1

Knowledge Incorporation in Multi-Objective Evolutionary Algorithms

Ricardo Landa-Becerra, Luis V. Santana-Quintero
and Carlos A. Coello Coello

CINVESTAV-IPN (Evolutionary Computation Group)
Departamento de Computación
Av. IPN No. 2508, Col. San Pedro Zacatenco
México, D.F. 07360, MEXICO
rlanda@computacion.cs.cinvestav.mx, lvspenny@hotmail.com
ccoello@cs.cinvestav.mx

Summary. This chapter presents a survey of techniques used to incorporate knowledge into evolutionary algorithms, with a particular emphasis on multi-objective optimization. We focus on two main groups of techniques: those that incorporate knowledge into the fitness evaluation, and those that incorporate knowledge in the initialization process and the operators of an evolutionary algorithm. Several techniques representative of each of these groups are briefly discussed, together with some examples found in the specialized literature. In the last part of the chapter, we provide some research ideas that are worth exploring in the future by researchers interested in this topic.

1.1 Introduction

In many disciplines, optimization problems have two or more objectives, which are normally in conflict with each other, and that we wish to optimize simultaneously. These problems are called "multi-objective", and their solution involves the design of different algorithms than when dealing with single-objective optimization problems. Multi-objective problems, in fact, normally give rise not to one, but to a set of solutions (called Pareto optimal set) which are all equally good.

Evolutionary algorithms have been very popular for solving multi-objective optimization problems [10, 11], mainly because of their ease of use, and their wide applicability. However, multi-objective evolutionary algorithms (MOEAs) tend to consume an important number of objective function evaluations, in order to achieve a reasonably good approximation of the Pareto front, even when dealing with benchmark problems of low dimensionality. This is a major concern when attempting to use MOEAs for real-world applications,

since we normally can afford only a fairly limited number of fitness function evaluations in such cases.

Despite these concerns, little efforts have been reported in the literature to reduce the computational cost of MOEAs, and several of them only focus on algorithmic complexity (see for example [29]), in which little else can be done because of the theoretical bounds related to nondominance checking [35]. It has been until relatively recently, that researchers have developed techniques to achieve a reduction of fitness function evaluations by exploiting knowledge acquired during the search [34]. This knowledge can, for instance, be used to adapt the recombination and mutation operators in order to sample offspring in promising areas of the search space (as done through the use of cultural algorithms [59]). Knowledge of past evaluations can also be used to build an empirical model that approximates the fitness function to optimize. This approximation can then be used to predict promising new solutions at a smaller evaluation cost than that of the original problem [34, 32].

This chapter describes some of the possible schemes by which knowledge can be incorporated into an evolutionary algorithm, with a particular emphasis on MOEAs. The taxonomy of approaches that we will cover in this chapter is shown in Figure 1.1. In this proposed taxonomy, we divided the techniques for knowledge incorporation in two groups: (1) those that incorporate knowledge into the fitness evaluations, and (2) those which incorporate knowledge in the initialization process and the operators of an evolutionary algorithm. Each of these two groups has several ramifications (as shown in Figure 1.1), each of which are discussed in this chapter.

The remainder of this chapter is organized as follows. In Section 1.2, we discuss schemes that incorporate knowledge into the fitness evaluations of an evolutionary algorithm. The three schemes normally adopted (problem approximation, functional approximation, and evolutionary approximation) are all discussed in this section. Section 1.3 discusses the main schemes normally adopted to incorporate knowledge in the initialization and the operators of an evolutionary algorithm (namely, case-based reasoning and cultural algorithms). Finally, in Section 1.5, we provide some of the paths for future research within this topic that we consider worth exploring.

1.2 Knowledge Incorporation into the Fitness Evaluations

The high number of fitness evaluations often required by evolutionary algorithms is normally expensive, time-consuming or otherwise problematic in many real-world applications. Particularly in the following cases, a computationally efficient approximation of the original fitness function reducing either the number of duration of the fitness evaluations, is necessary:

- If the evaluation of the fitness function is computationally expensive.

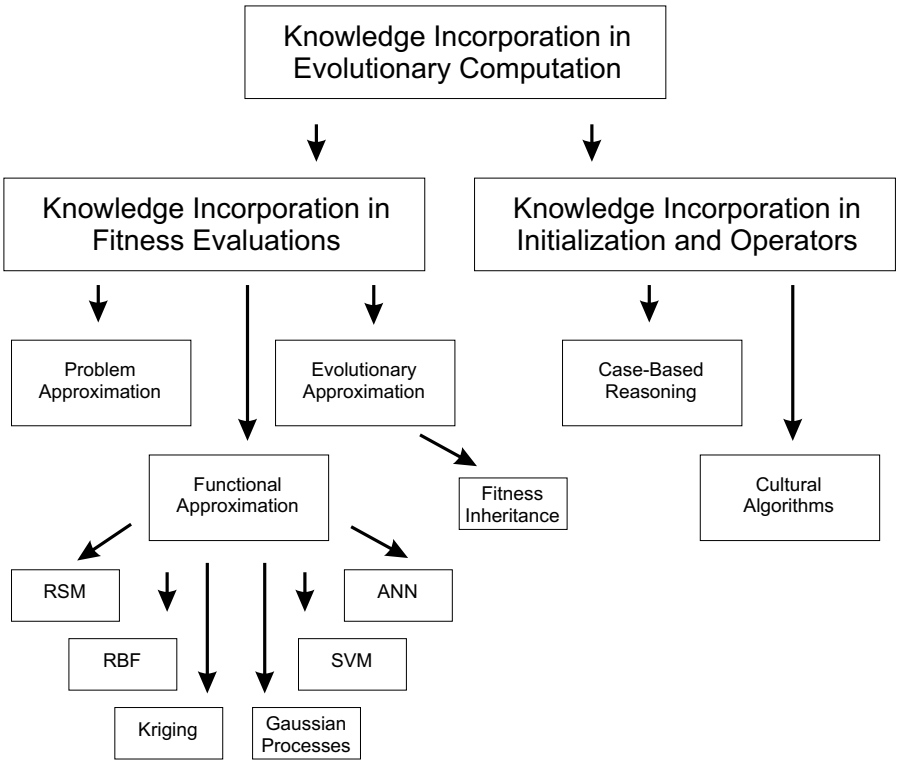

Fig. 1.1. A taxonomy of approaches for incorporating knowledge into evolutionary algorithms.

- If the fitness function cannot be defined in an algebraic form (e.g., when the fitness function is generated by a simulator).
- If additional physical devices must be used to determine the fitness function and this requires human interaction.
- If parallelism is not allowed.
- If the total number of evaluations of the fitness function is limited by financial constraints.

In the above cases, the approximation is then used to predict promising new solutions at a smaller evaluation cost than that of the original problem. Jin [32] discusses various approximation levels or strategies adopted for fitness approximation:

Problem Approximation: Tries to replace the original statement of the problem by one which is approximately the same as the original problem but which is easier to solve. To save the cost of experiments, numerical simulations instead of physical experiments are used to pseudo-evaluate the performance of a design.

Functional Approximation: In this approximation, a new expression is constructed for the objective function based on previous data obtained from the real objective functions. In this case, models obtained from data are often known as meta-models or surrogates (see section 1.2.1)

Evolutionary Approximation: This approximation is specific for EAs and tries to save function evaluations by estimating an individual's fitness from other similar individuals. A popular subclass in this category is known as fitness inheritance (see section 1.2.1)

Currently, there exist several evolutionary algorithms that use a meta-model to approximate the real fitness function and reduce the total number of fitness evaluations without degrading the quality of the results obtained. To achieve this goal, meta-models should be combined with the original fitness function in a proper manner. The mechanism adopted to balance the use of the meta-model and the real objective function is known as *evolution control*. Evolution control takes an important role when meta-models are combined with the original fitness function. In such cases, most meta-models could converge to a local optimum if they are provided incorrect knowledge (or information) about the real function. There are two different forms to combine the approximated model and the real function:

Individual-based evolution control: In this case, some individuals use meta-models to evaluate their fitness value and others (in the same generation) use the real fitness function. The main issue in individual-based evolution control is to determine which individuals should use the meta-model and which ones should use the real fitness function for fitness evaluation. They can be randomly chosen from the population, or one could simply choose the best individuals in the population to be evaluated using the real function (see Figure 1.2).

Generation-based evolution control: The main issue in generation-based evolution control is to determine in which generations the meta-model should be used and in which generations the real fitness function should be used. In this control, the real fitness function is applied at every g generations, where g is predefined and fixed throughout the evolutionary process (see Figure 1.3).

In the above cases, the approximation is used to predict promising new solutions at a smaller evaluation cost than that of the original problem. Current functional approximation models include Polynomials (e.g., response surface methodologies [50, 22]), neural networks (e.g., multi-layer perceptrons (MLPs) [24, 27, 49]), radial-basis-function (RBF) networks [46, 67, 73], support vector machines (SVMs) [65, 1], Gaussian processes [68, 3], and Kriging [18, 51]; all of them can be used for constructing meta-models.

Various approximation levels or strategies adopted for fitness approximation in evolutionary computation are proposed in [32].

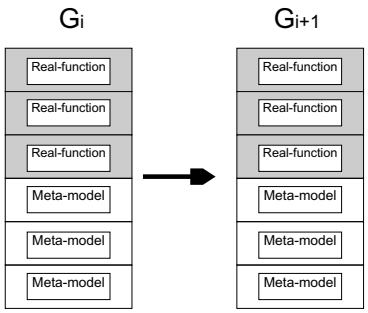

Fig. 1.2. Individual-based evolution control.

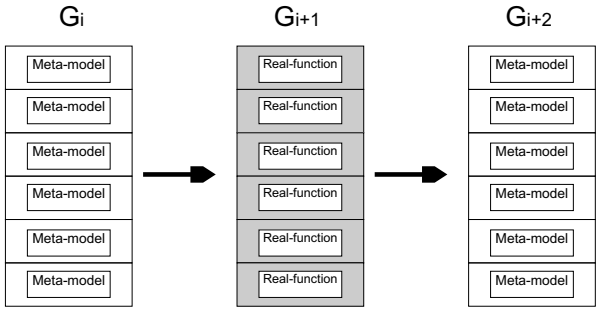

Fig. 1.3. Generation-based evolution control.

1.2.1 Surrogates

Surrogate models can perform a number of tasks in support of a computational analysis. Through interpolation, extrapolation and/or integration, these models can be used to address complex problems involving experimental design, system analysis and prediction.

In a single-objective optimization context, surrogate models have been successful in dealing with highly demanding problems where the cost of evaluating the real fitness function is very expensive (computationally speaking).

The accuracy of the surrogate model relies on the number of samples provided in the search space, as well as on the selection of the appropriate model to represent the objective functions. There exist a variety of techniques for constructing surrogate models (see for example [69]). One approach is least-square regression using low-order polynomials, also known as response surface methods. A statistical alternative for constructing surrogate models is Kriging, which is also referred to as "Design and Analysis of Computer Experiments" (DACE) models [61] and Gaussian process regression [71]. Comparisons of several surrogate modeling techniques are presented by Giunta and Watson [21] and by Jin et al. [30].

A surrogate model is built when the objective functions are to be estimated. This local model is built using a set of data points that lie on the

local neighboorhood of the design. Since surrogate models will probably be built thousands of times during the search, computational efficiency is the main objective. This motivates the use of radial basis functions, which can be applied to approximate multiple data, particularly when hundreds of data points are used for training.

Chafekar et al. [5] proposed a multi-objective evolutionary algorithm called OEGADO, which runs several Genetic Algorithms (GAs) concurrently with each GA optimizing one objective function at a time, and forming a reduced model (based on quadratic approximation functions) with this information. At regular intervals, each GA exchanges its reduced model with the others. This technique can solve constrained optimization problems in 3,500 and 8,000 evaluations and is compared with respect to the NSGA-II [14] and the $\epsilon - MOEA$ [12, 13].

Emmerich et al. [19] present a local Gaussian random field meta-model (GRFM) to predict objective function values by exploiting information obtained during previous evaluations. This scheme was created for optimizing computationally expensive problems. This method selects the most promising population members at each generation so that they are evaluated using the real objective function. This approach was tested on a 10 dimensional airfoil optimization problem and was compared with respect to the NSGA-II in the generalized Schaffer problems.

Polynomials: Response Surface Methods (RSM)

The response surface methodology comprises regression surface fitting in order to obtain approximate responses, design of experiments in order to obtain minimum variances of the responses and optimizations using the approximated responses.

An advantage of this technique is that the fitness of the approximated response surfaces can be evaluated using powerful statistical tools. Additionally, the minimum variances of the response surfaces can be obtained using design of experiments with a small number of experiments.

For most response surfaces, the functions adopted for the approximations are polynomials because of their simplicity, although other types of functions are, of course, possible. For the cases of quadratic polynomials, the response surface is described as follows:

$$\hat{y} = (\beta_0) + \sum_{1 \leq i \leq k} (\beta_i * x_i) + \sum_{1 \leq i \leq j \leq n} (\beta_{k-1+i+j} * x_i * x_j) \tag{1.1}$$

where k is the number of variables, and β_0 and β_i are the coefficients to be calculated. To estimate the unknown coefficients of the polynomial model, both the least squares method (LSM) and the gradient method can be used, but either of them requires at least the same number of samples of the real objective function than the k_t coefficients in order to obtain good results.

Goel et al. [22] is one of the few works that has used RSM in multi-objective problems. In this paper, the NSGA-II [14] an a local search strategy called "$\epsilon - constraint$" are adopted to generate a solution set that is used for approximating the Pareto optimal front by a response surface method (RSM). This method is applied to a rocket injector design problem.

There are few applications of the use of surrogates in evolutionary multi-objective optimization. Two of them are briefly discussed next.

Bramanti et al. [2] tried to reduce the computational cost of a multi-objective evolutionary algorithm using neural networks interpolation for building an objective response surface in order to find multiple trade-off solutions in electromagnetic design problems.

Wilson et al. [72] used two types of surrogate approximations (response surfaces and Kriging models) in order to reduce the computational expense of designing piezomorph actuators. The method shows that is flexible and can accommodate a wide variety of experimental designs and approximations. The authors also show that this method works well for both convex and non-convex Pareto fronts.

Radial Basis Functions

Radial Basis Functions (RBFs) were first introduced by R. Hardy in 1971 [23]. This term is made up of two different words: *radial* and *basis functions*. A radial function refers to a function of the type:

$$g : \mathbb{R}^d \rightarrow \mathbb{R} : (x_1, \ldots, x_d) \mapsto \phi(\|x_1, \ldots, x_d\|_2)$$

for some function $\phi : \mathbb{R} \rightarrow \mathbb{R}$. This means that the function value of g at a point $\vec{x} = (x_1, \ldots, x_d)$ only depends on the Euclidean norm of $\vec{x}$:

$$\|\vec{x}\|_2 = \sqrt{\sum_{i=0}^{d} x_i^2} = \text{distance of } \vec{x} \text{ to the origin}$$

And this explains the term *radial*. The term *basis function* is explained next. Let's suppose we have certain points (called centers) $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^d$. The linear combination of the function g centered at the points $\vec{x}$ is given by:

$$f : \mathbb{R}^d \mapsto \mathbb{R} : \vec{x} \mapsto \sum_{i=1}^{n} \lambda_i g(\vec{x} - \vec{x_i}) = \sum_{i=1}^{n} \lambda_i \phi(\|\vec{x} - \vec{x_i}\|) \qquad (1.2)$$

where $\|\vec{x} - \vec{x_i}\|$ is the Euclidean distance between the points $\vec{x}$ and $\vec{x}_i$. So, f becomes a function which is in the finite dimensional space spanned by the basis functions:

$$g_i : \vec{x} \mapsto g(\|\vec{x} - \vec{x_i}\|)$$

Now, let's suppose that we already know the values of a certain function $H : \mathbb{R}^d \mapsto \mathbb{R}$ at a set of fixed locations $\overrightarrow{x_1}, \ldots, \overrightarrow{x_n}$. These values are named $f_j = H(\overrightarrow{x_j})$, so we try to use the $\overrightarrow{x_j}$ as centers in the equation 1.2. If we want to force the function f to take the values f_j at the different points $\overrightarrow{x_j}$, then we have to put some conditions on the λ_i. This implies the following:

$$\forall j \in \{1, \ldots, n\} \; f_j = f(\overrightarrow{x_j}) = \sum_{i=1}^{n} (\lambda_i \cdot \phi(\|\overrightarrow{x_j} - \overrightarrow{x_i}\|))$$

In these equations, only the λ_i are unknown, and the equations are linear in their unknowns. Therefore, we can write these equations in matrix form:

$$\begin{bmatrix} \phi(0) & \phi(\|x_1 - x_2\|) & \ldots & \phi(\|x_1 - x_n\|) \\ \phi(\|x_2 - x_1\|) & \phi(0) & \ldots & \phi(\|x_2 - x_n\|) \\ \vdots & \vdots & & \vdots \\ \phi(\|x_n - x_1\|) & \phi(\|x_n - x_2\|) & \ldots & \phi(0) \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \tag{1.3}$$

Typical choices for the kernel $g(\overrightarrow{x})$ include linear splines, cubic splines, multiquadrics, thin-plate splines and Gaussian functions as shown in Table 1.1.

Type of Radial Function		
LS	linear splines	$\|r\|$
CS	cubic splines	$\|r\|^3$
MQS	multiquadrics splines	$\sqrt{1 + (\epsilon r)^2}$
TPS	thin plate splines	$\|r\|^{2m+1} \ln \|r\|$
GA	Gaussian	$e^{-(\epsilon r)^2}$

Table 1.1. Radial Basis Functions

Ong et al. [47] used surrogate models (RBFs) to solve computationally expensive design problems with constraints. The authors used a combination of a parallel evolutionary algorithm coupled with sequential quadratic programming in order to find optimal solutions of an aircraft wing design problem. In this case, the authors construct a local surrogate model based on radial basis functions in order to approximate the objective and constraint functions of the problem.

Karakasis et al. [33] used surrogate models based on radial basis functions in order to deal with computationally expensive problems. A method called Inexact Pre-Evaluation (IPE) is applied into a MOEA's selection mechanism. Such method helps to choose the individuals that are to be evaluated using the real objective function, right after a meta-model approximation has been obtained by the surrogate. The results are compared against a conventional

MOEA in two test-problems, one from a benchmark and one from the turbomachinery field.

Voutchkov & Keane [70] studied several surrogate models (RSM, RBF and Kriging) in the context of multi-objective optimization using the NSGA-II [14] as the MOEA that optimized the meta-model function given by the surrogate. The surrogate model is trained with 20 initial points and the NSGA-II is run on the surrogate model. Then, the 20 best resultant points given by the optimization are added to the existing data pool of real function evaluations and the surrogate is re-trained with these new solutions. A comparison of results is made in 4 test functions (from 2 to 10 variables), performing 400 real fitness function evaluations.

Kriging

In Kriging, the meta-model prediction is formed by adding up two different models as follows:

$$y(\overrightarrow{x}) = a(\overrightarrow{x}) + b(\overrightarrow{x})$$

where $a(\overrightarrow{x})$ represents the "average" long-term range behavior and the expected value of the true function. This function can be modeled in various ways, such as with polynomials or with trigonometric series as:

$$a(\overrightarrow{x}) = a_0 + \sum_{i=1}^{L}\sum_{j=1}^{R} a_{ij}(x_i)^j$$

where: R is the polynomial order with L dimensions and $b(\overrightarrow{x})$ stands for a local deviation term. $b(\overrightarrow{x})$ is a Gaussian random function with zero mean and non-zero covariance that represents a localized deviation from the global model. This function represents a short-distance influence of every data point over the global model. The general formulation for $b(\overrightarrow{x})$ is a weighted sum of N functions, $K_n(\mathbf{x})$ that represent the covariance functions between the n^{th} data point and any point $\mathbf{x}$:

$$b(\overrightarrow{x}) = \sum_{n=1}^{N} b_n K(h(x, x_n)) \;\&\; h(x, x_n) = \sqrt{\sum_{i=1}^{L}\left(\frac{x_i - x_{in}}{x_i^{max} - x_i^{min}}\right)^2}$$

where x_i^{min} and x_i^{max} are the lower an upper bounds of the search space and x_{in} denotes the $i-th$ component of the data point x_n. However, the shape of $K(h)$ has a strong influence on the resulting aspect of the statistical model. And that is why it is said that Kriging is used as a estimator or an interpolator.

Knowles [34] proposed "ParEGO", which consists of a hybrid algorithm based on a single optimization model (EGO) and a Gaussian process, which is updated after every function evaluation, coupled to an evolutionary algorithm.

EGO is a single-objective optimization algorithm that uses Kriging to model the search landscape from the solutions visited during the search and learns a model based on Gaussian processes (called DACE). This approach is used to solve multi-objective optimization problems of low dimensionality (up to 6 decision variables) with only 100 and 250 fitness function evaluations.

Neural Networks

Artificial neural networks (ANNs) provide a general, practical method for learning real-valued, discrete-valued, and vector-valued functions from examples. ANN learning is robust to errors in the training data and has been succesfully applied to several problems. An ANN basicaly builds a map between a set of inputs and the respective outputs and are good to deal with nonlinear regression analysis and incomplete data. The main objective in the construction of an ANN is defining its appropriate architecture, that is, the number of layers and the number of nodes in each layer, given a certain number of inputs and outputs.

The Multi-Layer Perceptron (MLP) network has been widely used in function approximation problems, because it has been often found to provide compact representations of mappings in real-world problems. An MLP is composed of neurons which are very close to the ones represented in the case of the linear network. The linear neurons are modified so that a slight nonlinearity is added after the linear summation. The output (y) of each neuron is thus:

$$y = \phi(\sum_{i=1}^{n} w_i \cdot a_i + b)$$

where a_i are the inputs of the neuron and w_i are the weights of the neuron. The nonlinear function ϕ is called the activation function as it determines the activation level of the neuron.

In Figure 1.4, we show an MLP network with one layer of linear output neurons and one layer of nonlinear neurons between the input and output neurons. The middle layers are usually called *hidden layers*. Note that a graphical model of the conditional dependences would not include the middle layer because the computational units are unknown variables of the model whereas the weights would be included as nodes of the model.

Some applications of ANNs to evolutionary multi-objective optimization are the following:

Farina [20] proposed "NN-GRS", which is an extension of the single-objective Neural Network-based GRS (Generalized Response Surface) method. The main idea of this approach is to maintain two different objective functions, one real and another one which is an approximation (neural network based).

Nain & Deb [45] proposed "NSGA-II-ANN", which combines the NSGA-II algorithm [14] with a new method based on neural networks as the basic

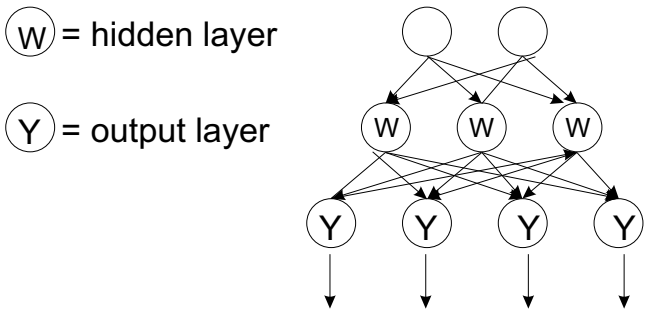

Fig. 1.4. A graphical representation of an MLP network with one hidden layer

approximation technique for fitness computation. This meta-model is updated at each generation, and it provides a more refined approximate model to guide the search of the NSGA-II in subsequent generations.

Fitness Inheritance

Fitness Inheritance is a technique proposed by Smith et al. [66], whose main motivation is to reduce the total number of fitness evaluations made by an evolutionary algorithm. The mechanism works as follows: when assigning the fitness to an individual, sometimes the objective function is evaluated as usual, but the rest of the time, the fitness of an individual is assigned as the average of the fitnesses of its parents, thus avoiding a fitness function evaluation based on the assumption of similarity of the individual to its parents.

In the original proposal, inheritance was made in two ways: first, it is computed as the average of the two parents:

$$f(x) = \frac{f(p_1) + f(p_2)}{2}$$

where x is the individual to evaluate, and p_1 and p_2 are the parents of x. The second way is a weighted sum, where one of the parents may be more important. This weighted inheritance is obtained by:

$$f(x) = \frac{w_1 f(p_1) + w_2 f(p_2)}{w_1 + w_2}$$

where w_1 and w_2 are the weigths for the two parents.

The OneMax problem was used in this early proposal [66], and the authors showed a decrease in the number of fitness function evaluations, when comparing this approach to another one without fitness inheritance. Other applications of fitness inheritance are: the design of vector quantization codebooks [74], where good results were obtained; and image compression [62], where the authors reported a significant reduction in the number of evaluations. This last approach is a more complex inheritance mechanism, where the

individuals store also a variable that reflects the quality of the approximations with respect to the actual fitness values (a value of one means that the real objective function was evaluated). Such quality is decreased over generations, since the evolutionary algorithm tends to converge and the distance of an individual with respect to its parents tends to decrease over time.

Fitness inheritance must not be always applied, since the algorithm needs to use the true fitness function several times, in order to obtain enough information to guide the search. The percentage of time in which fitness inheritance is applied is called *inheritance proportion*. As this inheritance proportion tends to one (i.e., inherit all the time), the algorithm is most likely to prematurely converge [6].

There are a few theoretical studies about the fintess inheritance technique. For example, in [64], the authors present a theoretical model, which is used to obtain the convergence time, the optimal population sizing and the optimal inheritance proportion. Regarding the inheritance proportion, the authors found that, for problems of moderate and large size, the optimal values are between 0.54 and 0.558. They also defined the speedup of fitness inheritance when comparing it with respect to an algorithm without fitness inheritance (this is analogous to the speedup achieved when parallelizing an algorithm, with respect to executing in sequentially).

The work of Sastry et al. [64] was extended to the multi-objective case by Chen et al. [6]. In this paper, the authors use fitness sharing to maintain diversity in the population an to be able to cover a larger extension of the Pareto front. The problem they solve is a bi-objective extension of the One-Max problem originally solved by Sastry et al. [64]. Chen et al. [6] also present generalizations of the theoretical work reported in [64] about the convergence time, the optimal population sizing, the optimal inheritance proportion and the speedup. The experiments carried out show that, savings of up to 40% of the total number of evaluations can be achieved when using fitness inheritnance alone. Additionally, when using fitness inheritance in combination with fitness sharing, savings of up to 25% can be obtained.

Bui el al. [4] presented a multi-objective approach for dealing with noisy functions. As they try to characterize the noise present, they evaluate a single individual several times, and this increases the computational cost associated with evaluations of the fitness function. They use fintess inheritance as an alternative to reduce the cost required by the approach. In their experiments they use the NSGA-II, with some specific anti-noise mechanisms, such as a probabilistic model and resampling. The test functions adopted are a noisy version of the ZDT set [75], which is a commonly used benchmark in evolutionary multi-objective optimization. The results show savings of up to 33% of the total number of fitness evaluations performed.

Fitness inheritance has also been used with particle swarm optimization. The main mechanism needs a modification in this case, because of the fact that in particle swarm optimization there are no parents or children, but leaders and previous positions of a particle. In [52], the authors propose several

ways to make the inheritance, taking into account the previous position of the partice (the so called *pbest*), and the leader. They compare the performance of different versions of fitness inheritance in some multi-objective problems (the ZDT test functions [75] and the DTLZ test functions [15]), and also compare the approach with respect to other multi-objective versions of particle swarm optimization. The results show that the proposed approach is very competitive with respect to other techniques, providing savings that range between 30% and 40%

A more recent approach proposes the use of dynamic rules to assign the inheritance proportion in particle swarm optimization [53]. This scheme is proposed again for multi-objective problems. Five rules were proposed in [53]:

$$p_i(gen) = \left(\frac{gen}{Gmax}\right)^4$$
$$p_i(gen) = \left(\frac{gen}{Gmax}\right)^2$$
$$p_i(gen) = \frac{gen}{Gmax} - \frac{\sin\left(2\pi\frac{gen}{Gmax}\right)}{6.3}$$
$$p_i(gen) = \frac{gen}{Gmax}$$
$$p_i(gen) = \left(\frac{gen}{Gmax}\right)^{1/2}$$
$$p_i(gen) = \left(\frac{gen}{Gmax}\right)^{1/4}$$

where $p_i(gen)$ is the inheritance proportion at generation gen, and $Gmax$ is the total number of generations that the algorithm will run.

The use of these rules results on savings from 19% up to 78% of the total number of evaluations. However, the greater the savings in the number of evaluations, the greater is the degradation in the quality of the results. Nevertheless, the authors show that functions that provide up to 49% of savings present no significant loss in the quality of the results. The approach was tested with the well known ZDT problems [75].

As a final note on fitness inheritance, it is important to mention that some researchers consider this mechanism not so useful in complex or real world problems, because it has been only applied to "easy" problems. For example [17] tested the original scheme of fitness inheritance on a standard binary GA and the ZDT problems [75]. From this study, the authors concluded that fitness inheritance is not useful when the shape of the Pareto front is nonconvex or discontinuous. These conclusions are true for the original proposal, but it is possible to overcome them, as shown in [54], so that fitness inheritance can be successfully adopted with Pareto fronts of any shape.

1.3 Knowledge Incorporation in the Initialization and the Operators

In addition to the techniques for incorporating knowledge during the evaluation of new individuals (i.e., directly affecting fitness calculation), some researchers have explored the use of knowledge at different stages of the evolutionary search. The aim, however, is the same: either to speedup convergence or to reduce the computational cost of the algorithm (measured in terms of the number of fitness function evaluations performed).

In this regard, several options are possible. For example, some domain knowledge previously obtained (or provided by a human expert) may be used to build a database of cases that are adopted as a reference for conducting a more efficient search. It is also possible to seed good solutions in the initial population of an evolutionary algorithm, or to add knowledge (perhaps based on experience) to the variation operators (recombination and mutation), in order to facilitate the generation of good solutions. Next, we will these choices in more detail.

1.3.1 Case-Based Reasoning

The case-based reasoning method [60] consists of a historical database with cases of study of the problem to be solved. When new problems appear, the system looks for a case in the database which matches the current problem, and adapts the previous solutions. Sometimes the system can even store the process to obtain a solution, in order to execute the same or a similar process when similar problems appear.

Early applications of case-based reasoning in evolutionary computation consist only of a tool to analyze the evolutionary process, making use of the building blocks theory [42]. The authors in this case, used several test functions, including the circuits design problem, and they applied the case-based reasoning techniques to discover the search path the algorithm followed, basically through the discovery of "good" building blocks. The authors argue that these tools can be used to post-process solutions, and they are also useful when dealing with deceptive problems.

An attempt to improve the performance of an evolutionary algorithm with case-based reasoning, is the use of a memory that contains previous solutions, obtained from similar problems [40]. The population is initialized in this case with some of these solutions, in order to move the search to potentially good regions. Louis [40] proposed a methodology for selecting appropriate initialization cases. The approach works very well in combinatorial circuit design problems, showing that the time required to solve a problem decreases with respect to that required by the original algorithm. A similar application of case-based reasoning techniques to the extraction of design patterns within the context of combinational circuit design is reported by Islas et al. [28].

Similar approaches of injection of solutions obtained from a memory of previous runs were used to solve different instances of the traveling salesman problem [43], for dynamic strike force asset allocation [39], for electronic circuits design [41], and for games [44].

In the case of [43], the strategy is not only to inject potential solution at the beginning of the evolutionary process, but also periodically during the search. The authors also develop a similarity measure to select the appropriate cases for the traveling salesman problem.

In [39], the authors investigate the number of individuals that can be injected to the algorithm while it obtains good results. They conclude that if too many individuals are injected, the algorithm is more likely to have premature convergence (a similar feature present when using surrogates and fitness inheritance), and recommend to inject only between 5% and 15% of solutions.

More recent work investigates the changes in the convergence rates when applying injection of solutions [16], the identification of the similarity of the solved problems, and those to be solved.

Despite the lack of applications of case-based reasoning in evolutionary multi-objective optimization, there are a very interesting potential applications of this technique, including a multiple objective version of the traveling salesman problem, and several types of scheduling problems. These and many other applications could certainly benefit from the use of databases of cases and/or from domain knowledge provided by human experts.

1.4 Cultural Algorithms

Cultural algorithms were proposed by Robert Reynolds [55], as an approach that tries to add domain knowledge to an evolutionary algorithm during the search process, avoiding the need to add it *a priori*. This approach uses, in addition to the population space commonly adopted in evolutionary algorithms, a belief space, which encodes the knowledge obtained from the search points and their evaluation, in order to influence the evolutionary operators that guide the search. However, the belief space is commonly designed based on the group of problems that is to be solved.

At each generation, the cultural algorithm selects some exemplar individuals from the population, in order to extract information from them that can be useful during the search. Such an information is used to update the belief space. The belief space will then influence the operators of the evolutionary algorithm, to transform them in informed operators and enhance the search process. These interactions between the spaces of a cultural algorithm are depicted in Figure 1.5.

Regarding optimization, cultural algorithms have been applied mainly to single-objective problems. One of the first applications was the Boole problem [55], which uses a genetic algorithm for the population space, and version

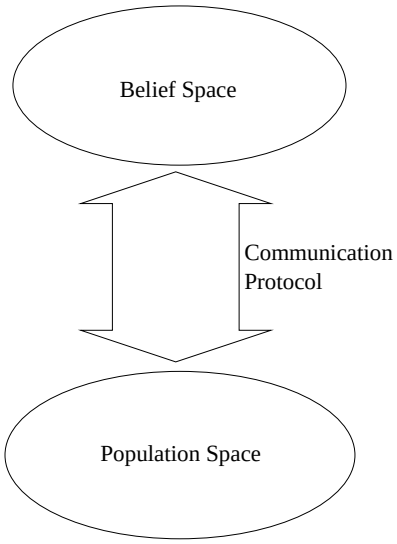

Fig. 1.5. Spaces of a cultural algorithm

spaces for the belief space. In this case, a graph (the belief space) of the schemes in the population is built and classified based on the fitness of each particular instance. This is a very illustrative approach, because the graph's dynamics will reflect the discovering of good and bad schemes.

Constrained and dynamic optimization problems have been a a very active application area for cultural algorithms. Reynolds et al. [59] proposed a cultural version of GENOCOP for solving convex constrained optimization problems. The belief space in this approach constructs boundaries to include the feasible region, and then produce new solutions only within that region.

Later on, Chung and Reynolds [57] developed the CAEP (Cultural Algorithms with Evolutionary Programming) for global optimization, with a very rich model in the belief space, and very encouraging results. For example, in [58] the authors proposed a formal model of self-adaptation in cultural algorithms, that supports the three main levels of self-adaptation in evolutionary algorithms (population, individual and component level). The royal road functions were adopted as a case of study in this work.

The CAEP was tested on a number of global optimization problems [7], showing its improvement when compared to the standard evolutionary programming algorithm. In this work, the belief space was divided in two parts, called knowledge sources, specifically designed for real-valued problems: the situational knowledge and the normative knowledge. A general description of these knowledge sources, and those designed and added later, is provided in Table 1.2.

Table 1.2. Knowledge sources for real-parameter optimization in cultural algorithms

Knowledge source	**Description**
Situational knowledge	Consists of the best exemplars found in the population, which represent leaders to follow. The individuals generated through this source, will tend to be closer to the leaders.
Normative knowledge	Consists of a set of intervals for each decision variable where good solutions have been found. The individuals generated through this source are more likely to be within the intervals, so they exploit good regions.
Topographical knowledge	Consists of a set of cells that represent a region of the search space. Each cell stores a characteristic of the region it represents; for example, the feasability of that region. The individuals generated through this source will be closer to the best cells.
History knowledge	Consists of a set of previous local optima, and its function is to extract patterns about their position. The individuals generated through this source will try to find in advance the location of the next local optimum. This knowledge source can also be used to add diversity to the algorithm, since it attempts to explore new regions.
Domain knowledge	It has no defined structure, because it depends of the problem which is to be solved. Its function is to exploit some knowledge about the problem, if available.

A CAEP for nonlinear constrained optimization, a more general problem than the one tackled in GENOCOP, was proposed in [31]. To handle constraints, a third knowledge source, called topographical knowledge, was added to the belief space. It consists of a set of cells, which store some characteristic of the region of the search space they represent. In this case, they store the feasibility of the region, based on the explored points within them.

Jin's approach [31] was extended in [8], improving its computational efficiency, and overcoming its scalability problems. In the original approach the topographical knowledge was stored as a n-dimentional grid of the search space. This was replaced by a spatial data structure, that requires a controlled amount of memory even when the number of dimensions grows. Additionally, the authors present an empirical study in which this approach is validated using a well-known benchmark adopted in evolutionary constrained optimization, and results are compared with respect to constraint-handling techniques representative of the state-of-the-art in the area. The cultural algorithm reported in [8] is able to find competitive results, while performing only about 15% of the total number of fitness function evaluations required by the other approaches with respect to which it was compared.

Saleem and Reynolds [63] added two more knowledge sources to cultural algorithms, in order to deal with dynamic environments: history knowledge and domain knowledge. The first of these sources was designed to extract patterns about the changes of position of optimal points at each environmental change. The second source was designed to exploit the known characteristics of the function generator. Even when these knowledge sources were designed for dynamic problems, they have also been used in static environments [48, 36].

Another cultural algorithm in which the population space adopts particle swarm optimization was proposed by Peng et al. [48]. The authors use all of the previously designed knowledge sources, and they investigate the role of the belief space in the different stages of the optimization process.

Differential evolution has also been used for the population space of a cultural algorithm [36, 37], also focusing on constrained optimization. In this case, all the knowledge sources were adapted for its use with the differential evolution operator, providing some adaptation of its components. The approach was tested on a well known benchmark and also on some engineering optimization problems, with good results.

Finally, there is a cultural algorithm in which genetic programming is used for the population space [56]. In this case, the belief space consists of a set of subgraphs which have been frequently found in the population.

There are very few attempts to use cultural algorithms for multi-objective optimization. The first is a CAEP [9], which uses Pareto ranking, and an approximation of the dimensions of the Pareto front in the belief space. The belief space works as a guide for the individuals to reach regions where nondominated solutions have been found. The belief space includes also a mechanism to obtain a good distribution of the resulting points along the Pareto front.

Recently, the cultural algorithm with differential evolution ([36]) for constrained optimization was proposed, together with the ε-constraint method, to solve multi-objective problems [38]. In this work, the authors identify some hard problems from the DTLZ [15] and WFG [25, 26] benchmarks, and then they apply their technique to them. This approach is computationally expensive (because of the optimization processes to obtain one point at a time), but this cost, when affordable, is worth trying, because this approach can solve very difficult problems that other modern multi-objective evolutionary algorithms (e.g., the NSGA-II [14]) cannot solve, even if allowed to perform a very high number of fitness function evaluations.

1.5 Future Perspectives

From our previous discussion of the most representative work regarding incorporation of knowledge into evolutionary algorithms, we could identify several topics in which researchers working on evolutionary multi-objective optimization have not done much work (or where there is no work at all). Some of these topics are the following:

- One of the key issues when adopting meta-models for reducing the number of evaluations of a MOEA, is the design of the *evolution control* that keeps a proper balance between the evaluations done in the meta-model and those performed with the actual objective function. More detailed studies regarding the design of such evolution control are necessary in the context of multi-objective optimization.
- To the authors' best knowledge, so far, nobody has used support vector machines within the context of evolutionary multi-objective optimization (i.e., for predicting good solutions).
- The development of cultural algorithms for multi-objective optimization is another promising path for future research. The aim in this case, should be not only to reduce the number of fitness function evaluations to be performed, but also to be able to solve problems that are very hard for other multi-objective evolutionary algorithms.
- Case-based reasoning techniques have not been used so far (to the authors' best knowledge) within the context of evolutionary multi-objective optimization. This sort of approach may be very useful to seed good solutions in the initial population of a MOEA, or to extract certain "patterns" from the population of a MOEA that could be used to speedup convergence in similar multi-objective optimization problems.
- Scalability is another important topic that needs to be studied in more depth. In the few papers that we reviewed in this chapter, MOEAs that benefit from knowledge incorporation (particularly those adopting meta-models) have been used only in problems with few decision variables (no more than ten). Thus, the scalability of current techniques to large dimensional problems is a very important research topic that certainly deserves attention.

Acknowledgments

The first and second authors acknowledge support from CONACyT through a scholarship to pursue graduate studies at the Computer Science Department of CINVESTAV-IPN. The third author acknowledges support from CONACyT project number 42435-Y.

References

1. M. Bhattacharya and G. Lu. A dynamic approximate fitness based hybrid ea for optimization problems. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 1879–1886, 2003.
2. A. Bramanti, P. Di Barba, M. Farina, and A. Savini. Combining response surfaces and evolutionary strategies for multiobjective Pareto-optimization in electromagnetics. *International Journal of Applied Electromagnetics and Mechanics*, 15(1):231 – 236, 2001.

3. D. Bueche, N.N. Schraudolph, and P. Koumoutsakos. Accelerating evolutionary algorithms with gaussian process fitness function models. *IEEE Trans. on Systems, Man, and Cybernetics: Part C*, 2004. In press.
4. L.T. Bui, Hussein Abbass, and D. Essam. Fitness inheritance for noisy evolutionary multi-objective optimization. In *Proceedings of genetic and Evolutionary Computation Conference (GECCO-2005)*, pages 779–785. ACM, 2005.
5. D. Chafekar, L. Shi, K. Rasheed, and J. Xuan. Multi-objective GA optimization using reduced models. *IEEE Transactions on Systems, Man, and Cybernetics: Part C*, 35(2):261–265, May 2005.
6. Jian-Hung Chen, David E. Goldberg, Shinn-Ying Ho, and Kumara Sastry. Fitness Inheritance in Multi-Objective Optimization. In W.B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M.A. Potter, A.C. Schultz, J.F. Miller, E. Burke, and N. Jonoska, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pages 319–326, San Francisco, California, July 2002. Morgan Kaufmann Publishers.
7. Chan-Jin Chung and Robert G. Reynolds. CAEP: An Evolution-based Tool for Real-Valued Function Optimization using Cultural Algorithms. *Journal on Artificial Intelligence Tools*, 7(3):239–292, 1998.
8. Carlos A. Coello Coello and Ricardo Landa Becerra. Adding knowledge and efficient data structures to evolutionary programming: A cultural algorithm for constrained optimization. In Erick Cantú-Paz et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pages 201–209, San Francisco, California, July 2002. Morgan Kaufmann Publishers.
9. Carlos A. Coello Coello and Ricardo Landa Becerra. Evolutionary Multiobjective Optimization using a Cultural Algorithm. In *2003 IEEE Swarm Intelligence Symposium Proceedings*, pages 6–13, Indianapolis, Indiana, USA, April 2003. IEEE Service Center.
10. Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002. ISBN 0-3064-6762-3.
11. Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001. ISBN 0-471-87339-X.
12. Kalyanmoy Deb, Manikanth Mohan, and Shikhar Mishra. Towards a Quick Computation of Well-Spread Pareto-Optimal Solutions. In Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele, editors, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 222–236, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.
13. Kalyanmoy Deb, Manikanth Mohan, and Shikhar Mishra. Evaluating the ϵ-Domination Based Multi-Objective Evolutionary Algorithm for a Quick Computation of Pareto-Optimal Solutions. *Evolutionary Computation*, 13(4):501–525, Winter 2005.
14. Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.
15. Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Test Problems for Evolutionary Multiobjective Optimization. In Ajith Abraham, Lakhmi Jain, and Robert Goldberg, editors, *Evolutionary Multiobjective*

Optimization. Theoretical Advances and Applications, pages 105–145. Springer, USA, 2005.

16. Rich Drewes, Sushil J. Louis, Chris Miles, John McDonnell, and Nick Gizzi. Use of case injection to bias genetic algorithm solution of similar problems. In *Proceedings of the International Congress on Evolutionary Computation*, Canberra, Australia, 2003. IEEE Press.
17. Els I. Ducheyne, Bernard De Baets, and Robert De Wulf. Is Fitness Inheritance Useful for Real-World Applications? In Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele, editors, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 31–42, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.
18. M. Emmerich, A. Giotis, M. Özdenir, T. Bäck, and K. Giannakoglou. Metamodel-assisted evolution strategies. In *Parallel Problem Solving from Nature*, number 2439 in Lecture Notes in Computer Science, pages 371–380. Springer, 2002.
19. Michael T.M. Emmerich, Kyriakos C. Giannakoglou, and Boris Naujoks. Single- and Multiobjective Evolutionary Optimization Assisted by Gaussian Random Field Metamodels. *IEEE Transactions on Evolutionary Computation*, 10(4):421–439, August 2006.
20. M. Farina. A neural network based generalized response surface multiobjective evolutionary algorithms. In *Congress on Evolutionary Computation*, pages 956–961. IEEE Press, 2002.
21. A.A. Giunta and L. Watson. A comparison of approximation modeling techniques: Polynomial versus interpolating models. Technical Report 98-4758, AIAA, 1998.
22. T. Goel, R. Vaidyanathan, R. Haftka, W. Shyy, N. Queipo, and K. Tucker. Response surface approximation of pareto optimal front in multiobjective optimization. Technical Report 2004-4501, AIAA, 2004.
23. R. L. Hardy. Multiquadric equations of topography and other irregular surfaces. *J. Geophys. res*, 76:1905–1915, 1971.
24. 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.
25. Simon Huband, Luigi Barone, Lyndon While, and Phil Hingston. A Scalable Multi-objective Test Problem Toolkit. In Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, editors, *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, pages 280–295, Guanajuato, México, March 2005. Springer. Lecture Notes in Computer Science Vol. 3410.
26. Simon Huband, Phil Hingston, Luigi Barone, and Lyndon While. A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, October 2006.
27. M. Hüscken, Y. Jin, and B. Sendhoff. Structure optimization of neural networks for aerodynamic optimization. *Soft Computing Journal*, 9(1):21–28, 2005.
28. Eduardo Islas Pérez, Carlos A. Coello Coello, and Arturo Hernández Aguirre. Extraction and reuse of design patterns from genetic algorithms using case-based reasoning. *Soft Computing–A Fusion of Foundations, Methodologies and Applications*, 9(1):44–53, January 2005.

29. Mikkel T. Jensen. Reducing the Run-Time Complexity of Multiobjective EAs: The NSGA-II and Other Algorithms. *IEEE Transactions on Evolutionary Computation*, 7(5):503–515, October 2003.
30. R. Jin, W. Chen, and T.W. Simpson. Comparative studies of metamodeling techniques under miltiple modeling criteria. Technical Report 2000-4801, AIAA, 2000.
31. Xidong Jin and Robert G. Reynolds. Using Knowledge-Based Evolutionary Computation to Solve Nonlinear Constraint Optimization Problems: a Cultural Algorithm Approach. In *1999 Congress on Evolutionary Computation*, pages 1672–1678, Washington, D.C., July 1999. IEEE Service Center.
32. Yaochu Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12, 2005.
33. Marios K. Karakasis and Kyriakos C. Giannakoglou. Metamodel-Assisted Multi-Objective Evolutionary Optimization. In R. Schilling, W. Haase, J. Periaux, H. Baier, and G. Bugeda, editors, *EUROGEN 2005. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, Munich, Germany, 2005.
34. Joshua Knowles. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, January 2006.
35. H.T. Kung, F. Luccio, and F.P. Preparata. On finding the maxima of a set of vectors. *Journal of the Association for Computing Machinery*, 22(4):469–476, 1975.
36. Ricardo Landa Becerra and Carlos A. Coello Coello. Optimization with Constraints using a Cultured Differential Evolution Approach. In Hans-Georg Beyer et al., editor, *Genetic and Evolutionary Computation Conference (GECCO'2005)*, volume 1, pages 27–34, Washington, DC, USA, June 2005. ACM Press. ISBN 1-59593-010-8.
37. Ricardo Landa Becerra and Carlos A. Coello Coello. Cultured differential evolution for constrained optimization. *Computer Methods in Applied Mechanics and Engineering*, 195(33–36):4303–4322, July 1 2006.
38. Ricardo Landa Becerra and Carlos A. Coello Coello. Solving hard multiobjective optimization problems using e-constraint with cultured differential evolution. In *Parallel Problem Solving from Nature - PPSN VIII*, LNCS, Reykjavik, Iceland, 2006. Springer-Verlag.
39. Sushil J. Louis. Genetic learning for combinational logic design. In *Proceedings of the GECCO-2002 Workshop on Approximation and Learning in Evolutionary Computation*, page 2126, New York, NY, 2002.
40. Sushil J. Louis and J. Johnson. Solving similar problems using genetic algorithms and case-based memory. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97)*, San Francisco, CA, 1997. Morgan Kaufmann.
41. Sushil J. Louis, John McDonnell, and N. Gizzi. Dynamic strike force asset allocation using genetic algorithms and case-based reasoning. In *Proceedings of the Sixth Conference on Systemics, Cybernetics, and Informatics*, pages 855–861, Orlando, Florida, USA, 2002.
42. Sushil J. Louis, Gary McGraw, and Richard Wyckoff. Case-based reasoning assisted explanation of genetic algorithm results. *Journal of Experimental and Theoretical Artificial Intelligence*, 5:21–37, 1993.

43. Sushil J. Louis and Yongmian Zhang. A sequential similarity metric for case injected genetic algorithms applied to TSPs. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 377–384, Orlando, Florida, USA, 13-17 1999. Morgan Kaufmann.
44. Chris Miles and Sushil J Louis. Case-injection improves response time for a real-time strategy game. In *Proceedings of the 2005 IEEE Symposium on Computational Intelligence in Games*, New York, NY, 2005. IEEE Press.
45. P. K. S. Nain and K. Deb. Computationally effective search and optimization procedure using coarse to fine approximation. In *Congress on Evolutionary Computation*, pages 2081–2088, 2003.
46. 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.
47. Y.S. Ong, P.B. Nair, and A.J. Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA Journal*, 41(4):687–696, 2003.
48. Bin Peng, Robert G. Reynolds, and Jon Brewster. Cultural swarms. In *Proceedings of the Congress on Evolutionary Computation 2003 (CEC'2003)*. IEEE Service Center, 2003.
49. S. Pierret. Turbomachinery blade design using a Navier-Stokes solver and artificial neural network. *ASME Journal of Turbomachinery*, 121(3):326–332, 1999.
50. K. Rasheed, X. Ni, and S. Vattam. Comparison of methods for developing dynamic reduced models for design optimization. *Soft Computing Journal*, 2003. In press.
51. A. Ratle. Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In A. Eiben, Th. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature*, volume V, pages 87–96, 1998.
52. Margarita Reyes Sierra and Carlos A. Coello Coello. A Study of Fitness Inheritance and Approximation Techniques for Multi-Objective Particle Swarm Optimization. In *2005 IEEE Congress on Evolutionary Computation (CEC'2005)*, volume 1, pages 65–72, Edinburgh, Scotland, September 2005. IEEE Service Center.
53. Margarita Reyes-Sierra and Carlos A. Coello Coello. Dynamic Fitness Inheritance Proportion For Multi-Objective Particle Swarm Optimization. In Maarten Keijzer et al., editor, *2006 Genetic and Evolutionary Computation Conference (GECCO'2006)*, volume 1, pages 89–90, Seattle, Washington, USA, July 2006. ACM Press. ISBN 1-59593-186-4.
54. María Margarita Reyes Sierra. *Use of Coevolution and Fitness Inheritance for Multiobjective Particle Swarm Optimization*. PhD thesis, Computer Science Section, Department of Electrical Engineering, CINVESTAV-IPN, Mexico, August 2006.
55. Robert G. Reynolds. An Introduction to Cultural Algorithms. In A. V. Sebald and L. J. Fogel, editors, *Proceedings of the Third Annual Conference on Evolutionary Programming*, pages 131–139. World Scientific, River Edge, New Jersey, 1994.

56. Robert G. Reynolds. Cultural algorithms: Theory and applications. In David Corne, Marco Dorigo, and Fred Glover, editors, *New Ideas in Optimization*, pages 367–377. McGraw-Hill, London, UK, 1999.
57. Robert G. Reynolds and Chan-Jin Chung. A cultural algorithm framework to evolve multi-agent cooperation with evolutionary programming. In *EP '97: Proceedings of the 6th International Conference on Evolutionary Programming VI*, pages 323–334, London, UK, 1997. Springer-Verlag.
58. Robert G. Reynolds and Chan-Jin Chung. Knowledge-based self-adaptation in evolutionary programming using cultural algorithms. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC 97)*, pages 71–76, 1997.
59. Robert G. Reynolds, Zbigniew Michalewicz, and M. Cavaretta. Using cultural algorithms for constraint handling in GENOCOP. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, pages 298–305. MIT Press, Cambridge, Massachusetts, 1995.
60. Christopher K. Riesbeck and Roger C. Schank. *Inside Case-Based Reasoning.* Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA, 1989.
61. J. Sacks, W. Welch, T. Mitchell, and H. Wynn. Design and analysis of computer experiments (with discussion). In *Statistical Science*, volume 4, pages 409 – 435, 1989.
62. M. Salami and T. Hendtlass. A fast evaluation strategy for evolutionary algorithms. *Applied Soft Computing*, 2:156–173, 2003.
63. S. Saleem and R. Reynolds. Cultural algorithms in dynamic environments. In *Proceedings of the Congress on Evolutionary Computation 2000*, volume 2, pages 1513–1520, 2000.
64. 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 Publishers, 2001.
65. K. Abboud M. Schoenauer. Surrogate deterministic mutation. In *Artificial Evolution'01*, pages 103–115. Springer, 2002.
66. Robert E. Smith, B. A. Dike, and S. A. Stegmann. Fitness inheritance in genetic algorithms. In *SAC '95: Proceedings of the 1995 ACM symposium on Applied computing*, pages 345–350, New York, NY, USA, 1995. ACM Press.
67. 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.
68. 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.
69. V. Vapnik. *Statistical Learning Theory.* Wiley, 1998.
70. Ivan Voutchkov and A.J. Keane. Multiobjective Optimization using Surrogates. In I.C. Parmee, editor, *Adaptive Computing in Design and Manufacture. Proceedings of the Seventh International Conference*, pages 167–175, Bristol, UK, April 2006. The Institute for People-centered Computation (IP-CC).
71. C. K. I. Williams and C. E. Rasmussen. Gaussian processes for regression. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo., editors, *Advances in Neural Information Processing Systems 8.* MIT Press, 1996.

72. Benjamin Wilson, David J. Cappelleri, Timothy W. Simpson, and Mary I. Frecker. Efficient pareto frontier exploration using surrogate approximations. In *Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA,, September 2000.
73. K.S. Won and T. Ray. Performance of kriging and cokriging based surrogate models within the unified framework for surrogate assisted optimization. In *Congress on Evolutionary Computation*, pages 1577–1585. IEEE, 2004.
74. X. Zheng, B. A. Julstrom, and W. Cheng. Design of vector quantization codebooks using an genetic algorithm. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC 97)*, pages 525–530, 1997.
75. Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.