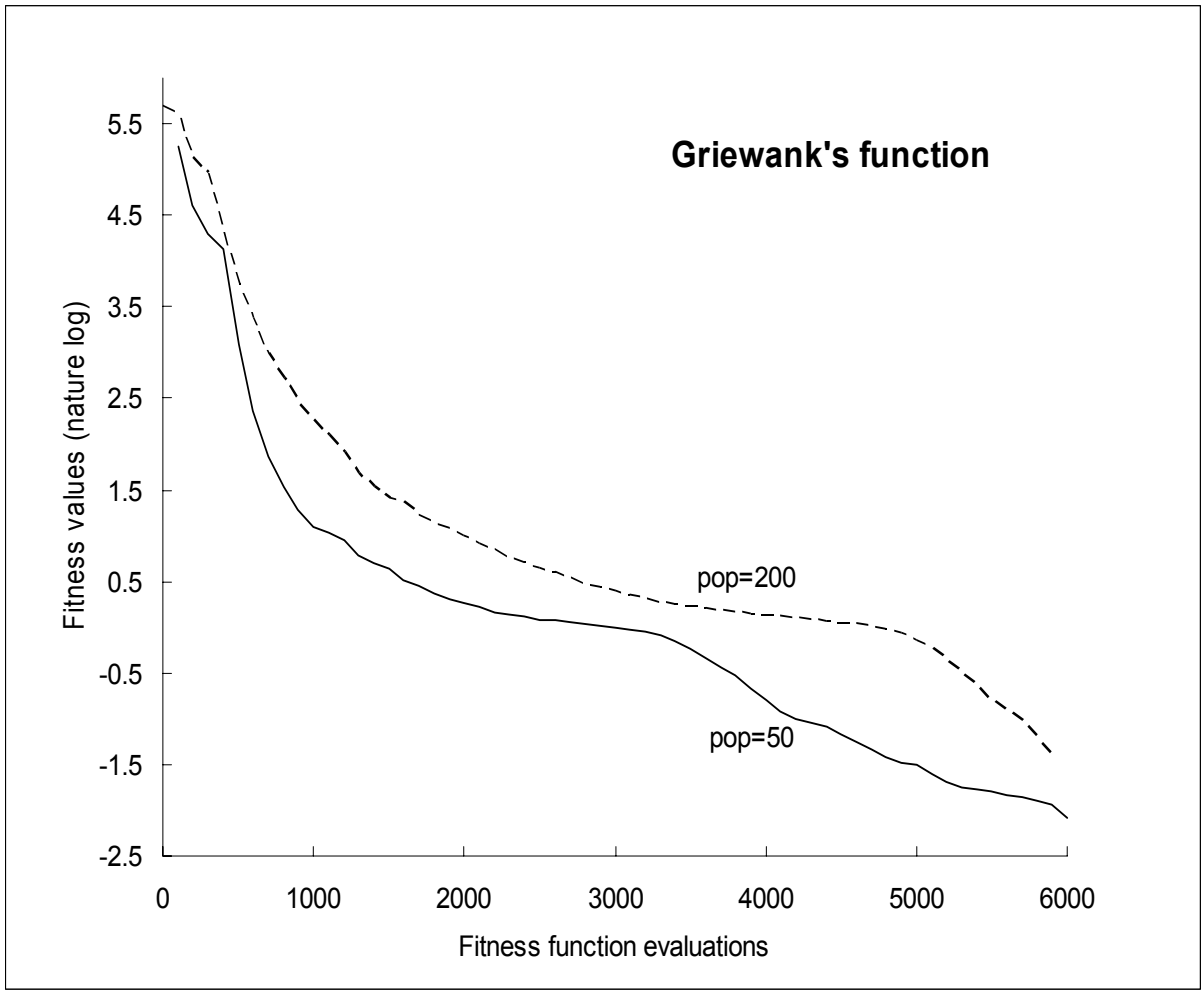

Fig. 5.2 ASAGA performance with different population sizes tested by a 20D Griewank's function

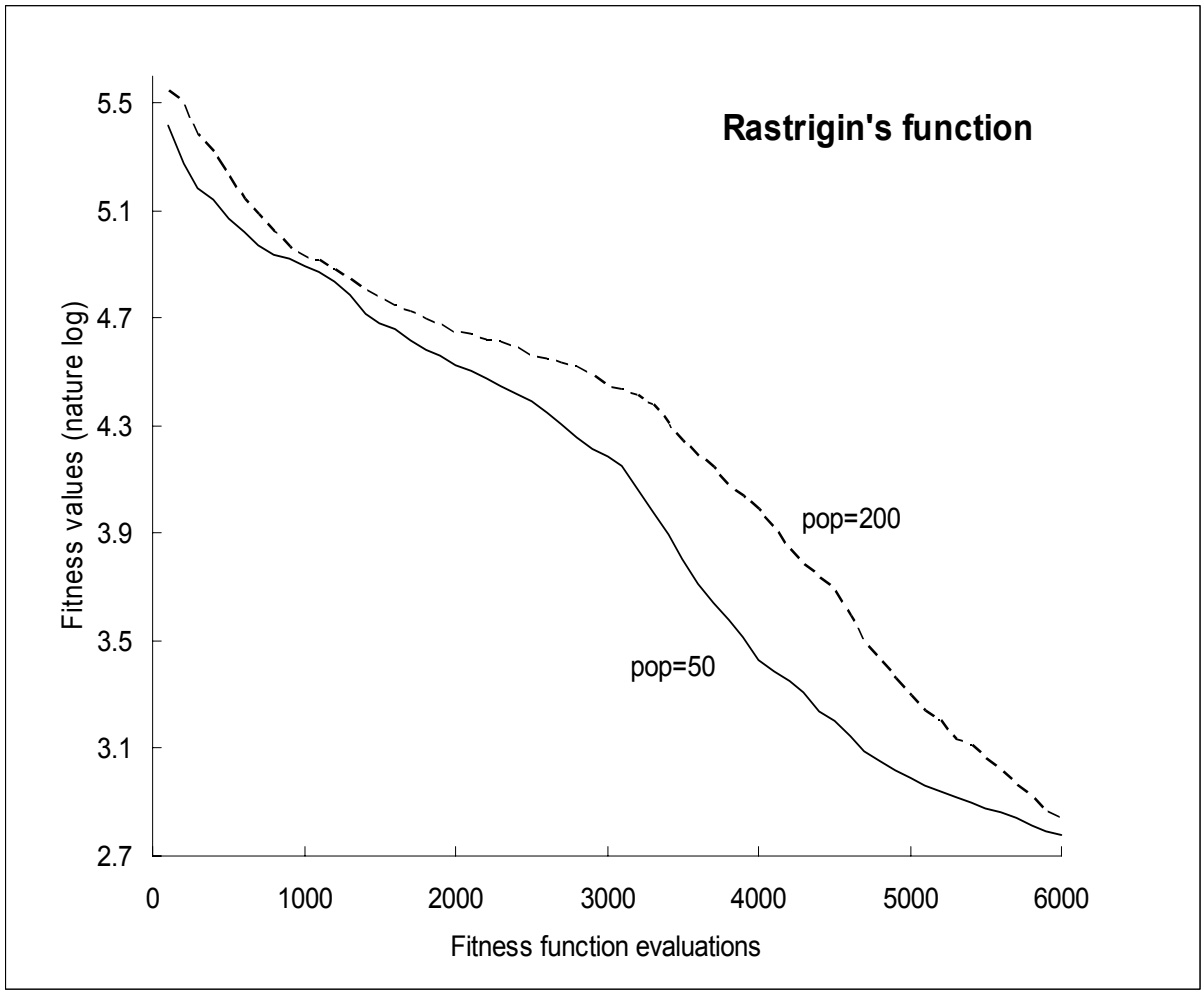

Fig. 5.3 ASAGA performance with different population sizes tested by a 20D Rastrigin's function

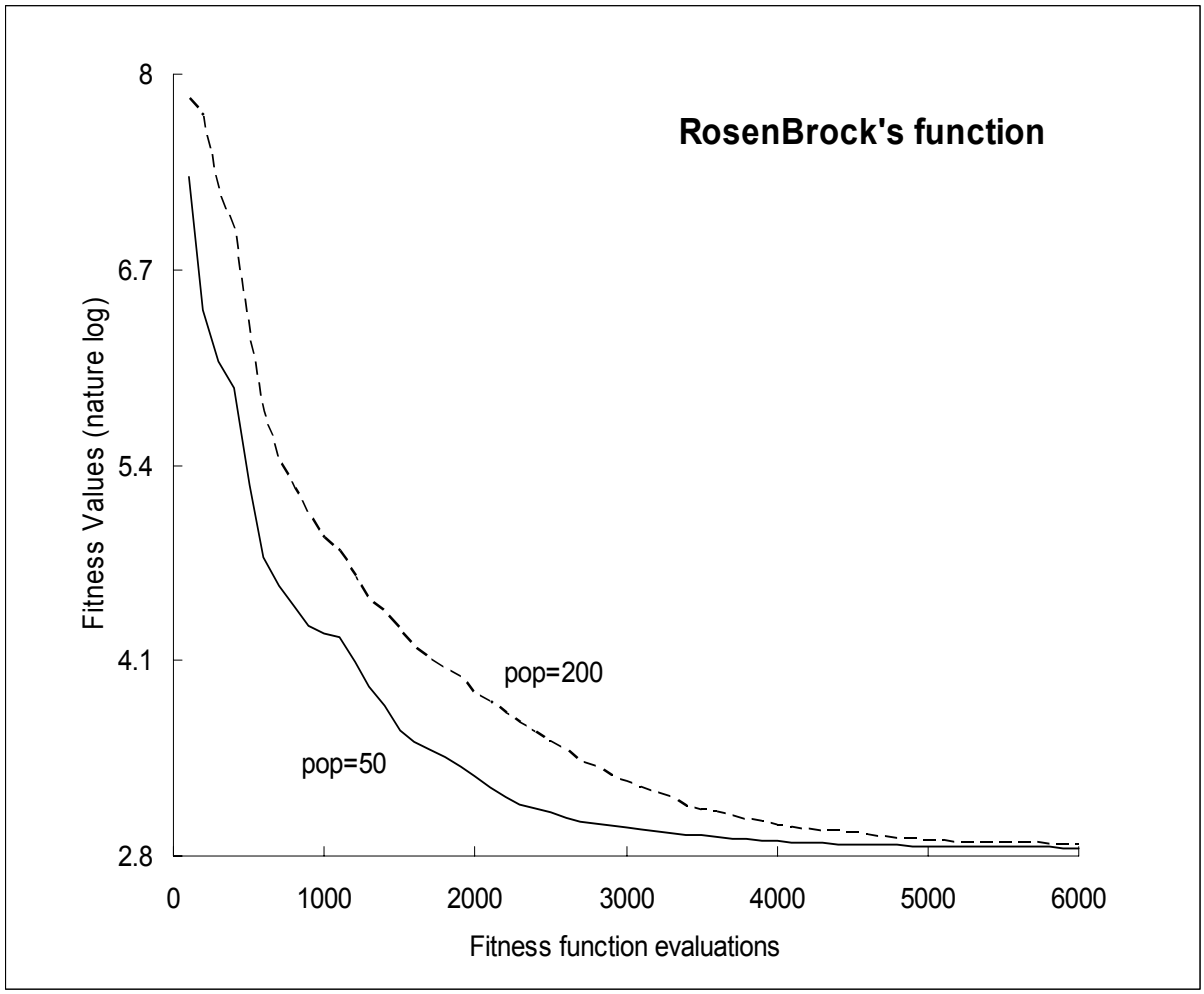

Fig. 5.4 ASAGA performance with different population sizes tested by a 20D Rosenbrock's function

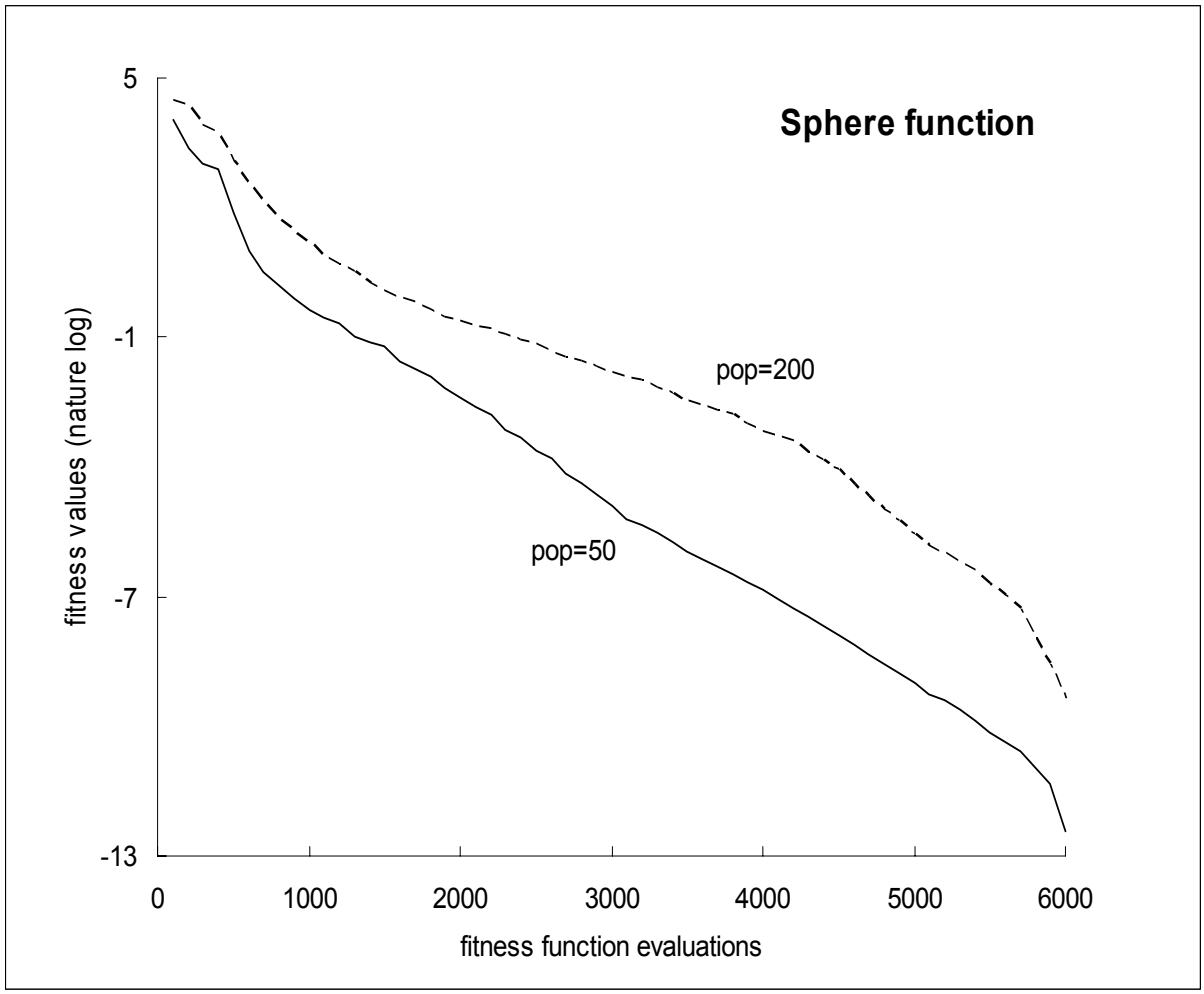

Fig. 5.5 ASAGA performance with different population sizes tested by a 20D Sphere's function

Figures 5.1 through 5.5 show ASAGA's performance with a population size of 50 and 200 in five benchmark domains, which are all minimization problems having global optima at zero. ASAGA with a population size of 50 outperforms a population size of 200 in all problems and in almost all optimization stages, suggesting that a larger population size in these benchmark domains damages performance. In general, a larger GA population size results in higher population diversity making it less likely to become stuck in local optima. However, large

population sizes can lead to slower convergence. In the unconstrained benchmark domains selected, the search space is usually continuous and not very complex, and the optimization task is relatively easy, so using a large population size has some benefits, but negatively impacts the convergence speed. However, it is not the case that an even smaller population size will get better results. A population size of 20 was tested further; mixed results indicated that it does not perform better than a population size of 50.

GA population size is usually set according to the dimension of the optimization problem. The five benchmark problems used in these experiments all have 20 dimensions, so it is safe to set the default ASAGA population size to 2.5 times the problem's dimension (50) for unconstrained problems. However, the user can easily change the population size in ASAGA if the problem complexity is known beforehand.

Following the same process, ASAGA was tested with different population sizes on constrained benchmark domains. Figure 5.6 shows ASAGA's performance when tested on a 5 dimension constrained function using population size of 20 and 50, which are 4 times the problem's dimension and 10 times problem's dimension, respectively. The results of this experiment still suggest that the smaller population size is better. Table 5.1 shows ASAGA's performance when tested on a 13 dimension constrained function using population size of 52 and 130. Only the first 10 runs are shown. ASAGA with a population size of 52 was unable to find any feasible points during all 10 runs, whereas ASAGA with a population size of 130 found feasible points in every run. This suggests that a larger population size is much better in Sandgren's function number 21, since this function has very complex feasible regions. In general, constrained optimization problems are more difficult and have more complex and discontinuous

search spaces. It is worthwhile to use a larger population size even though we might face performance loss when we deal with simple problems like the one shown in Figure 5.6. 10 times the problem's dimension was determined to be the default setting for ASAGA when dealing with constrained problems, which is 4 times larger than the setting for unconstrained problems.

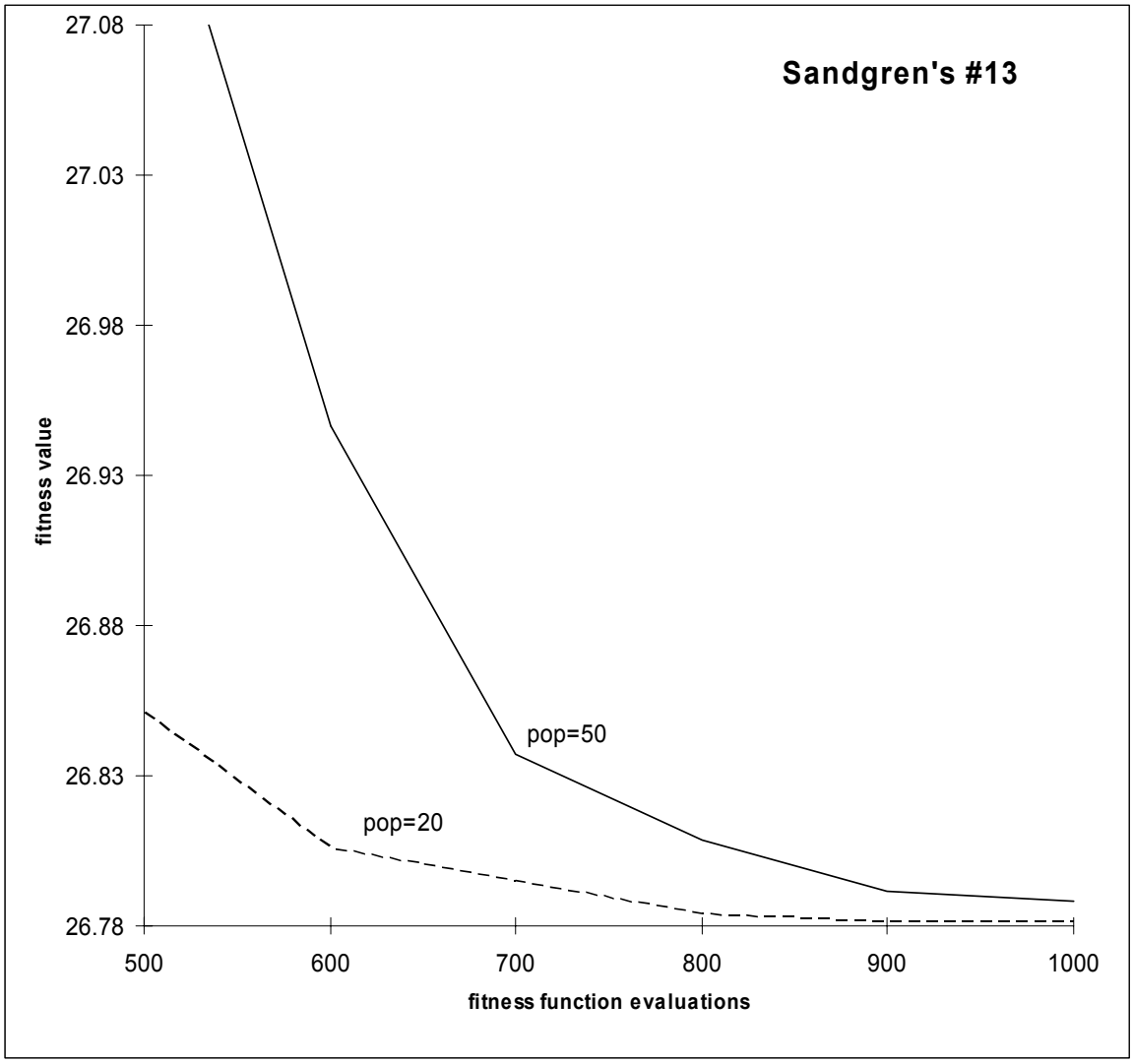

Fig. 5.6 ASAGA performance with different population sizes tested by 5D Sandgren's number 13 function

16000 Fitness Evaluations	Population size 52		Population size 130	
	Violation	Fitness	Violation	Fitness
Run 1	0.2829		0	120.48
Run 2	0.2910		0	160.12
Run 3	0.2893		0	107.37
Run 4	0.2898		0	106.31
Run 5	0.1931		0	134.46
Run 6	0.2914		0	108.97
Run 7	0.2306		0	161.02
Run 8	0.2916		0	117.35
Run 9	0.2917		0	148.08
Run 10	0.2410		0	137.19

Table 5.1 ASAGA performance with different population sizes tested by 13D Sandgren's number 21 function having global optima 97.5

SVM model parameter tuning:

Another parameter of ASAGA which requires tuning is contained within the SVM surrogate which ASAGA uses as its final global model. Different parameter settings of the SVM model may yield different performance levels. ASAGA uses the SVM^{light} package to implement the

SVM model. The SVM^{light} package itself has a strategy to adjust its parameters. The most important parameter is C, from formula (12) and (13). The SVM^{light} package adjusts its value according to the sample size, as well as how close the sample points are to each other. The SVM^{light} package begins with a very small C value. In general, larger C values will result in better fitting in local regions, but also cause the model to run more slowly. Figure 5.7 and 5.8 show ASAGA run using a large SVM C value equal to 1. The larger C value ASAGA performs better in Ackley's domain, but worse in Rastrigin's domain. Considering that the larger C version needs much more time for the SVM model fitting, ASAGA keeps the default C setting from the original SVM^{light} package, which is a relatively small C value.

Surrogate model-accuracy-only strategy vs. ASAGA:

In this section a model-accuracy-only strategy is compared to ASAGA. The model-accuracy-only strategy GA is similar to ASAGA, except that it uses model accuracy as the only criterion for upgrading the surrogates, and does not make use of the time criterion. When the correlation coefficient of the objective fitness function approximation falls below 0.9, the system switches to the next approximation model in the upgrade path. It should be able to achieve better fitting, since it does not consider the fitting time limit. Figures 5.9 through 5.13 show the two methods working in five unconstrained benchmark domains that are introduced in the previous section. These figures show that ASAGA still performs better in Ackley's and Rastrigin's functions. The two methods perform equally in Rosenbrock's function domains. The fixed surrogate selection GA outperforms ASAGA in Griewank's and Sphere's function domain. Note that the Ackley's and Rastrigin's domains are the most difficult domains among these five

benchmarks. Also note that ASAGA runs with much less CPU time. These results show that it is worth adding the adaptive surrogate selection to ASAGA. Considering both time and model accuracy in ASAGA can save optimization time and still yield good results.

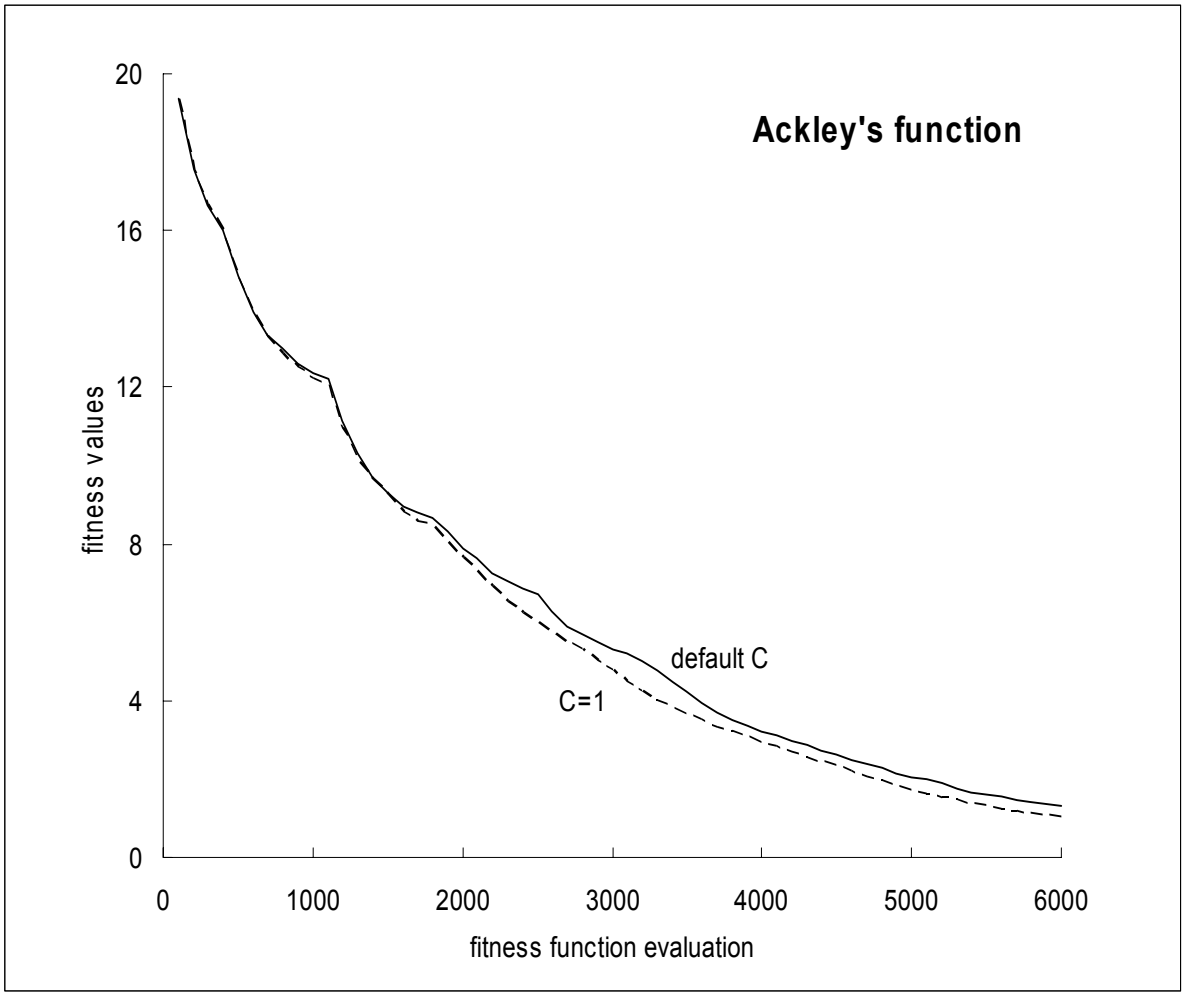

Fig. 5.7 ASAGA performance with different SVM model parameter C tested by a 20D Ackley's function

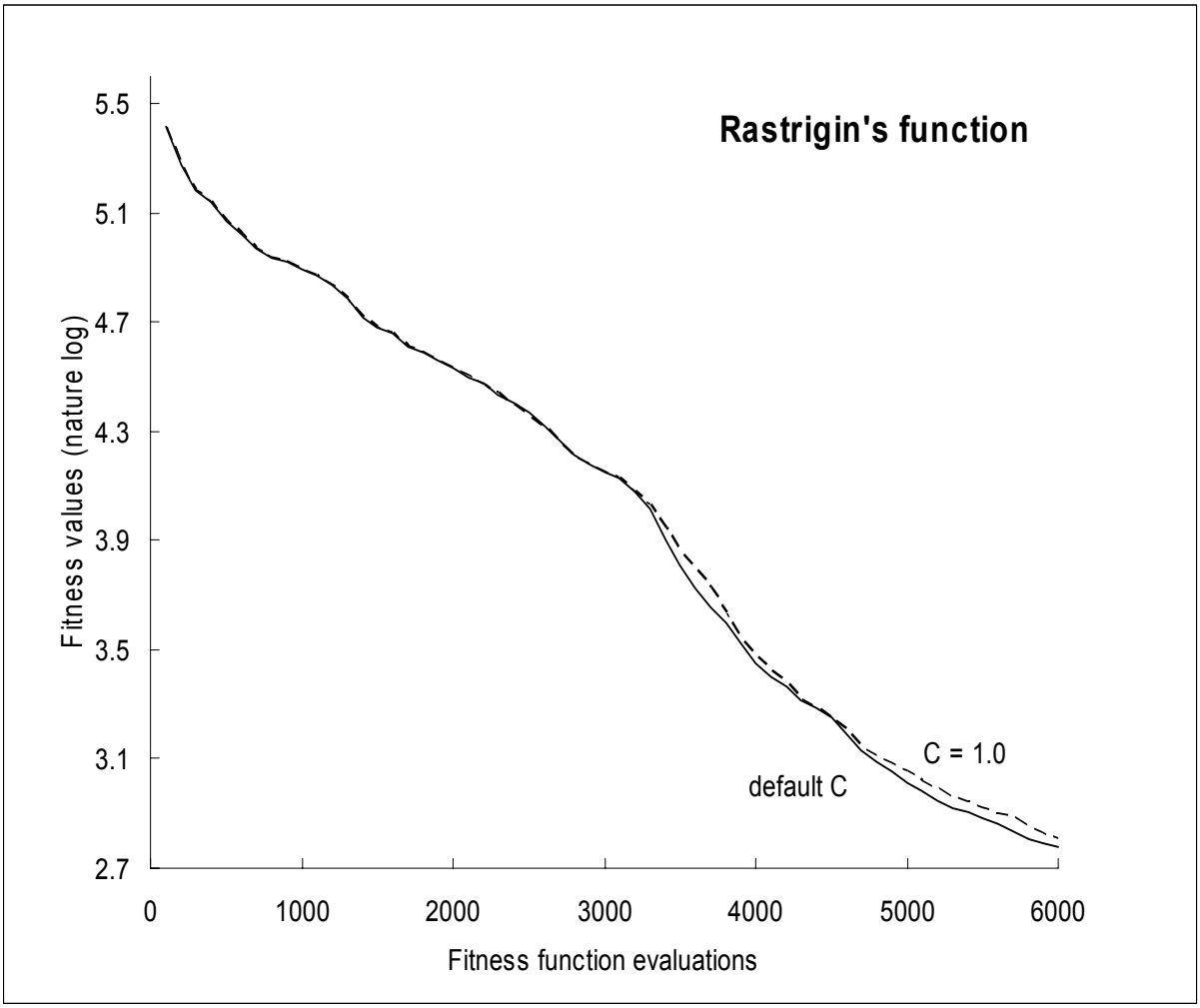

Fig. 5.8 ASAGA performance with different SVM model parameter C tested by a 20D Rastrigin's function

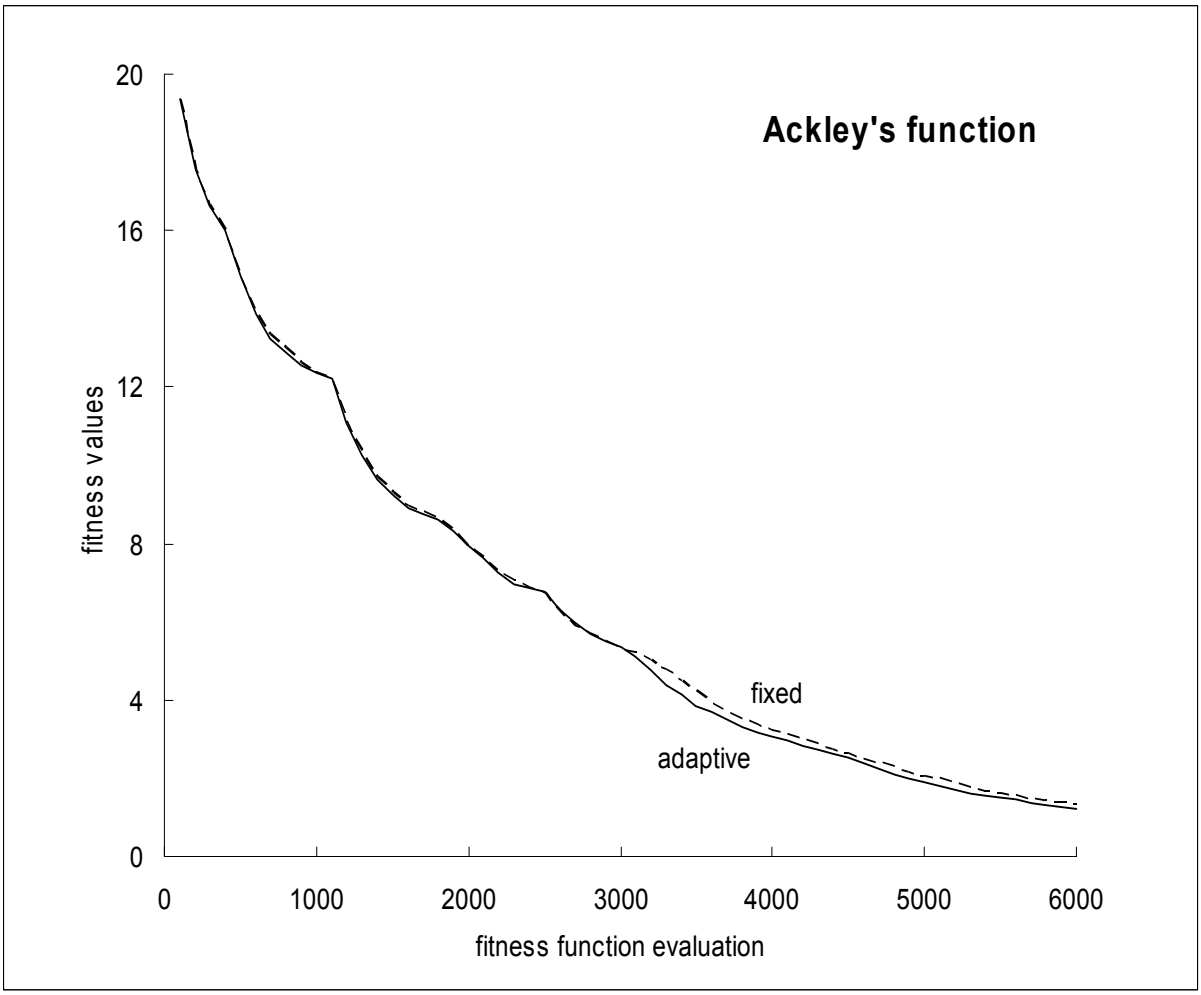

Fig. 5.9 Performance comparison between a model-accuracy-only strategy and an adaptive surrogate selection GA (ASAGA) using a 20D Ackley's function

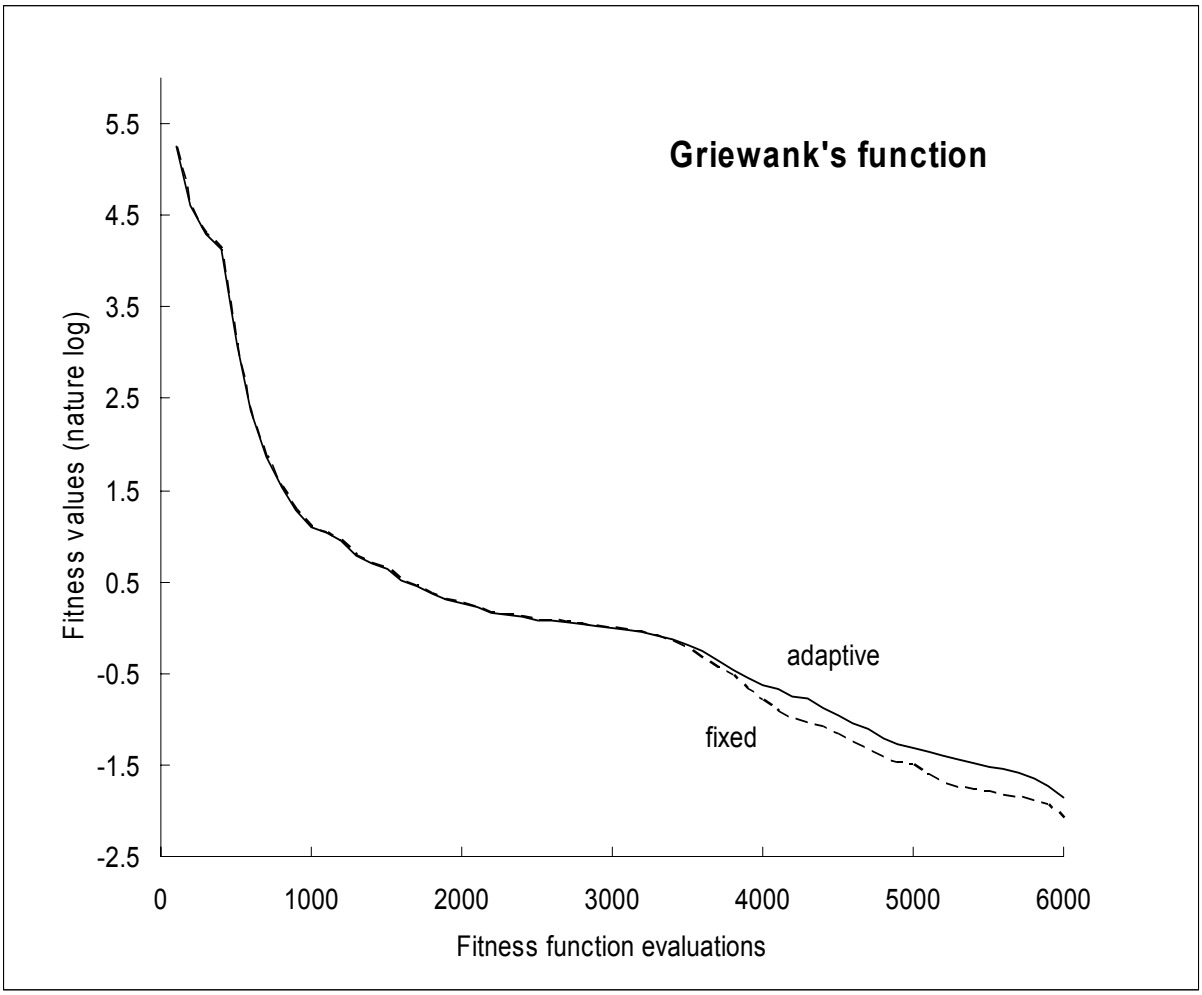

Fig. 5.10 Performance comparison between a model-accuracy-only strategy and an adaptive surrogate selection GA (ASAGA) using a 20D Griewank's function

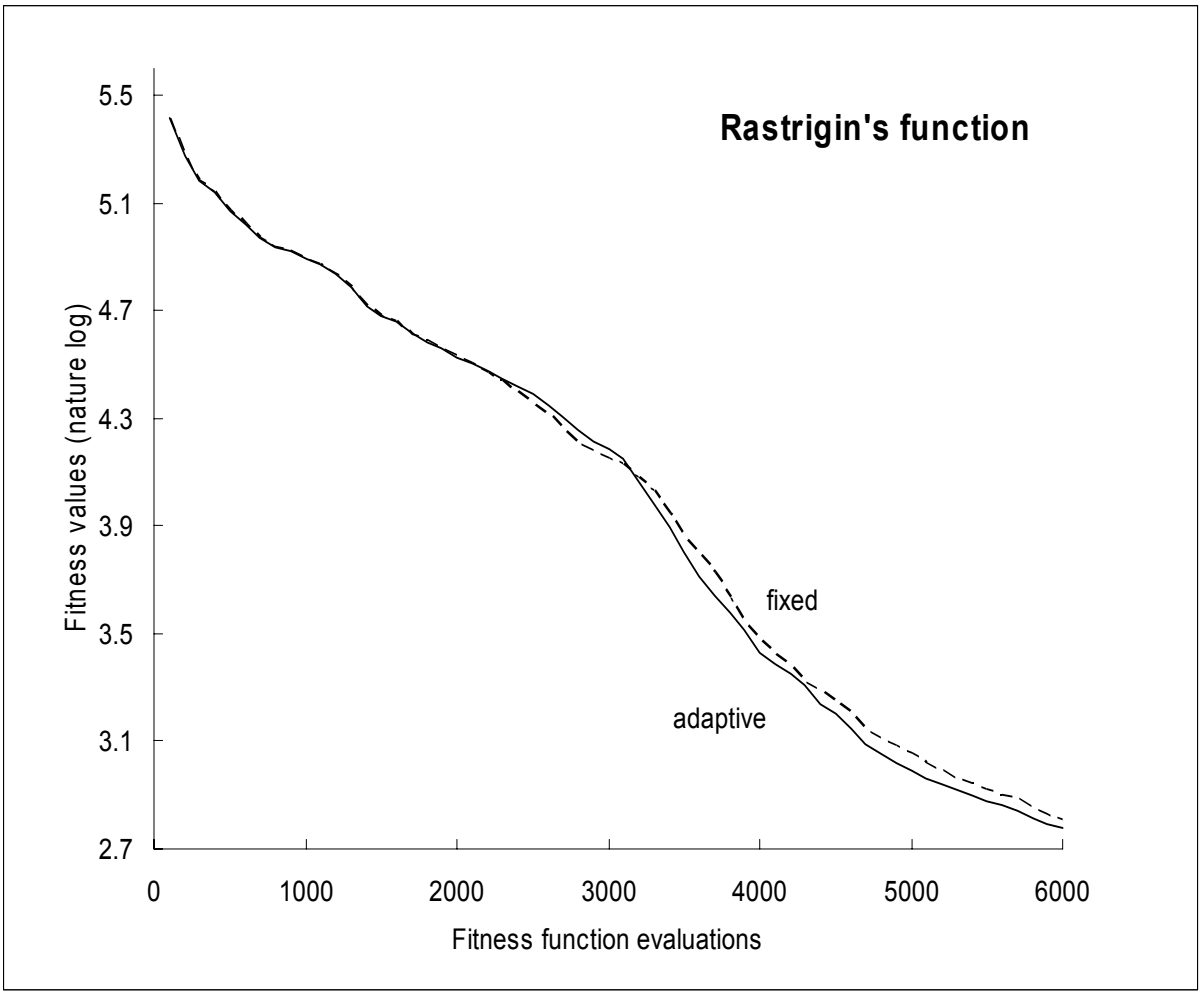

Fig. 5.11 Performance comparison between a model-accuracy-only strategy and an adaptive surrogate selection GA (ASAGA) using a 20D Rastrigin's function

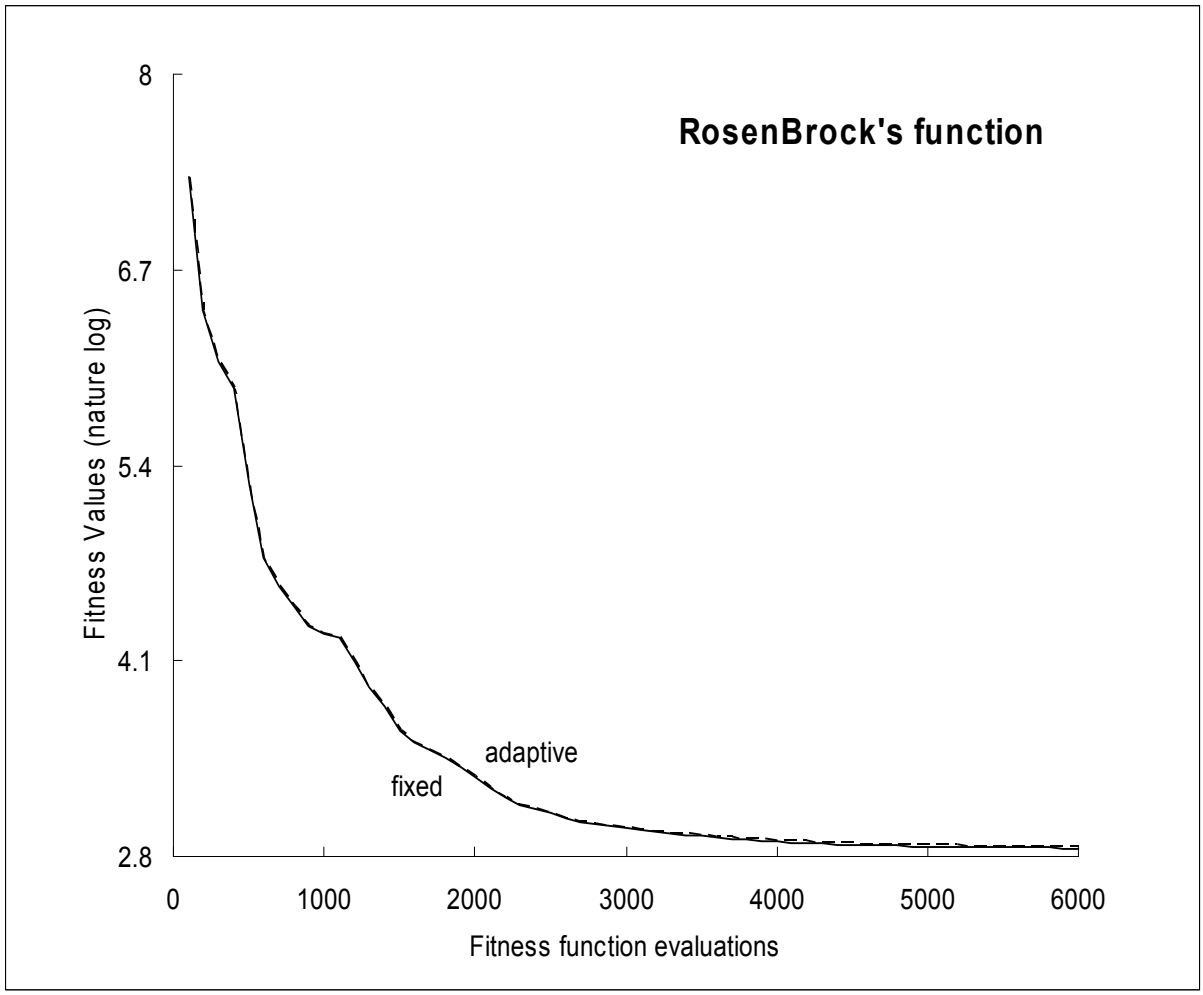

Fig. 5.12 Performance comparison between a model-accuracy-only strategy and an adaptive surrogate selection GA (ASAGA) using a 20D Rosenbrock's function

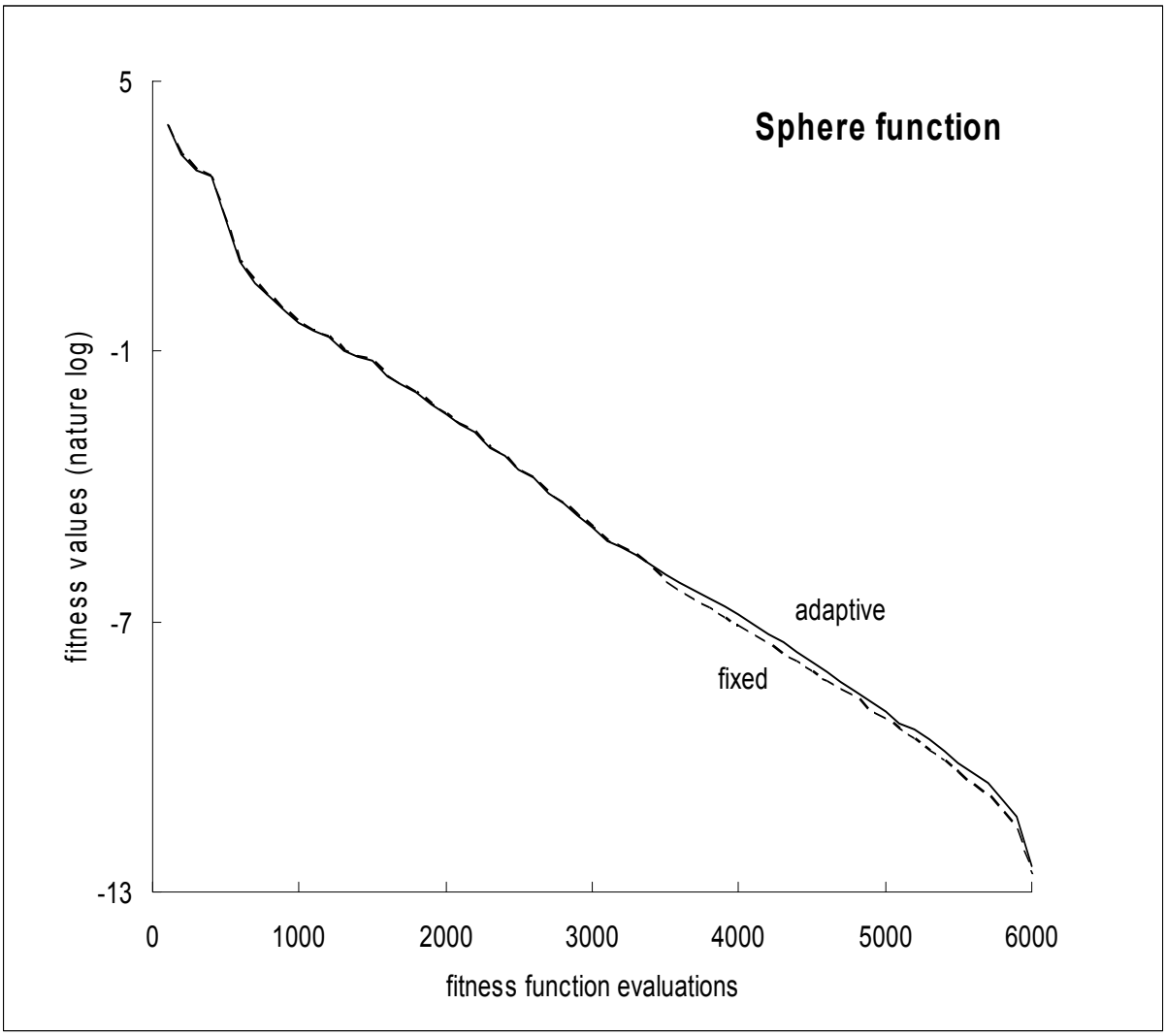

Fig. 5.13 Performance comparison between a model-accuracy-only strategy and an adaptive surrogate selection GA (ASAGA) using a 20D Sphere's function

No Surrogate Selection vs. Adaptive Surrogate Selection:

In this section ASAGA is compared to GADO-R, which stands for GADO with Reduced models. GADO-R is a fixed quadratic polynomial model assisted version of GADO. Figures 5.14 through 5.18 show the performance of ASAGA and GADO-R. It is clear that ASAGA

outperforms or matches the performance of GADO-R in all five benchmark domains. Because ASAGA considers the fitting time in the optimization process, it uses less time than GADO-R, even though ASAGA makes use of the much more complex SVM model. Note that ASAGA performs worse than GADO-R in the early stages of the optimization in some test domains. The reason for this is that ASAGA uses simple models at the beginning of the optimization, and this may hurt the performance in the early stages.

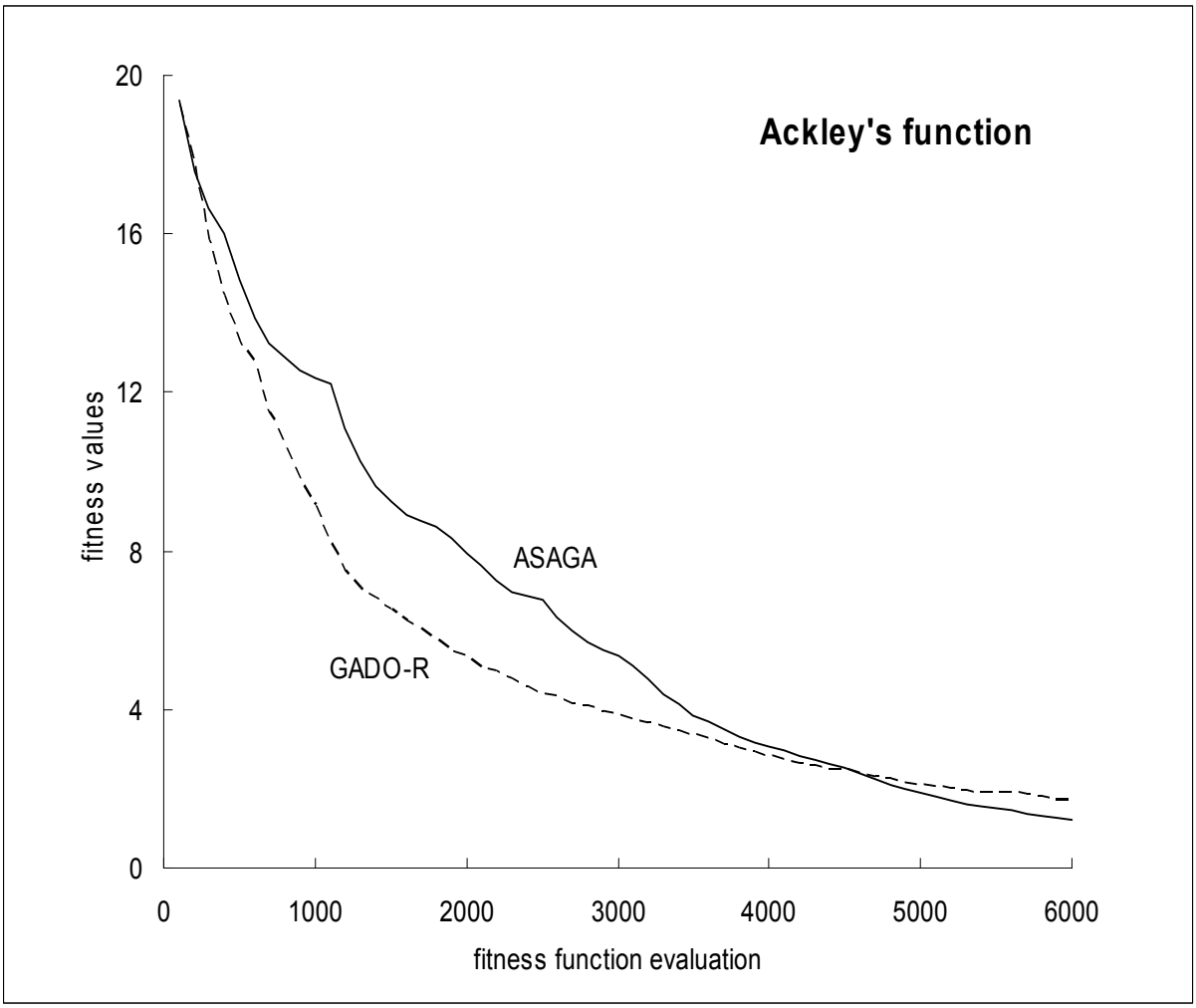

Fig. 5.14 Performance comparison between GADO-R and ASAGA using a 20D Ackley's function

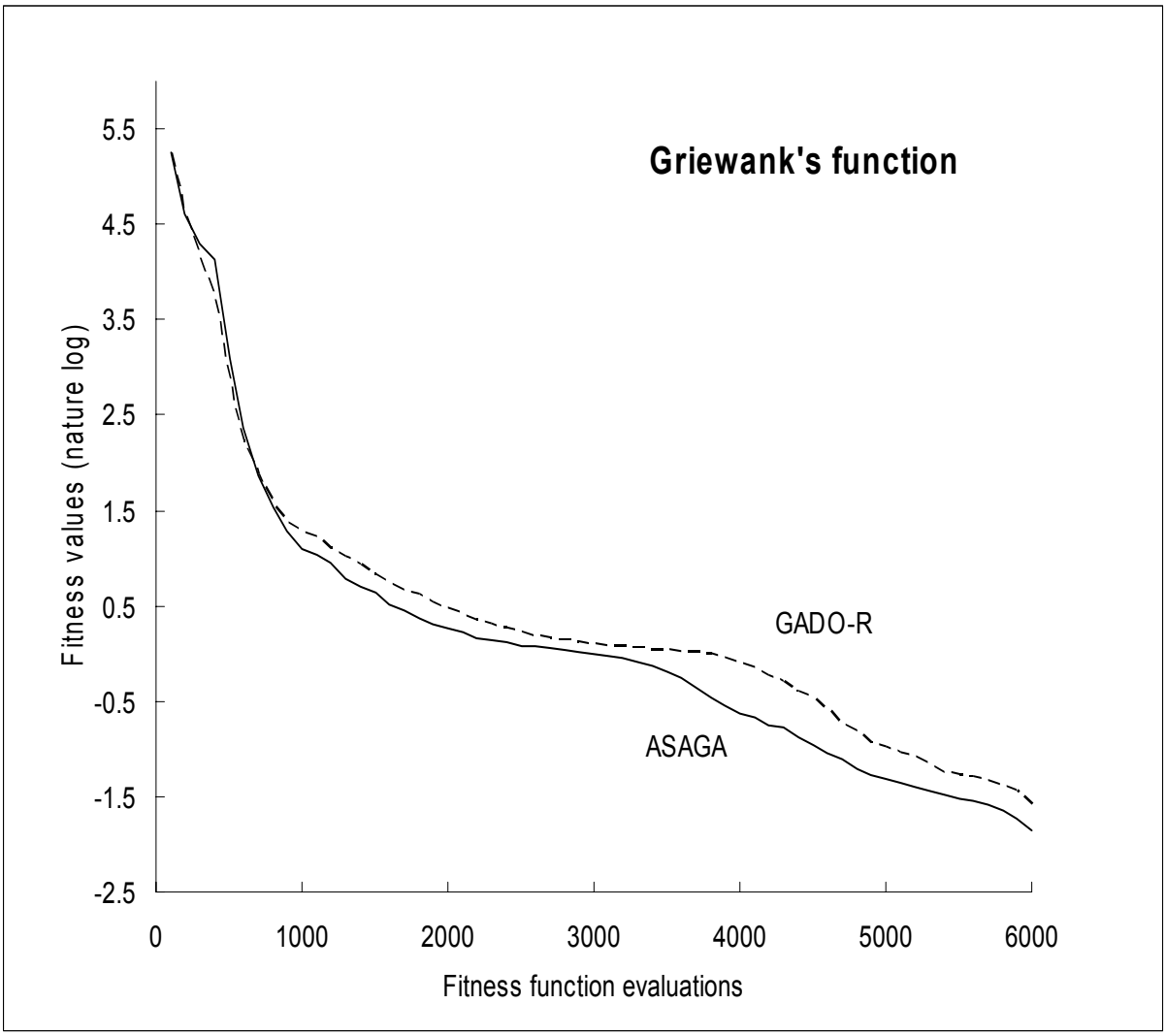

Fig. 5.15 Performance comparison between GADO-R and ASAGA using a 20D Griewank's function

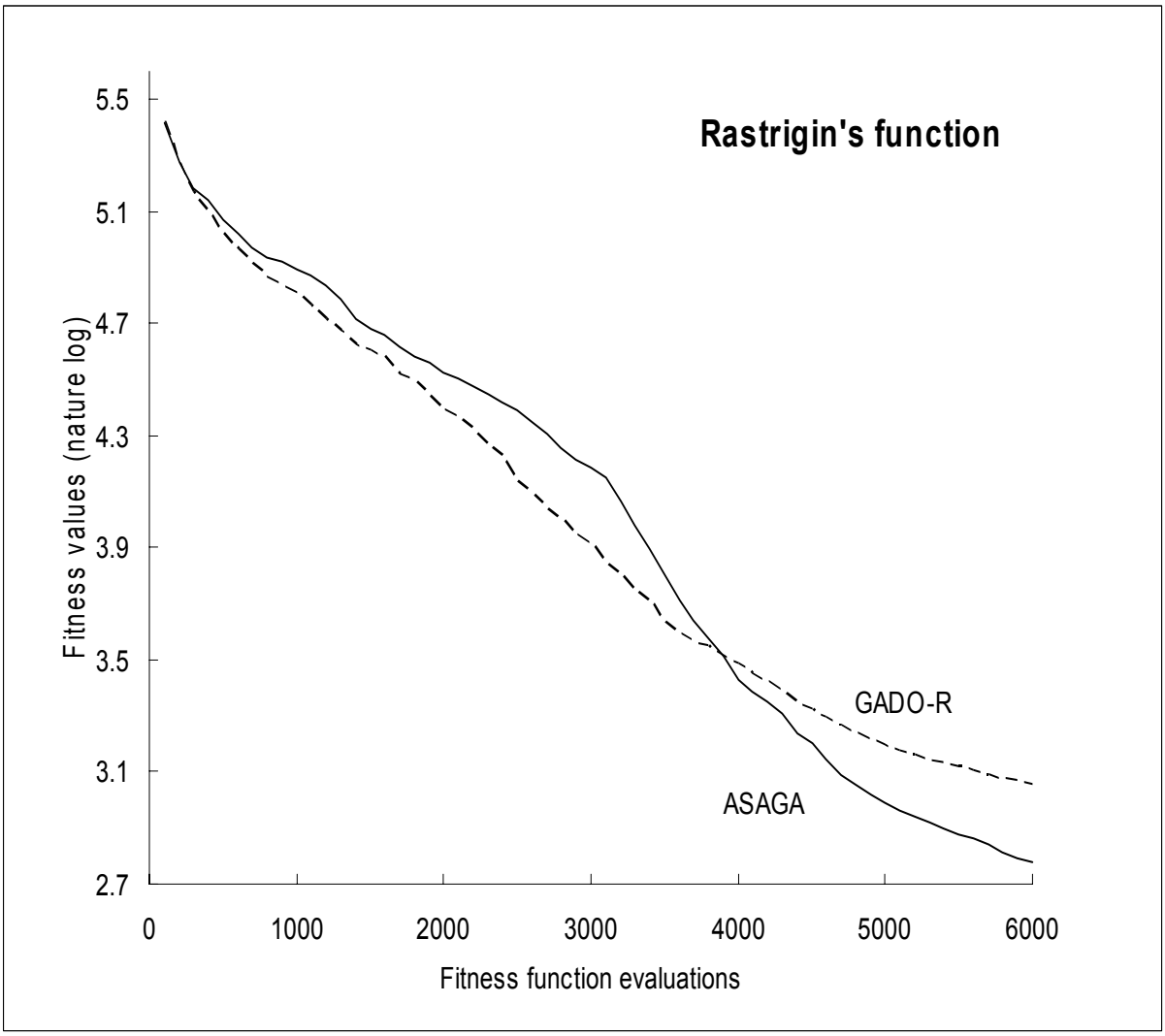

Fig. 5.16 Performance comparison between GADO-R and ASAGA using a 20D Rastrigin's function

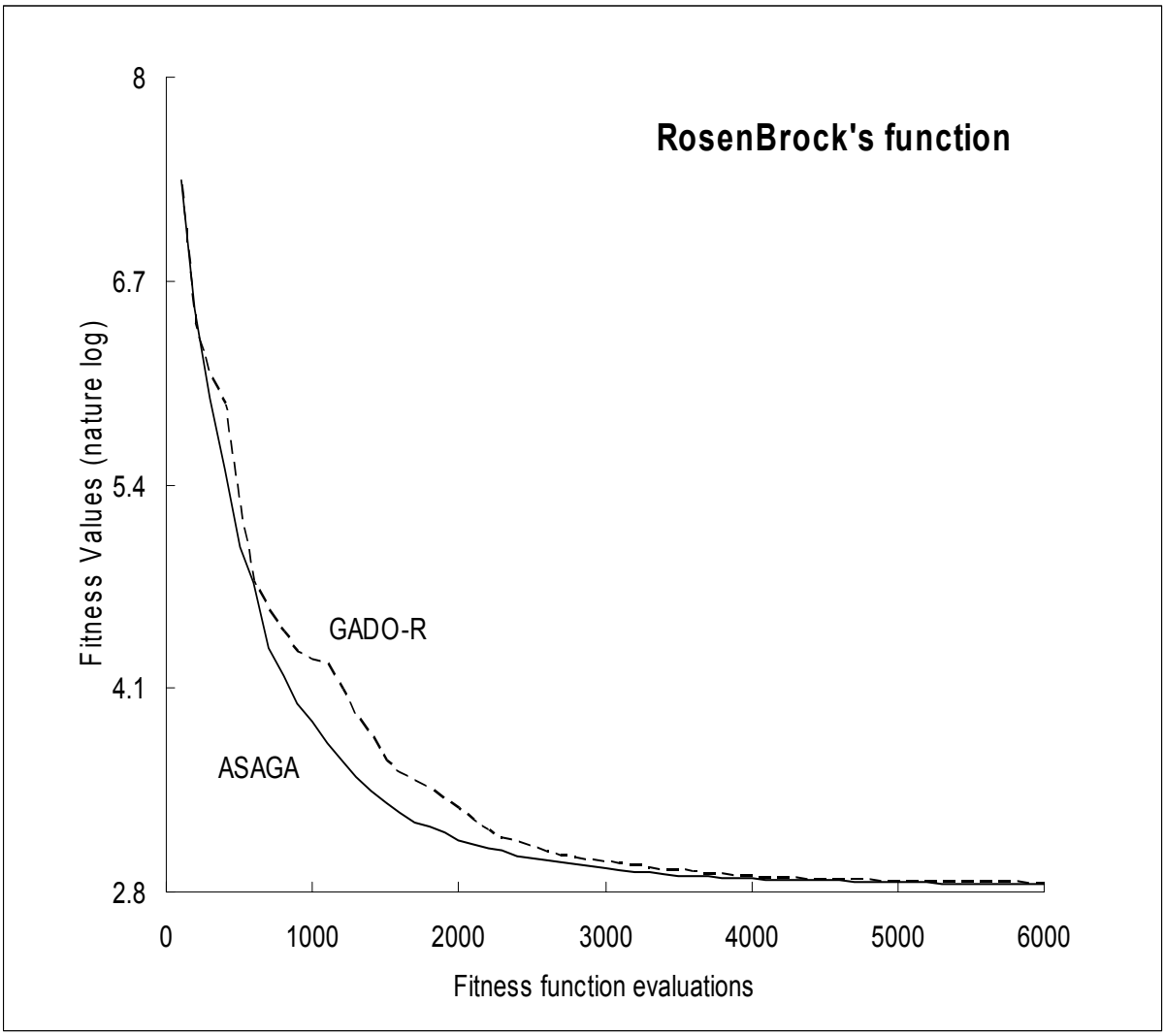

Fig. 5.17 Performance comparison between GADO-R and ASAGA using a 20D Rosenbrock's function

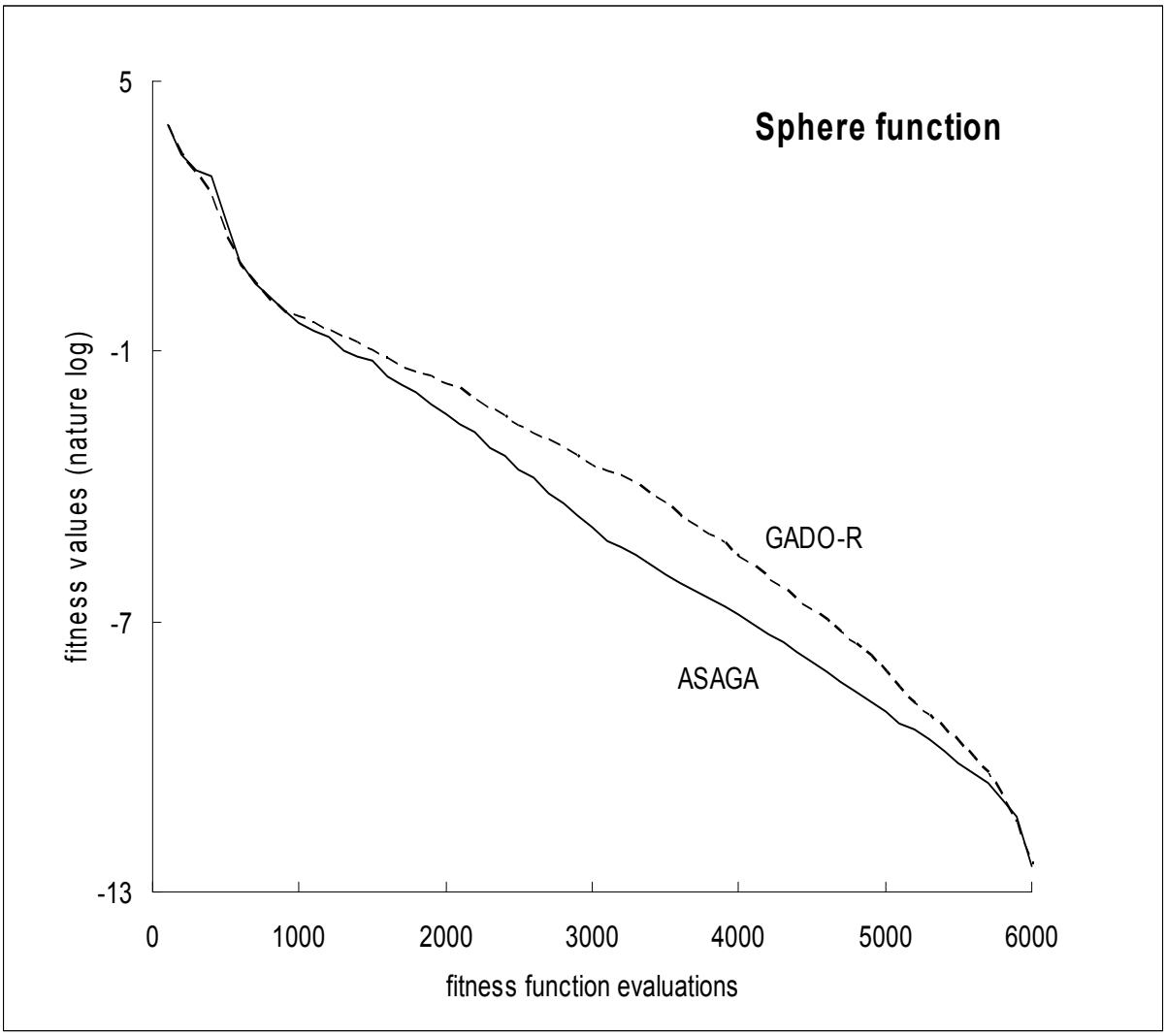

Fig. 5.18 Performance comparison between GADO-R and ASAGA using a 20D Sphere's function

Chapter 6

Conclusion

Using fitness approximation methods to assist GAs and other types of Evolutionary Algorithms has 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 can be used.

So far many researchers use only one type of approximation model, as in [25]. Some researchers use multiple models for different levels of approximation, but the approximate model itself is still fixed [13, 29, 46]. 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 fitness approximation.

ASAGA makes use of global and local approximate models, using a clustering technique with one local model for each cluster. The global model can adaptively evolve from a quadratic polynomial model to a Support Vector Machine (SVM) model. The model evolution depends on time spent and model accuracy. The local models follow similar rules, except that they can evolve from a simple average of the fitness of all individuals in a cluster to a final quadratic model (not SVM), since each local model is based on a much smaller sample set. The number of clusters is also adaptively controlled according to the type of model currently being used, model accuracy, and the size of the cluster.

In ASAGA, the frequency of model usage is also adaptively adjusted according to a formula that takes into account the percentage of runtime spent on fitness approximation, as well as model accuracy. A formula that takes into account both factors to reach a trade-off point is applied in ASAGA. A stochastic penalty function method is introduced in ASAGA to further strengthen its performance when dealing with constrained problem domains.

Experiments show that ASAGA performs better than or comparable to several state-of-art surrogate-assisted GAs in benchmark domains and engineering design. Performance analysis was used to tune the most influential parameters in ASAGA.

There are still several open questions regarding this method that could be addressed in future work. One question is whether there is a better way to control model updates. Another question would be whether model updates should happen in the reverse direction as well, i.e. the more sophisticated model may be replaced by a simpler model at some point, perhaps when many more clusters are introduced. Finally, adaptive surrogate assistance can be incorporated into other EAs such as evolution strategies and evolution programming.

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] 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.

[3] Schwefel H-P Evolution and Optimum Seeking. Wiley, 1995.

[4] D. Chafekar, L. Shi, K. Rasheed, and J. Xuan. Multi-objective GA optimization using reduced models. *IEEE Trans. on Systems, Man, and Cybernetics: Part C*, 9(2):261-265, 2005.

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

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

[7] R. Smith, B. Dike, and S. Stegmann. Fitness inheritance in genetic algorithms. In *Proceedings of ACM Symposiums on Applied Computing*, pages 345-350, ACM, 1995.

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

[9] M. Pelikan and K. Sastry. Fitness inheritance in the Bayesian optimization algorithms. In *Genetic and Evolutionary Computation Conference*, pages 48--59, Springer, 2004.

[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, IEEE, 2001.

[12] L. Elliott and et al. An informed operator based genetic algorithm for tuning the reaction rate parameters of chemical kenetics mechanisms. In *Genetic and Evolutionary Computation Conference*, pages 945--956, 2004.

[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, Springer, 2004.

[14] Y. S. Ong, A.J. Keane, and P.B. Nair. Surrogate-Assisted Coevolutionary Search. In *9th International Conference on Neural Information Processing, Special Session on Trends in Global Optimization*, Singapore, pages 2195--2199, 2002.

[15] K. Rasheed. An incremental-approximate-clustering approach for developing dynamic reduced models for design optimization. In *Proceedings of the Congress on Evolutionary Computation (CEC'2002)*, pp. 986—993, 2002.

[16] M Pelikan, K Sastry, DE Goldberg. Multiobjective hBOA, clustering, and scalability. *Proceedings of the 2005 conference on Genetic and evolutionary computation,* pages: 663-670. Washington DC, USA. 2005.

[17] H. Takagi. Interactive evolutionary computation. Fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275 – 1296, 2001.

[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] M Gibbs, DJC MacKay. Efficient Implementation of Gaussian Processes. Unpublished manuscript. Cavendish Laboratory, Cambridge, UK, 1997.

[20] Gaussian Processes for regression. CKI Williams, CE Rasmussen. In *Advances in Neural Information Processing Systems 8 eds*. D. S. Touretzky, M. C. Mozer, M. E. Hasselmo, MIT Press, 1996.

[21] M. Emmerich, A. Giotis, M. Özdenir, T. Bäck, and K. Giannakoglou. Metamodel-assisted evolution strategies. In *Parallel Problem Solving from Nature*, number 2439 of Lecture Notes in Computer Science, pages 371-380, Springer, 2002.

[22] M.A. El-Beltagy and A.J. Keane. Evolutionary optimization for computationally expensive problems using Gaussian processes. In *Proceedings of International Conference on Artificial Intelligence*, pages 708--714, CSREA, 2001.

[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] H. Ulmer, F. Streichert, and A. Zell. Evolution startegies assisted by Gaussian processes with improved pre-selection criterion. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 692--699, 2003.

[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] H. Ulmer, F. Streicher, and A. Zell. Model-assisted steady-state evolution strategies. In *Proceedings of Genetic and Evolutionary Computation Conference*, LNCS 2723, pages 610-621, 2003.

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

[28] L. Graening, Y. Jin, and B. Sendhoff. Efficient evolutionary optimization using individual-based evolution control and neural networks: A comparative study. In *European Symposium on Artificial Neural Networks*, pages 273-278, 2005.

[29] Y.-S. Hong, H.Lee, and M.-J. Tahk. Acceleration of the convergence speed of evolutionary algorithms using multi-layer neural networks. *Engineering Optimization*, 35(1):91-102, 2003.

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

[31] 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.

[32] M. Papadrakakis, N. Lagaros, and Y. Tsompanakis. Optimization of large-scale 3D trusses using Evolution Strategies and Neural Networks. *Int. J. Space Structures*, 14(3):211-223, 1999.

[33] G. Schneider. Neural networks are useful tools for drug design. *Neural Networks*, 13:15-16, 2000.

[34] W. Shyy, P. K. Tucker, and R. Vaidyanathan. Response surface and neural network techniques for rocket engine injector optimization. Technical report 99-2455, AIAA, 1999.

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

[36] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Patter Analysis*. Cambridge Press, 2004.

[37] 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.

[38] 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.

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

[40] 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.

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

[42] 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.

[43] J. Branke and C. Schmidt. Fast convergence by means of fitness estimation. *Soft Computing Journal*, 9(1):13-20, 2005.

[44] 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.

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

[46] 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.

[47] M. Sefrioui and J. Periaux. A hierarchical genetic algorithm using multiple models for optimization. In *Parallel Problem Solving from Nature*, volume 1917 of *Lecture Notes in Computer Science*, pages 879-888, Springer, 2000.

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

[49] Khaled Rasheed. Guided Crossover: A New Operator for Genetic Algorithm Based Optimization". *Congress on Evolutionary Computation (CEC'99)*, 1999.

[50] D. Hidovic and J.E. Rowe. Validating a model of colon colouration using an evolution strategy with adaptive approximations. In *Genetic and Evolutionary Computation Conference*, pages 1005--1017, 2004.

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

[52] J. Ziegler and W. Banzhaf. Decreasing the number of evaluations in evolutionary algorithms by using a meta-model of the fitness function. In C. Ryan, T. Soule, M. Keijzer, E. Tsang, R. Poli, and E. Costa, editors, *Proc. Sixth European Conf. Genetic Programming (EuroGP'03)*, volume 2610 of *Lecture Notes in Computer Science*, Berlin, pages 264--275, Springer, 2003.

[53] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[54] 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.

[55] J. A. Biles. GenJam. A genetic algorithm for generating jazz solos. In *Proceedings of International Computer Music Conference*, pages 131-137, 1994.

[56] M. Sefrioui and J. Periaux. A hierarchical genetic algorithm using multiple models for optimization. In *Parallel Problem Solving from Nature*, volume 1917 of *Lecture Notes in Computer Science*, pages 879-888, Springer, 2000.

[57] Z Skolicki, K De Jong. The influence of migration sizes and intervals on island models. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages: 1295-1302. 2005.

[58] 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.

[59] R.G. Regis and C.A. Shoemaker. Local Function Approximation in Evolutionary Algorithms for the Optimization of Costly Functions. *IEEE Transactions on Evolutionary Computation*, 8(5):490--505, 2004.

[60] L. Shi and K. Rasheed. ASAGA: An Adaptive Surrogate-Assisted Genetic Algorithm. *Genetic and Evolutionary Computation Conference*, pages: 1049-1056, Atlanta, USA, GECCO'2008. ACM Press, 2008.

[61] Reklaitis, G. V., Ravindran, A. and Ragsdell, K.M. Engineering Optimization Methods and Application. New York. Wiley, 1983.

[62] Deb, K. Optimization for Engineering Design: Algorithms and Examples. New Delhi. Prentice-Hall, 1995.

[63] ED Weinberger. Fourier and Taylor series on fitness landscapes. *Biological Cybernetics* 65:55, 321-330. Springer, 1991.

[64] W. Hordijk and P. F. Stadler. Amplitude Spectra of Fitness Landscapes. *J. Complex Systems*, 1:39—66, 1998.

[65] D. Lim, Y. S. Ong, Y. Jin and B. Sendhoff. A Study on Metamodeling Techniques, Ensembles, and Multi-Surrogates in Evolutionary Computation. *Genetic and Evolutionary Computation Conference*. London, UK, pp. 1288 - 1295, ACM Press, 2007.

[66] L. E. Zerpa, N.V. Queipo, S. Pintos, J.-L. Salager. An Optimization Methodology of Alkaline-surfactant-polymer Flooding Processes Using Field Scale Numerical Simulation and Multiple Surrogates, *Journal of Petroleum Science and Engineering*, 47:197-208, 2005.

[67] Z Zhou, YS Ong, MH Lim, BS Lee. Memetic Algorithm Using Multi-surrogates for Computationally Expensive Optimization Problems, *Soft Computing*, Vol. 11, No. 10, pp. 957-971, 2007.

[68] T. Goel, R. T. Haftka, W. Shyy, N. V. Queipo. Ensemble of Surrogates? *Structural and Multidisciplinary Optimization* 33: 199-216, 2007.

[69] D. Lundström, S. Staffan, W. Shyy. Hydraulic Turbine Diffuser Shape Optimization by Multiple Surrogate Model Approximations of Pareto Fronts. *Journal of fluids engineering*, Vol. 129, No. 9, September. p. 1228-1240, 2007.

[70] K. Sastry, C. F. Lima, D. E. Goldberg. Evaluation Relaxation Using Substructural Information and Linear Estimation, *Proceedings of the 8th annuak conference on Genetic and Evolutionary Computation Conference*, 2006.

[71]D. Powell and M. Skolnick. Using genetic algorithms in engineering design optimization with non-linear constraints, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 424- 431. Morgan Kaufmann, 1993.