A Graphical Model for Evolutionary Optimization

SUBMISSION COPY — DO NOT DISTRIBUTE

Christopher K. Monson c@cs.byu.edu
Department of Computer Science, Brigham Young University, Provo, Utah 84602, USA

Kevin D. Seppi k@byu.edu
Department of Computer Science, Brigham Young University, Provo, Utah 84602, USA

Abstract

We present a statistical model of empirical optimization that admits the creation of algorithms with explicit and intuitively defined desiderata. Because No Free Lunch theorems dictate that no optimization algorithm can be considered more efficient than any other when considering all possible functions, the desired function class plays a prominent role in the model. In particular, this provides a direct way to answer the traditionally difficult question of what algorithm is best matched to a particular class of functions. Among the benefits of the model are the ability to specify the function class in a straightforward manner, a natural way to specify noisy or dynamic functions, and a new source of insight into No Free Lunch theorems for optimization.

Keywords

Genetic algorithms, Mathematical optimization models, Evolutionary optimization.

1 Introduction

Empirical continuous function optimization is the problem of searching for a global minimum by sampling the function at discrete points in its domain. Evolutionary algorithms have enjoyed popularity in this setting in recent years, including such approaches as Genetic Algorithms (GAs) (Holland, 1975; Goldberg, 1989), Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995), Estimation of Distribution Algorithms (EDAs) (Larrañaga and Lozano, 2001), and others.

While evolutionary algorithms are frequently used as black-box optimizers, No Free Lunch (NFL) theorems indicate that none of them can efficiently optimize the set of all possible functions (Macready and Wolpert, 1996; Wolpert and Macready, 1997). While these theorems were developed and proved for discrete problems, it is widely believed that they also apply to continuous applications. This poses an interesting problem for researchers and algorithm designers: if any optimization algorithm is known *a priori* to be ineffective on a large class of functions, on which class will it be effective? Ideally, an algorithm efficiently finds the optimum of a useful class of functions, allowing its performance on any other functions to be ignored.

Finding a pairing between algorithms and function classes is a difficult task in general and is usually tackled with the scientific method: a pairing hypothesis is stated, benchmark experiments are performed, and the hypothesis is either refuted or it is not. The benchmarks are commonly chosen to represent various high-level function characteristics: continuity, deceptiveness, smoothness, symmetry, etc. As is true in other scientific endeavors based on observation and reasoning, this approach is cumbersome and can only refute or support a hypothesis, not prove it. Every optimization algorithm is suitable for use on a particular class of functions, but this class is usually hidden and must be teased out of the algorithm's machinery with extensive testing (Whitley and Watson, 2006).

Estimation of Distribution Algorithms (EDAs) represent a promising direction in evolutionary research (Syswerda, 1993; Baluja, 1994; Baluja and Caruana, 1995; de Bonet et al., 1997; Pelikan et al., 1999; Pelikan and Goldberg, 2000a,b; Larrañaga and Lozano, 2001; Pelikan et al., 2002), in some cases allowing much to be learned about the behavior of the algorithm on specific function classes by analysis (González et al., 2002; Grahl et al., 2005). They represent a compelling way of thinking about optimization and an interesting first step toward better function class determination.

We propose an approach that is even more explicit about the connection between function class and algorithm design: the Graphical Model for Evolutionary Optimization (GMEO), a model that uses a straightforward declaration of the function class to produce a maximally efficient algorithm for it. While inference within this model functions in ways similar to EDAs, inferring a distribution over a population of solutions in order to create another, it differs from them in its fundamentally declarative nature; it effectively changes algorithm design into function class specification.

The complete model is created and explained by first considering the simpler problem of function sampling, then adding new constructs in a sensible manner until the full evolutionary model is obtained. Throughout this work, examples are strategically placed to facilitate understanding of how the model is created, how it is used for optimization, and the theoretical significance of its characteristics.

2 Bayesian Models of Sampling and Optimization

To facilitate unity in exposition and understanding, Section 2.1 is a brief introduction to Bayesian inference within graphical models. Those familiar with the material can safely skip to Section 2.2.

2.1 Bayesian Models

Bayes' Law states that for continuous random variables x and y:

$$\rho(x \mid y) = \frac{\rho(y \mid x)\rho(x)}{\int \rho(y \mid x)\rho(x)\,dx} \quad . \tag{1}$$

Typically, x is considered to represent some state or fact about the world, and y is a resulting observation, e.g., $x =$ **Happy** affects whether $y =$ **Smiling**. Bayes' Law is used to compute the inverse relationship, allowing conclusions to be drawn about the state x when observing only y. To do this, it is required that the *prior distribution* $\rho(x)$ and the *sampling distribution* or *likelihood* $\rho(y \mid x)$ be specified; the former represents what is believed about x without any observations, and the latter represents a belief about how the state of the world impacts those observations.

While it is possible to reason about these probabilities directly in the given notation, it is useful to depict the variables and relationships in a *graphical model*: a graph where nodes represent random variables and edges represent information dependencies. In a directed graphical model (or Bayesian Network), the edges typically represent causal relationships, and every node in the model has an associated distribution: conditional if the node has parents, unconditional if not.

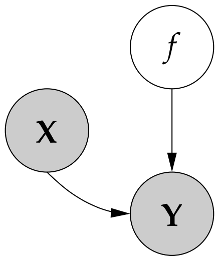


Figure 1: Graphical Sampling Model

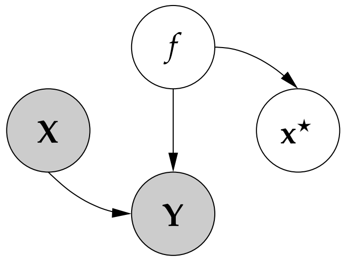


Figure 2: Graphical Optimization Model

These models can be quite complex, but are typically acyclic when intended to be used for Bayesian inference. In this case, Bayes' Law and other simple inference rules can be combined to compute a distribution over any query variable given information about any number of evidence variables in the model: $\rho(\text{query} \mid \text{evidence})$. Any variables that do not represent queries or evidence are integrated away in the inference process. This is a powerful tool for determining information about hidden state from available observations.

2.2 Optimization

The use of graphical models for optimization is motivated by their fundamentally declarative nature. The algorithm for obtaining information about the world given a set of observations is fixed: Bayesian inference; all that is required is a causal model of how the true state of the world affects observable phenomena. As the assumptions change, so will the results of inference, but the process used to obtain those results is unchanged. This is an appealing characteristic of graphical models in the context of function optimization; given a statement of the function class, it should be possible to use a fixed algorithm to perform optimization within it.

The creation of such a model begins with consideration of the most fundamental process in continuous empirical optimization: *sampling*, or querying the function for outputs at various locations. These samples are then used to reason about the location of the global optimum. This process operates on three essential random variables:

- f: a function,
- $\mathbf{X} = (\mathbf{x}_1 \cdots \mathbf{x}_{N_\mathbf{X}})$: a matrix of $N_\mathbf{X}$ location vectors to be sampled, and
- $\mathbf{Y} = (\mathbf{y}_1 \cdots \mathbf{y}_{N_\mathbf{X}})$: all information obtained from f at those locations[1].

Additionally, the unknown function that allows sampling and whose optimum is sought is denoted $f^\star$ and is called the *true function*. Ideally $f^\star$ or some sufficiently similar function will be drawn from a distribution over f. The relationship between these elements is depicted in the causal Graphical Sampling Model (GSM) in Figure 1, a graphical model that depicts f and $\mathbf{X}$ as influencing $\mathbf{Y}$. Each relationship is specified as a probability density, lending its expression both flexibility and precision.

Shaded nodes represent variables whose values are known or observable. The rest are assumed to be hidden. Although it must at least be possible to evaluate the true function $f^\star$ at an arbitrary point, a fact that often (but not always) implies the

[1]Typically the only information available at a given point is a scalar representing the function's value. This construct, however, is not limited to this viewpoint: $\mathbf{Y}$ may just as easily represent a set of gradient vectors as value scalars.

availability of its complete definition, the function as a whole is considered "hidden"; among all representations allowed for f, the one that best fits the *characteristics of interest* (such as the location of its global optimum) is unknown.

For each variable in the GSM, a corresponding distribution or probability density is needed, conditional for nodes with parents and unconditional for those without:

- $\rho(f)$: the prior probability density over possible functions,
- $\rho(\mathbf{X})$: the prior probability density over positions to be sampled, and
- $\rho(\mathbf{Y} \mid f, \mathbf{X})$: the conditional distribution over values obtained from sampling.

When supplied with such definitions, Bayesian inference may be applied to the model to calculate distributions over any of its unknowns. It is possible, for example, to use the model to compute $\rho(f \mid \mathbf{X}, \mathbf{Y})$, a distribution over the functions that are consistent with observed samples. This distribution encapsulates the goal of function approximation, e.g., learning a distribution over the weights in an artificial neural network (De Freitas et al., 2000).

While function *approximation* is interesting, function *optimization* requires not that $f^\star$ be wholly discovered or approximated, but that the space of possible functions be narrowed sufficiently to discover the location of its global optimum; many possible functions may share this important characteristic with $f^\star$, and so long as those that do not are eliminated from consideration, optimization is successful.

The location of the optimum, denoted $\mathbf{x}^\star$ in Figure 2 (hereafter referred to as the Graphical Optimization Model or GOM), is a characteristic of the function and is therefore influenced only by that node. The distribution $\rho(\mathbf{x}^\star \mid f)$ models the relationship between what is known about the function and the location of its optimum. As long as this *goal distribution* $\rho(\mathbf{x}^\star \mid f)$ is defined, it is possible to infer $\rho(\mathbf{x}^\star \mid \mathbf{X}, \mathbf{Y})$, a distribution that represents the ultimate goal of optimization: using samples to obtain information about the global optimum.

Defining the goal of optimization in this way effectively expands the definition of success in this context: traditionally, successful optimization entails finding the single point that is most likely to produce the globally minimal value; this new definition captures much more of the tacit intent of optimization, allowing not only for the determination of a likely minimum, but also of the degree of confidence that should be placed in it and the extent to which the data may support other locations as possible minima (e.g., in functions with multiple equally fit minima). If a single point is desired, it is easily obtained from $\rho(\mathbf{x}^\star \mid \mathbf{X}, \mathbf{Y})$: for example, one may sample the distribution to obtain a single point, or take the distribution's mode to be the optimum.

In summary, the following variables and distributions are specified in the model:

- f: a function,
- $\mathbf{X}$: a matrix of location vectors,
- $\mathbf{Y}$: a matrix of value vectors,
- $\mathbf{x}^\star$: a global optimum,
- $f^\star$: the true function from which samples are to be taken,
- $\rho(\mathbf{X})$: a distribution over sample locations (generally a supplied set of points),
- $\rho(\mathbf{Y} \mid f, \mathbf{X})$: the sampling distribution, and
- $\rho(\mathbf{x}^\star \mid f)$: the goal distribution.

The ultimate goal of optimization is to find the distribution $\rho(\mathbf{x}^\star \mid \mathbf{X}, \mathbf{Y})$, an indication of where the optimum is likely to be given the information obtained from sampling $f^\star$.

2.3 Related Work

A Bayesian approach to optimization is not unique to this work: Bayesian sampling and global optimization techniques surveyed by Stuckman and Easom (1992) and Törn and Žilinskas (1989) also make use of Bayesian inference in optimization. Mockus (1989); Mockus et al. (1996) and Törn and Žilinskas (1989) also contribute important ideas to these methods. Many existing techniques rely on Gaussian Processes (see the tutorial by MacKay (1998)), including kriging (Krige, 1951). Particle Swarm Optimization has also been adapted to create and use distributions inferred through a Bayesian model (Monson and Seppi, 2004, 2005).

There are also algorithms which make use of a function class in the design of the algorithm, for example, Muhlenbein's work on the genetic FDA and UMDA algorithms for additively decomposable functions (Muhlenbein and Mahnig, 1999) and Sequential Quadratic Programming (Boggs and Tolle, 1995) . Meta models can also be used to improve an algorithm for a specified function class by reducing the need for expensive real function evaluations (Jones et al., 1998; Sasena, 2002; Emmerich et al., 2002).

Also related to this work is the field of function approximation. Linear, quadratic, Bayesian and other forms of regression are all related to this work in the sense that data is used to draw conclusions about a function. In this work, however, the focus is on the discovery of the minimum, not the approximation of the entire function. In some cases a very good estimate for the location of the minimum can be obtained while leaving much about the rest of the function undiscovered.

As mentioned briefly in the introduction, EDAs also generate distributions over likely minima to do their work, creating them using predefined representations and acquisition heuristics over the fittest members of the population. While similar in spirit, EDAs and the GMEO have at least one critical difference: EDAs rely on a *fixed* and usually *implicit* choice of function class, encoded in the distribution representation and the algorithm used to obtain it; the Graphical Model for Evolutionary Optimization relies on a *exchangeable* and *explicit* choice of function class, encoded in the various distributions that describe the model. This distinction will be justified and discussed in greater detail throughout this work.

3 Inference in the Graphical Optimization Model

Optimization in the GOM is achieved by inferring $\rho(\mathbf{x}^\star \,|\, \mathbf{X}, \mathbf{Y})$, a distribution over likely locations for the global minimum conditioned on information obtained from sampling. Inference of $\rho(\mathbf{x}^\star \,|\, \mathbf{X}, \mathbf{Y})$ in the Bayesian optimization model is accomplished using the Chain Rule and Bayes' Law (Russel and Norvig, 2003):

$$\rho(\mathbf{x}^\star \,|\, \mathbf{X}, \mathbf{Y}) = \int \rho(\mathbf{x}^\star \,|\, f)\rho(f \,|\, \mathbf{X}, \mathbf{Y})\, df \propto \int \rho(\mathbf{x}^\star \,|\, f)\rho(\mathbf{Y} \,|\, f, \mathbf{X})\rho(f)\, df \quad . \tag{2}$$

As stated previously, the *function prior* $\rho(f)$, *goal distribution* $\rho(\mathbf{x}^\star \,|\, f)$, and *sampling distribution* $\rho(\mathbf{Y} \,|\, f, \mathbf{X})$ must be specified before inference of $\rho(\mathbf{x}^\star \,|\, \mathbf{X}, \mathbf{Y})$ can proceed.

3.1 Practical Inference

It is not immediately clear how to specify a parametric prior over the space of all continuous functions that is both simple and useful, ruling out many analytical methods of inference, e.g., methods requiring Gaussian distributions. Numerical inference methods such as Particle Filtering must be used instead (Russel and Norvig, 2003). In spite of their inefficiencies and the existence of methods that are in many ways superior, particle filters are easy to implement and describe, making them the method of choice in this work.

Particle filters sample "particles" from distributions over all nodes without parents, then push the particles forward through the links in the graph structure by sampling from the conditional child distributions. If an observed node is encountered, the observed value is used and the likelihood of the particle (initially 1) is multiplied by the likelihood of the observation given its parents. Since this process is ordered by the acyclic graph structure, all conditions will be fixed either by sampling or observation when the likelihood of a child is computed. The resulting weighted sample will approximate the posterior distribution and can be used directly or re-sampled proportionally to the weights. The only parameter to this algorithm, other than the graph itself, is the number of particles $N_{\mathbf{P}}$. As is the case with most sampling algorithms, more particles provide a better approximation.

In many cases, including in the examples to follow, sampling from distributions in the model is simple. In cases where very complex distributions are used, however, there may not be efficient standard libraries available for sampling. Sampling is still possible in these latter cases by making use of more general algorithms such as Metropolis-Hastings (Metropolis et al., 1953; Hastings, 1970), provided that more complexity in both implementation and computation is tolerated.

In the context of the GOM, the particle filtering process consists roughly of the following steps, described in greater detail in Algorithm 1:

- Select candidate solutions $\mathbf{X}$,
- Evaluate the true function $f^\star$ at each position, producing a matrix of cost values $\mathbf{Y}$,
- Draw a set of particles from $\rho(f)$, each of which represents a function, and
- Calculate the likelihood of each particle (function) with respect to $\mathbf{X}$ and $\mathbf{Y}$.
- Generate $\rho(\mathbf{x}^\star \mid \mathbf{X}, \mathbf{Y})$ from these likelihoods and $\rho(\mathbf{x}^\star \mid f)$.

Algorithm 1 Particle filter for the GOM

```
1: # Create a population of $N_{\mathbf{X}}$ candidate locations and obtain values
2: $\mathbf{X} = (\mathbf{x}_1 \ldots \mathbf{x}_{N_{\mathbf{X}}})^\top$, with $\mathbf{x}_i \sim \rho(\mathbf{x}^\star \mid f_i)$ and $f_i \sim \rho(f)$
3: $\mathbf{Y} = (\mathbf{y}_1 \ldots \mathbf{y}_{N_{\mathbf{X}}})^\top$, with $\mathbf{y}_i = f^\star(\mathbf{x}_i)$
4: # Create $N_{\mathbf{P}}$ particles and calculate the normalized likelihoods, forming an empirical $\rho(f \mid \mathbf{X}, \mathbf{Y})$
5: $\mathbf{P} = (f_1 \ldots f_{N_{\mathbf{X}}})^\top$, with $f_i \sim \rho(f)$
6: $\mathbf{L} = \left(\mathcal{L}(f_1 \mid \mathbf{X}, \mathbf{Y}) \ldots \mathcal{L}(f_{N_{\mathbf{P}}} \mid \mathbf{X}, \mathbf{Y})\right)^\top / \sum_{i=1}^{N_{\mathbf{P}}} \mathcal{L}(f_i \mid \mathbf{X}, \mathbf{Y})$ with $\mathcal{L}(f_i \mid \mathbf{X}, \mathbf{Y}) = \prod_{j=1}^{N_{\mathbf{X}}} \rho(\mathbf{y}_j \mid f_i, \mathbf{x}_j)$
7: # Generate an empirical $\rho(\mathbf{x}^\star \mid \mathbf{X}, \mathbf{Y})$ by sampling from $\rho(f \mid \mathbf{X}, \mathbf{Y})$ (represented as $\mathbf{P}, \mathbf{L}$) and $\rho(\mathbf{x}^\star \mid f)$
8: $\mathbf{X}^\star = (\mathbf{x}_1^\star \ldots \mathbf{x}_{N_{\mathbf{X}}}^\star)$, with $\mathbf{x}_i^\star \sim \rho(\mathbf{x}^\star \mid f_i)$ and $f_i \sim \rho(f \mid \mathbf{X}, \mathbf{Y})$
```

Several things are of note about Algorithm 1. First, the distributions $\rho(\mathbf{x}^\star \mid f)$ and $\rho(f)$ are used as surrogates for $\rho(\mathbf{X})$, but they need not be; if sample locations are provided, they should simply be used. Second, several steps of the algorithm require sampling from an empirical distribution, defined either as a bag of values or value-likelihood pairs. Such sampling is a solved problem (Doucet, 1998) and algorithms for accomplishing it are not supplied here. Finally, the algorithm makes the distinction between f and $f^\star$ clear. The true function $f^\star$ is the unknown real-world function that is the target of optimization; sampling it represents (potentially costly) function evaluations. In contrast, the variable f represents the algorithm's internal representation of its knowledge and assumptions about the true function. Ideally, sampling from a distribution over f will generate a useful approximation to $f^{\star}$.

The algorithm is not complete without the specification of the distributions in the model. The issues of choosing the distributions $\rho(f)$, $\rho(\mathbf{Y}\,|\,f,\mathbf{X})$, and $\rho(\mathbf{x}^{\star}\,|\,f)$ are most easily understood by example and are addressed in upcoming sections.

4 Example: Simple Parametric Class

In order to facilitate the necessary definition of a distribution over functions, the notation of lambda calculus will be employed. In brief, the notation $\lambda x, y.f(x,y)$ represents a function that takes parameters x and y and returns the value $f(x,y)$. The dot separates the parameters from the description of how the result is obtained. A more detailed treatment of lambda calculus is available in many texts on discrete mathematics, e.g., (Grassmann and Tremblay, 1996).

4.1 Defining a Basic Class

Armed with appropriate notation, it is now possible to consider the problem of specifying $\rho(f)$, $\rho(\mathbf{Y}\,|\,f,\mathbf{X})$, and $\rho(\mathbf{x}^{\star}\,|\,f)$. These specifications begin with a basic definition of the function class, which in nearly every example in this work will be related to the cone $\lambda\mathbf{x}.\|\mathbf{x}-\mathbf{c}\|_2$; the parameter $\mathbf{c}$ represents the location of its center or tip.

This representation admits a simple specification for $\rho(f)$, defined in two parts:

$$\rho(f) = \begin{cases} \rho(\mathbf{c}) & \text{if } f \in \{\lambda\mathbf{x}.g(\mathbf{x},\mathbf{c}) \mid \forall\mathbf{x}\in\mathbb{R}^D.g(\mathbf{x},\mathbf{c}) = \|\mathbf{x}-\mathbf{c}\|_2\} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$\rho(\mathbf{c}) = N_{\mathbf{0},\Sigma_{\mathbf{c}}}(\mathbf{c}) \ . \tag{4}$$

The first part narrows the set of functions that receive non-zero weight to those that match the provided definition of a cone, and the second defines a distribution over the unknown parameter $\mathbf{c}$. Note that $N_{\cdot,\cdot}$ is a D-dimensional multivariate Gaussian density function parameterized on mean and covariance, respectively.

4.2 Similarity-Based Boundaries

Although (3) and (4) form an infinitely large class of cones, the class is highly constrained. Allowances can be made for functions that are close to but not precisely contained within the given definition by way of an appropriate sampling distribution:

$$\rho(\mathbf{y}\,|\,f,\mathbf{x}) = N_{f(\mathbf{x}),\sigma_{\mathbf{y}}}(\mathbf{y}) \tag{5}$$

$$\rho(\mathbf{Y}\,|\,f,\mathbf{X}) = \prod_{i=1}^{N_{\mathbf{X}}} \rho(\mathbf{y}_i\,|\,f,\mathbf{x}_i) \ . \tag{6}$$

This distribution modifies the definition of a cone to include a function f if the difference $f(\mathbf{x})-\mathbf{y}$ is normally distributed with zero mean and $\sigma_{\mathbf{y}}$ standard deviation. This change allows functions that have noisy output or that vary from those defined in the class to be given a nonzero likelihood. In fact, the use of a distribution with infinite extent in (5) allows *any* function to have nonzero (but potentially vanishingly small) likelihood. This function class can therefore be considered to give at least a small amount of weight to every function, with particular emphasis on cones as defined in (3) and (4).

It is worth noting that if $\rho(\mathbf{Y}\,|\,f,\mathbf{X})$ did not allow for uncertainty in its specification, it would be impossible to obtain a useful empirical distribution over $\mathbf{x}^{\star}$ using likelihood weighting. In a continuous setting such as this, the likelihood of sampling the exact set of parameters which will produce perfectly consistent results is zero, and therefore some amount of uncertainty must be encoded in $\rho(\mathbf{Y} \mid f, \mathbf{X})$: an equivalence relation such as $\mathbf{y} = f(\mathbf{x})$ does not provide sufficient approximate information.

The function class is now fully defined and will be referred to as the Gaussian Value Cone (GVC) class. Its definition has followed an important and useful pattern: first, a precise definition of the desired class is given in $\rho(f)$, in this case a class of cones; second, a notion of *similarity* is precisely defined in $\rho(\mathbf{Y} \mid f, \mathbf{X})$. In more colloquial terms, the resulting class may be described as containing anything that is "similar to a cone", where $\rho(f)$ defines the meaning of "a cone", and $\rho(\mathbf{Y} \mid f, \mathbf{X})$ defines the meaning of "similar to". When creating new classes, it is often useful to think of the desired class in terms of such human-friendly language and then work backwards to generate precise and meaningful mathematical definitions of $\rho(f)$ and $\rho(\mathbf{Y} \mid f, \mathbf{X})$.

4.3 Finding the Minimum

The distributions $\rho(f)$ and $\rho(\mathbf{Y} \mid f, \mathbf{X})$ are now defined, completing the necessary specifications for the GSM. Using this model it is possible to infer a posterior distribution $\rho(f \mid \mathbf{X}, \mathbf{Y})$ that combines prior belief with information gained from sampling to form a new belief about $f^\star$, learned from observed data.

Learning $f^\star$ is interesting, but *optimization* of $f^\star$ is the real goal. Therefore, the distribution $\rho(\mathbf{x}^\star \mid \mathbf{X}, \mathbf{Y})$ is sought, requiring the prior specification of one more distribution: $\rho(\mathbf{x}^\star \mid f)$. This distribution models the relationship between a function and its global minimum (Figure 2), and represents an embedded optimization problem: sampling from this distribution requires the minimization of a function drawn from $\rho(f)$.

In general this embedded optimization problem is no easier to solve than the original problem, but with a carefully crafted representation of $\rho(f)$ it can be made trivial. In this particular example, all functions drawn from $\rho(f)$ are easily minimized because drawing a function f is equivalent to drawing its parameter $\mathbf{c}$, and the minimum will always be precisely located at $\mathbf{x}^\star = \mathbf{c}$. This is expressed using Dirac's delta distribution, which places all of its density at the origin:

$$\rho(\mathbf{x}^\star \mid f) = \delta(\mathbf{x}^\star - \mathbf{c}) \ . \tag{7}$$

4.4 Solving the Model

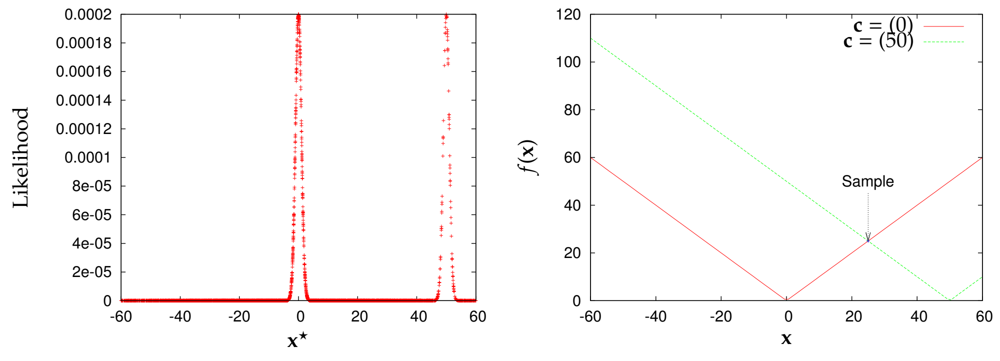
Together, $\rho(f)$, $\rho(\mathbf{Y} \mid f, \mathbf{X})$, and $\rho(\mathbf{x}^\star \mid f)$ complete the specification of all relationships in Figure 2, assuming that $N_\mathbf{X}$ sample locations are provided in $\mathbf{X}$. This information is sufficient to infer $\rho(\mathbf{x}^\star \mid \mathbf{X}, \mathbf{Y})$, the goal of optimization.

Of course, the various distribution and filtering parameters must also be specified. To lend concreteness to the discussion, consider the following settings:

$$\begin{aligned} D &= 1 & f^\star(\mathbf{x}) &= \|\mathbf{x} - 50\|_2 \\ N_\mathbf{P} &= 5000 & \Sigma_\mathbf{c} &= \left((50/3)^2\right) \\ N_\mathbf{X} &= 1 & \sigma_\mathbf{y} &= 1 \ . \end{aligned}$$

In other words, 5000 particles are used, a single observation is taken from $f^\star$, and the true function is a one-dimensional cone centered at $\mathbf{x}^\star = 50$.

Algorithm 1 selects the initial population by sampling $N_\mathbf{X}$ functions from $\rho(f)$, then sampling a location from $\rho(\mathbf{x}^\star \mid f)$ for each of them (Line 2). This is only one approach for generating locations $\mathbf{X}$; it could easily be done another way. In this example, exactly



(a) Filtered posterior likelihoods

(b) Possible functions

Figure 3: Cone prior with a true cone function

one sample is produced since $N_\mathbf{X} = 1$. Let the result of sampling from $\rho(\mathbf{c})$ be $\mathbf{c} = 25$; sampling from the trivial $\rho(\mathbf{x}^\star \,|\, f)$ therefore produces the (singleton) population of candidate solutions $\mathbf{X} = (25)$.

Line 3 of the algorithm describes the process of evaluating the true function $f^\star$ at all candidate locations, which in this case produces $\mathbf{Y} = (f^\star(25)) = (25)$. With a population of candidate solutions and their corresponding true values, it is now possible to perform inference; the observable variables are now assigned, so the particle filter may be applied.

Line 5 describes the process of obtaining particles, producing a vector $\mathbf{P}$ of 5000 cones, each centered at a location drawn from $\rho(\mathbf{c})$. Finally, line 6 gives a formula for calculating the normalized likelihood of each particle from $\rho(\mathbf{Y} \,|\, f, \mathbf{X})$, producing an empirical approximation to the posterior $\rho(\mathbf{x}^\star \,|\, \mathbf{X}, \mathbf{Y})$ as shown in Figure 3(a).

With exactly one sample at the location 25, the two cones centered at 0 and 50 are maximally consistent with the data, as shown in Figure 3(b). The peak in Figure 3(a) corresponding to the true function's minimum at 50 is more sparsely represented than the wrong peak at 0 because $\rho(\mathbf{c})$ favors the origin, but the posterior $\rho(\mathbf{x}^\star \,|\, \mathbf{X}, \mathbf{Y})$ still provides good likelihoods for the correct minimum.

Had the population of samples contained 2 distinct locations, there would have been exactly one peak in the graph, since an additional evaluation would serve to differentiate between the two peaks. Significantly, and in contrast to most evolutionary methods, the sample point is far from the optimum but still provides information about its location; it is possible to infer the minimum without ever sampling it.

4.5 Discussion

This example provides a good context for the discussion of problem complexity and what occurs when the function class does not reflect reality. A similar section will appear after each of the examples given in this work, spreading out the discussion of important concepts while placing them close to examples that provide critical context.

4.5.1 Problem Complexity in the Sampling Model

Every optimization problem has a particular inherent complexity, and this example highlights the facets of that complexity and how it might be calculated. Ignoring the complexity of particle filtering (it is an implementation detail), there are two major sources of complexity in the model: *sample complexity* and *computational complexity*.

Sample complexity is a notion researched extensively in the machine learning community, describing the number of samples of $f^\star$ required to obtain good information about it. This is relevant in the context of the Bayesian optimization model because it recasts part of the optimization problem as a learning problem; the resulting approximation need not be perfect everywhere, but must contain enough information to pinpoint the global minimum. In this particular example, the space of learning hypotheses is the set of all possible centers for the cone defined in (3).

It is clear that the one-dimensional GVC can be optimized with at least two distinct samples: one sample admits two cones, and a second unique sample serves to differentiate them. The sample complexity, then, is bounded from above by 2. Establishing such upper bounds is in general an unsolved problem, but many function class representations (learning hypotheses) admit discovery of bounded sample complexity (Mitchell, 1997; Christianini and Shawe-Taylor, 2000). Specifically, if the representation's "Fat-Shattering Dimension" is discoverable and finite, its sample complexity may be bounded (Valiant, 1984; Bartlett et al., 1996; Kearns and Schapire, 1994). This is important because it may provide a natural worst-case stopping criterion for optimization within the GOM. Although lower bounds are often available, they are not likely to be correct in the GOM because optimization often requires less information than approximation.

In addition to the sample complexity, the computational complexity of a problem is evident in the embedded optimization problem encoded in $\rho(\mathbf{x}^\star \mid f)$. In the GVC it is computationally trivial to draw samples from $\rho(\mathbf{x}^\star \mid f)$, but in general the computation can be much more complex. Consider, for example, the prior

$$\rho(f) = \begin{cases} \rho(\mathbf{c}) & \text{if } f \in \{\lambda\mathbf{x}.g(\mathbf{x},\mathbf{c}) \mid \forall\mathbf{x} \in \mathbb{R}.g(\mathbf{x},\mathbf{c}) = \text{ANN}(\mathbf{x},\mathbf{c})\} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where ANN is a multilayer feed-forward artificial neural network with some predefined structure. The parameter vector $\mathbf{c}$ in this case represents the weights on the model edges. If the model is large enough, it may be capable of immense representational flexibility, making it potentially difficult to minimize its output. This difficulty will be evident in a sampling algorithm for $\rho(\mathbf{x}^\star \mid f)$, thus hiding another optimization problem with its own associated complexity.

While in many cases this is unacceptable and a representation more amenable to optimization should be used, it can represent a reasonable trade-off; in many real-world scenarios, e.g., tuning the parameters of an oil refinery to produce optimal output, computer time is inexpensive compared to sampling $f^\star$, making complex off-line optimization of an ANN acceptable.

4.5.2 Wrong Priors

Any optimizer can be deceived, including inference within the GOM; No Free Lunch requires it. For example, if $f^\star$ is a parabola centered at the origin ($f^\star(\mathbf{x}) = \|\mathbf{x}\|_2^2$), and therefore not in the GVC class, the inferred $\rho(\mathbf{x}^\star \mid \mathbf{X}, \mathbf{Y})$ will give most of its likelihood to incorrect values as shown in Figure 4(a).

This behavior is explained by Figure 4(b), which shows that moving *away* from the sample, at least until a function is found which crosses through $(\mathbf{x}, \mathbf{y})$, causes the likelihood to increase. In this particular case, even adding more sample points fails to correct the distribution unless those points are generated very near the origin; as samples move further away, the value disparity between the parabola and its corresponding

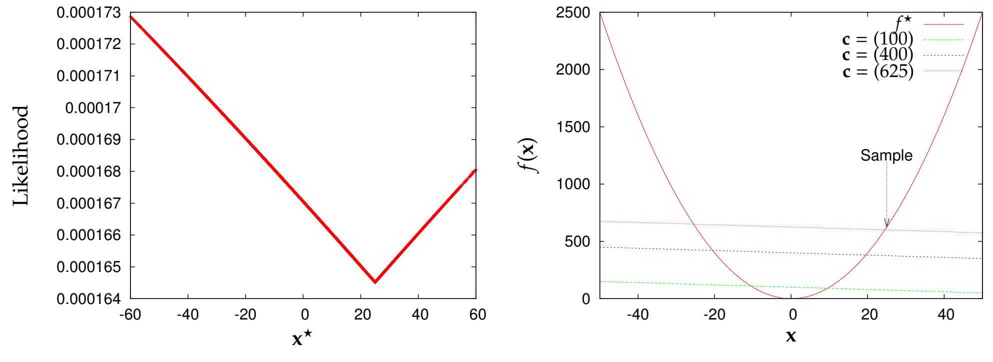


Figure 4: Conic prior with a parabolic $f^\star$

cone increases far beyond the scope allowed by $\rho(\mathbf{Y}\,|\,f,\mathbf{X})$. Even though GVC describes functions that are "similar to a cone", the definition of this similarity places vanishingly small likelihood on parabolas.

5 Example: Simple Membership Class

Departing from cones for the moment, perhaps the simplest and most general way to specify $\rho(f)$ is as a set of functions with associated fixed probabilities:

$$\rho(f) = \begin{cases} p_1 & \text{if } f = \lambda\mathbf{x}.f_1(\mathbf{x}) \\ p_2 & \text{if } f = \lambda\mathbf{x}.f_2(\mathbf{x}) \\ \vdots & \\ p_N & \text{if } f = \lambda\mathbf{x}.f_N(\mathbf{x}) \\ p_0 & \text{otherwise} \end{cases} \quad . \tag{9}$$

This prior makes the function class very clear: $f^\star$ is assumed to be one of those listed. As an example, the following is a prior over a set of common benchmark functions:

$$\rho(f) = \begin{cases} \frac{1}{3} & \text{if } f = \lambda\mathbf{x}.f_1(\mathbf{x}) \text{ where } f_1(\mathbf{x}) = \|\mathbf{x}\|_2^2 \\ \frac{1}{3} & \text{if } f = \lambda\mathbf{x}.f_2(\mathbf{x}) \text{ where } f_2(\mathbf{x}) = \|\mathbf{x}\|_2^2 + 10\sum_{i=1}^{D} 1 - \cos(2\pi x_i) \\ \frac{1}{3} & \text{if } f = \lambda\mathbf{x}.f_3(\mathbf{x}) \text{ where } f_3(\mathbf{x}) = 20 + e - 20\exp\left(\frac{-\|\mathbf{x}\|_2}{5\sqrt{D}}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i\right) \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where $D = 1$ is the dimensionality, $\sum_{i=0}^{N} p_i = 1$, and the functions are the well-known Sphere, Rastrigin, and Ackley, respectively. Because the minimum of each of these functions is at the origin, the optimization distribution is unconditionally

$$\rho(\mathbf{x}^\star\,|\,f) = \rho(\mathbf{x}^\star) = \delta(\mathbf{x}^\star - \mathbf{0}) \quad . \tag{11}$$

Additionally, $\rho(\mathbf{Y}\,|\,f,\mathbf{X})$ can dictate that values must precisely match function output

$$\rho(\mathbf{Y}\,|\,f,\mathbf{X}) = \prod_i \delta(\mathbf{y}_i - f(\mathbf{x}_i)) \quad . \tag{12}$$

A natural language description of this class might be "precisely one of Sphere, Ackley, or Rastrigin," referred to hereafter as the Precise Benchmark (PB) class.

It is relatively easy to see that Algorithm 1 will only generate candidate solutions at the origin, since all functions sampled from $\rho(f)$ have their minimum there. Because of this, the algorithm will fail to distinguish between functions but will still find the global minimum if $f^\star$ is in the class. In this extreme example, *no function evaluations are needed* to find a correct $\rho(\mathbf{x}^\star \,|\, \mathbf{X}, \mathbf{Y})$, since (11) is an unconditional distribution.

As expected, if $f^\star$ is not in the PB class, it will only be minimized correctly if its optimum is at the origin. Otherwise, the likelihood weighting algorithm will fail because all particles will have zero probability. Because $\rho(\mathbf{x}^\star \,|\, f)$ is defined to be unconditional, the natural language description of "functions whose optimum is at the origin" is more appropriate than that given previously.

5.1 Discussion: No Free Lunch

This example, though trivial and unrealistic, serves to illustrate an important point: the inference algorithm will make maximal use of all information available to it. If it is, in fact, known that $f^\star$ is one of several that can be finitely listed, then the algorithm will select among them as quickly as their similarities will allow, thus efficiently providing the location of the global minimum. If, however, a larger function class is needed, it is possible that many samples will be required to find the global minimum.

It is instructive to consider an attempt at representing arbitrary functions within the class. In this case, a single function is an uncountable list of input-output pairs, and there are uncountably many such functions. Even if it were possible to assign equal non-zero weight to an uncountable number of functions, no amount of sampling would differentiate them or give any indication of the optimal point; a single sample produces a value that may have no relationship to any other value in any other location in the function's domain. The situation only worsens in the presence of non-deterministic sample values.

There is, in other words, no way to optimize an arbitrary function efficiently, other than to take a number of samples and assume that the lowest value sampled corresponds to the global minimum. This is, in essence, a restatement of No Free Lunch, but in stronger terms for the continuous setting. Where the discrete No Free Lunch theorems state that no algorithm can perform better than uniform random search on average (Wolpert and Macready, 1997), in the continuous setting there is no way to rationally define a notion of "uniform random search" because it is impossible to define a continuous uniform distribution with infinite extent. That such a statement can be made in a sensible way for continuous functions is one benefit of the GOM.

According to NFL, the efficient solution of any optimization problem requires the specification of a limited class of functions; ideally this class will place high likelihood on the function of interest and allow it to be differentiated from its peers. In the extreme example above, the class is sufficiently narrow that inference finds the minimum without ever taking a sample, making it impossible to improve upon for functions within the stated class.

Because NFL is inescapable, there is increasing interest in finding the function classes implicitly defined in existing algorithms. The determination of an algorithm–class pairing is typically approximated by way of benchmark testing or convergence proofs. While this does not generally accomplish its goals (Whitley and Watson, 2006), it has long been the only generally-applicable approach available. The GOM solves this problem by fixing the algorithm and requiring rigorous specification of intent in the concrete language of distributions; what would normally be tested is now specified, and the initial process design step is eliminated.

6 Example: Pairwise Relationship Class

The previous examples have defined classes that contain a limited set of functions: the Gaussian Value Cone (GVC) class only admits symmetric cone-shaped functions with a zero-valued tip, and the Precise Benchmark (PB) class (initially assuming a limited list of functions) contains only those functions whose minimum is located precisely at the origin. A number of techniques, especially in the case of GVC, might be employed to broaden the class and therefore make it more useful in situations where little is known about $f^\star$. One option might be to add a scale parameter to the definition of the cone, allowing the slope to change in arbitrary ways. Another might be an offset, allowing the value at its tip to be something other than 0. In the end, however, the resulting function class will still favor cones over any other shape, and the additional scale parameters will perhaps inordinately increase the complexity of optimization.

The form taken by the distributions $\rho(f)$ and $\rho(\mathbf{Y} \,|\, f, \mathbf{X})$ can be fairly arbitrary and allows for infinite flexibility in the concepts expressed. This example defines a class of functions that, while still describable as "similar to a cone", has far greater expressive power than GVC.

6.1 A New Notion of Similarity

As previously discussed, the notion of similarity is encoded in the sampling distribution $\rho(\mathbf{Y} \,|\, f, \mathbf{X})$. The obvious definition of similarity, one that looks only at raw output values, has been explored and demonstrated to be fairly limiting. The class that follows shares its prior $\rho(f)$ and goal distribution $\rho(\mathbf{x}^\star \,|\, f)$ with the GVC class, but uses a different sampling distribution to expand the class in a profound and natural way:

$$\rho(\mathbf{Y} \,|\, f, \mathbf{X}) \propto \prod_{i<j} p(\mathbf{y}_i, \mathbf{y}_j \,|\, f, \mathbf{x}_i, \mathbf{x}_j) \tag{13}$$

$$p(\mathbf{y}_1, \mathbf{y}_2 \,|\, f, \mathbf{x}_1, \mathbf{x}_2) = \begin{cases} \alpha \in (\frac{1}{2}, 1] & \text{if } \operatorname{sgn}(\mathbf{y}_1 - \mathbf{y}_2) = \operatorname{sgn}(f(\mathbf{x}_1) - f(\mathbf{x}_2)) \\ 1 - \alpha & \text{otherwise} \end{cases} . \tag{14}$$

The definition of $\rho(\mathbf{Y} \,|\, f, \mathbf{X})$ relies on pairwise relationships between samples, and the new class is called the Pairwise Relationship Cone (PRC).

6.2 The Resulting Class

Defining similarity over relationships instead of raw values serves to significantly broaden the function class: the PRC includes all symmetric convex functions. The sampling distribution $\rho(\mathbf{Y} \,|\, f, \mathbf{X})$ places higher density on functions whose pairwise relationships agree with those of a cone, and lower density on those that do not. In this sense the function class is actually broader than the set of symmetric convex functions; any function is included when, given an arbitrary set of samples, the *majority* of pairwise relationships agree with those of a correspondingly centered cone; given a pair of samples, the lower-valued sample will usually be closest to the global minimum. The strength of the required majority is encoded in the parameter α.

In the results that follow, $\alpha = 0.525$, assigning a probability of 0.475 to the possibility that the lowest point in a pair is *not* closest to the minimum, thus accommodating functions that have many local minima.

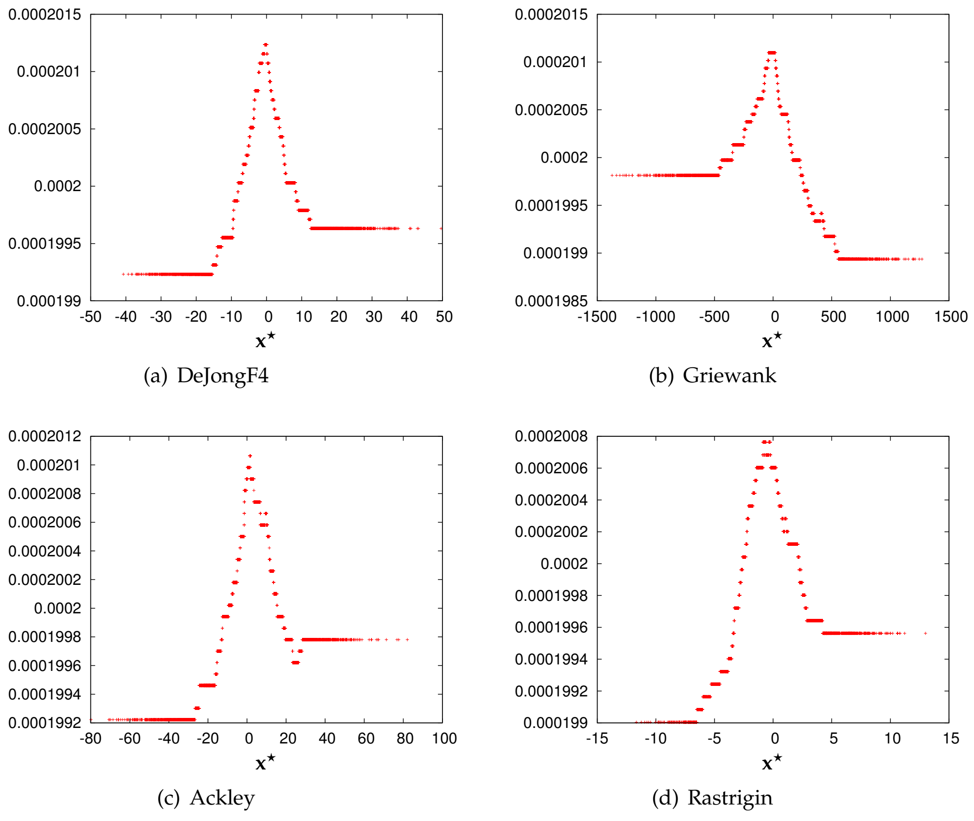


Figure 5: One-step likelihood on common one-dimensional benchmarks

6.3 Results

PRC requires $N_{\mathbf{X}} \geq 2$. Otherwise, the application of PRC is no different than the previous examples; Algorithm 1 still applies directly. The results of its application to several common benchmark functions in 1 dimension are shown in Figure 5. All graphs used $N_{\mathbf{X}} = 10$, $N_{\mathbf{P}} = 5000$, and $\alpha = 0.525$.

The behavior of inference in the PRC class is notable: it not only behaves rationally for DeJongF4, which is clearly convex, but also for non-convex functions such as Griewank, Ackley, and Rastrigin; PRC is tolerant of local minima, as predicted by the specification of α.

6.4 Discussion

Pairwise Relationship Cone serves to highlight several important characteristics of the GOM, especially in terms of how function classes may be defined in terms of conditional distributions in the model.

6.4.1 The Burden of Specification

The Graphical Optimization Model does not admit process-oriented algorithm design. Instead, it requires a complete, *a priori* specification of the function class, and it will infer all that can be known about the optimum given the specification and the data. While this solves the algorithm–class pairing problem by making class declaration a fundamental part of algorithm design, it appears to place an undue burden of specification on the optimization practitioner.

Optimization within the graphical model represents a fundamental trade-off: rather than design a process inspired by intuition, practitioners infuse the model with domain knowledge and leave the rest to inference. This has many benefits, but appears to cause problems when little is known about $f^\star$. In these settings, practitioners are accustomed to employing one or more algorithms from a toolbox of familiar approaches. They then either stop when an adequate algorithm is found or invent a new one that is better suited to their problem.

In the GOM, the same exploratory approach to optimization can be taken when given a function about which little is known. The difference is that the toolbox now contains function classes, not algorithms. Thus, a practitioner will use the *same* algorithm multiple times with different function classes, then either settle on the class that appears to match the given function or try to invent a new one.

In this sense, inference within the GOM is preferable to the traditional algorithm selection process for one very important reason: when settling on a function class that produces good results, it is now known that at least one class contains $f^\star$, thus providing *new information about the function*. When selecting an algorithm, all that is known is that its opaque machinery seems to work well for the problem at hand.

6.4.2 The Selection of Distributions

Any valid distribution can be used in the GOM. More complex distributions may affect the speed or accuracy of numerical inference within it, but the overall approach remains unaffected. Ideally, a set of distributions could be chosen that would admit a closed form solution; such a model would allow the posterior distribution to be found without using Algorithm 1 or any other numerical method. For example, in the widely studied case for Gaussian data with a Gaussian prior on the mean ($x \,|\, \theta \sim N_{\theta,\sigma}$ and $\theta \sim N_{\alpha,\tau}$) the distribution and parameters of the posterior can be derived analytically. Unfortunately none of the many well-known cases that can be computed analytically include a distribution over functions as required in the Graphical Optimization Model (DeGroot, 1970).

It is, of course, possible to use the GOM to model and optimize discrete-valued functions, but the differences between these and the continuous functions discussed here are great enough that this consideration is left for future work.

7 Introducing Evolution: The Graphical Model for Evolutionary Optimization

The discussion thus far has focused on a static Bayesian optimization model that makes use of a single population of samples. The success of evolutionary optimization methods, particularly EDAs, which make use of a distribution over possible minima to select new populations, suggests that something may be gained by using a similar technique here. The incorporation of distribution-based evolution into the GOM results in the more complete Graphical Model for Evolutionary Optimization (GMEO). This dynamic model is created by connecting multiple GOMs over time as shown in Figure 6.

This model introduces two new distributions: $\rho(f_t \,|\, f_{t-1})$ and $\rho(\mathbf{X}_t \,|\, \mathbf{x}^\star_{t-1})$. The former defines how functions change over time, and the latter defines how each new population is generated from previously obtained information. A closely related distribution, $\rho(\mathbf{X}_0)$, is used for generating the initial population. Though the previous examples have used

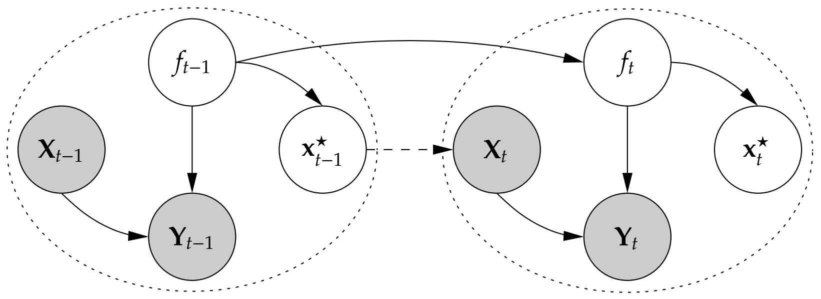


Figure 6: The Graphical Model for Evolutionary Optimization (GMEO)

$\rho(f)$ and $\rho(\mathbf{x}^\star \,|\, f)$ to generate the initial population, this is not required in general and the greater flexibility of a separate $\rho(\mathbf{X}_0)$ can be useful.

There are two basic types of evolution introduced by this model: *population evolution* and *artificial evolution*. Population evolution is a generic way of saying that different sample sets are used at different times, and is normally considered to be the meaning of the bare term "evolution" in evolutionary algorithms. Artificial evolution is an interesting statistical trick (with an unfortunately overloaded name) used to introduce needed diversity into empirical distributions produced by numerical inference methods such as particle filters. Each of these aspects of the model are described in greater detail below.

7.1 Population Evolution

Population evolution is the process of generating a population of samples using information obtained from a previous population. In a more general sense, it is better described as "population replacement", since the members of a population at time t need not have a direct relationship to members of the population at time $t-1$. Population evolution is introduced into the graphical model to accommodate the need for information refinement or intensification; given inferred information obtained from $\mathbf{X}_{t-1}$ and $\mathbf{Y}_{t-1}$, it is natural to ask whether more accurate information can be obtained by selecting additional samples. This is the motivation behind all evolutionary algorithms, and the approach taken here is most similar to that used by Estimation of Distribution Algorithms (EDAs).

EDAs, as the name implies, estimate a probability distribution from a population of candidate solutions, usually by culling locations that are not likely to be near the global minimum and looking for a pattern in the locations that remain. The distribution is then sampled to produce new candidate solutions, and the process is repeated. Over time, diversity in the population will decrease, indicating convergence to a single region of the domain. This region is then assumed to be the global minimum. This process tacitly assumes that locations of high fitness are likely to contain the most information about other locations of high fitness, and that assumption is shared by the GMEO and precisely defined in $\rho(\mathbf{X}_t \,|\, \mathbf{x}^\star_{t-1})$.

This assumption, while debatable, is sensible when considering functions at varying levels of detail, in the spirit of Graduated Non-Convexity or Sequential Quadratic Programming. Consider Rastrigin, which is essentially quadratic with additive high-frequency "noise". A posterior created by sampling from a low-confidence (high variance) prior will indicate a general area that contains the global minimum, but this smaller area must then be searched to find a more precise estimate of the minimum's location.

The proposed model throws away previous sample information, relying entirely upon $\rho(\mathbf{x}^\star \,|\, \mathbf{X}, \mathbf{Y})$ to generate new samples and $\rho(f \,|\, \mathbf{X}, \mathbf{Y})$ to contain all relevant information from the past. This allows for intensification of the search in the smaller resulting region while removing distractions introduced by distant samples.

7.2 Artificial Evolution

Artificial evolution is a term coined in the context of dynamic Bayesian models. It refers to a statistical trick that smooths the posterior distribution by introducing artificially noisy transitions (Liu and West, 2001).

While $\rho(f_t \,|\, f_{t-1})$ can obviously be used to model uncertainty about a function over time, it can simultaneously be used to facilitate the numerical inference process. For truly static functions it is tempting to define the distribution thus[2]:

$$\rho(f_t \,|\, f_{t-1}) = \begin{cases} 1 & \text{if } f_{t-1} = f_t \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

Given that the GMEO generally requires a numerical method of inference, however, this distribution is not likely to be useful. When employing a particle filter, for example, the estimate of $\rho_{t-1}(f \,|\, \mathbf{X}, \mathbf{Y})$ is represented by a discrete set of particles. Sampling from such a set is a solved problem (Liu and West, 2001), but over time the number of unique samples will decrease, since (15) admits no uncertainty; no noise is introduced that would allow other "nearby" particles to be generated.

Artificial evolution changes $\rho(f_t \,|\, f_{t-1})$ and $\rho(\mathbf{X}_t \,|\, \mathbf{x}^\star_{t-1})$ to introduce noise (e.g., additive Gaussian) into model dynamics, allowing for exploration by creating particles that have not been previously sampled. This approach is equivalent to smoothing the posterior before re-sampling (Liu and West, 2001), but is simpler to implement.

The definition of $\rho(f_t \,|\, f_{t-1})$, because it allows for the optimization of dynamic functions, represents an important new part of the function class specification. The introduction of artificial evolution into that distribution therefore affects the class specification. Care should therefore be taken that any definition of $\rho(f_t \,|\, f_{t-1})$ or $\rho(\mathbf{X}_t \,|\, \mathbf{x}^\star_{t-1})$ is consistent with the specified function class.

7.3 Results

In the results that follow, $\rho(f_t \,|\, f_{t-1})$ and $\rho(\mathbf{X}_t \,|\, \mathbf{x}^\star_{t-1})$ are assumed to be Gaussian distributions centered on the value of the conditional:

$$\rho(f_t \,|\, f_{t-1}) = \rho(\mathbf{c}_t \,|\, \mathbf{c}_{t-1}) = N_{\mathbf{c}_{t-1}, \Sigma_{\mathbf{c},t}}(\mathbf{c}_t) \tag{16}$$

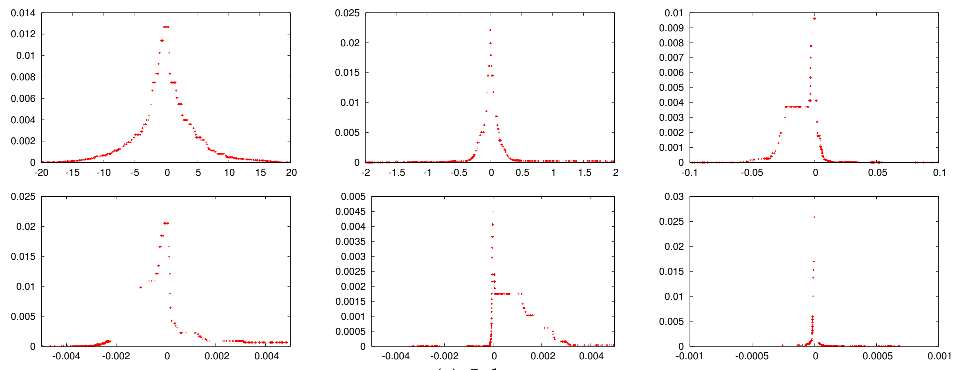
$$\rho(\mathbf{X}_t \,|\, \mathbf{x}^\star_{t-1}) = N_{\mathbf{x}^\star_{t-1}, \Sigma_{\mathbf{x},t}}(\mathbf{x}_t) \tag{17}$$

where $\Sigma_{\mathbf{c},t}$ and $\Sigma_{\mathbf{x},t}$ are covariance matrices that define the amount of jitter to add to a sample from an empirical distribution. In the results that follow, these matrices are defined thus:

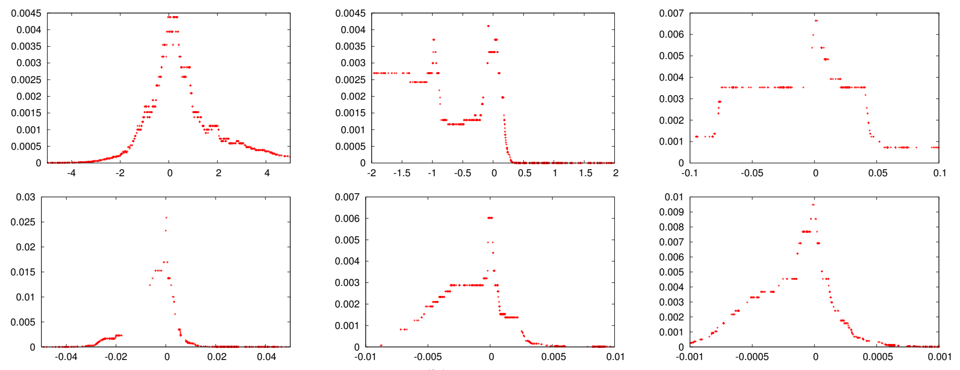
$$\Sigma_{\cdot,t} = \sigma^2_{\cdot,t}\mathbf{I} \tag{18}$$

with $\sigma_{\cdot,t}$ equal to twice the Euclidean distance to the nearest neighbor in the empirical distribution from which it was drawn.

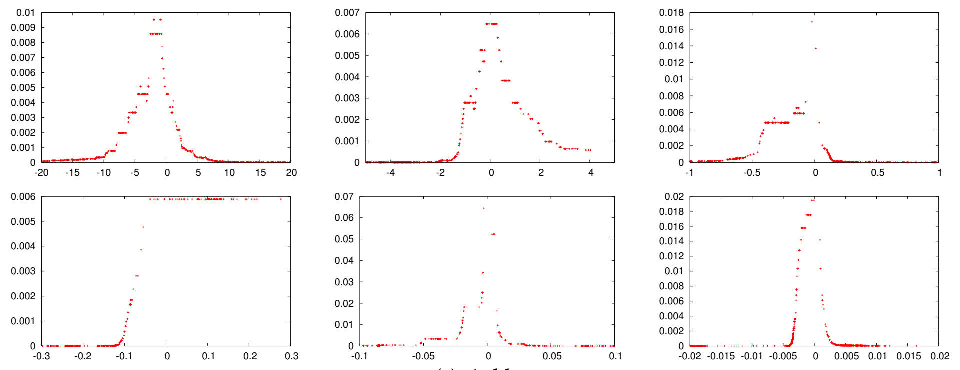
[2]This is an impossible density function and an abuse of notation, but the idea should at least be clear.



(a) Sphere



(b) Rastrigin



(c) Ackley

Figure 7: Plots of $\rho(\mathbf{x}^\star \,|\, \mathbf{X}, \mathbf{Y})$ over time for several one-dimensional functions in the GMEO

Results of applying this technique to PRC with $N_{\mathbf{X}} = 20$ and $N_{\mathbf{P}} = 1000$ are shown in Figure 7. The distributions should be read from left to right, top to bottom, with each new distribution representing one completed generation. Particular attention should be paid to the change in scale of the x-axis over time. The variance starts high and decreases over time as expected, searching successively smaller regions of the domain. This method behaves similarly in spaces of higher dimensionality.

The graphs have some characteristics worth noting. In Figure 7(c), the fourth generation has a posterior with a high, flat plateau on one side. This occurs with the PRC when all or most of the function evaluations occur on one side of the minimum; all pairs indicate a minimum in the same direction, and therefore it may be anywhere on one side. This is a rare but striking occurrence. Similar but less striking examples of this behavior appear in Figures 7(a) and 7(b). Of additional interest are the early generations for Rastrigin, where the multimodality of the function is in evidenced in the posteriors.

7.4 Discussion: Sources of Complexity

GMEO does not claim to do the impossible and overcome No Free Lunch, nor does it claim to overcome the curse of dimensionality; as the dimensionality of a problem increases, so does the size of the search space, independent of the applied optimization method. The GMEO does allow dimensionality to be reduced when prior information or sample observations cause $\rho(\mathbf{x}^\star \,|\, \mathbf{X}, \mathbf{Y})$ to shrink in one or more dimension. This process is superior in many ways to existing approaches to dimensionality reduction because it happens automatically with no additional intervention on the part of the optimization practitioner; once the function class is specified, Bayesian inference generates rational posterior distributions, and these distributions will not have weight along irrelevant dimensions.

The use of the particle filter unfortunately magnifies effects of the curse of dimensionality. While easy to implement and straightforward to explain, is not necessarily the best choice in these situations because it can require a very large number of particles to provide high quality distribution estimates in higher dimensions. This may be a tolerable problem in cases where function evaluations are very expensive, since the number of particles affects time in simulation, not in function evaluation; but is likely to be inappropriate when function evaluations are cheap. Other methods of numerical inference, e.g., Markov Chain Monte Carlo simulation (MCMC) (Pearl, 1987) or Variational Bayes (Jordan et al., 1999), should be the subject of future work in consideration of this issue.

If numerical inference is the bottleneck in evaluating the GMEO, the prior distribution $\rho(f)$ should be examined for unnecessary complexity. If $\rho(f)$ does not admit simplification of $\rho(\mathbf{x}^\star \,|\, f)$, or if $\rho(f)$ encodes a very large function class for which there is not sufficient information to reduce it, then the optimization practitioner has run headlong into No Free Lunch; either more information is needed to reduce the class size, or the problem at hand (e.g., needle-in-a-haystack) is inherently difficult.

8 Concluding Discussion

Throughout this work and in the context of each example, several concepts have been discussed. These included the role and definition of problem complexity, the full expression of No Free Lunch and its consequences, the nature of the burden placed on the optimization practitioner, and the curse of dimensionality. This section is devoted to insights and observations that have not yet received sufficient attention.

8.1 Post-Design Empiricism

Empirical results comparing GMEO with other continuous optimization algorithms are absent from this work. There are basically two common reasons behind performing comparative experiments and using them to draw conclusions at the end of a work on optimization:

- To discover the function class to which the proposed algorithm is well-tuned, and
- To advocate the use of the algorithm on a (hopefully related) set of problems.

The first is easily addressed because the function class is *specified* in the GMEO rather than merely analyzed; it is a fundamental requirement for the application of inference.

The second point is more interesting. The GMEO is a useful and instructive model of the general problem of optimization, one consequence of which is an inference-based algorithm that makes effective use of all supplied information. This statement applies to every optimization algorithm in existence, but in most cases the meaning and amount of embedded domain knowledge is encoded indirectly within the machinery of the algorithm; the practitioner can neither discern nor directly affect that knowledge.

A fair comparison of inference within the GMEO and any traditional algorithm is therefore impossible without first determining precisely what prior knowledge is available to the traditional algorithm and duplicating that information in the GMEO. Considering the fact that the bias of existing algorithms is generally unknown, this is a non-trivial task.

8.2 Model Expressiveness

Evolutionary algorithms are often designed for static, noise-free, single-objective optimization. If dynamic, noisy, or multi-objective optimization problems are of interest, these algorithms are generally adapted to accommodate them. The GMEO, however, admits specification of dynamics and noise in a straightforward and natural way via $\rho(f_t \mid f_{t-1})$ and $\rho(\mathbf{Y} \mid f, \mathbf{X})$, without algorithmic adaptation. In addition, because the relationship of f to $\mathbf{x}^\star$ is a statistical distribution, such a distribution can be devised to allow multiple minima to be tracked simultaneously[3].

8.3 Evolution and Clarity of Intent

When optimizing with the GMEO, the algorithm is fixed. Its successful application depends only on a specification of practitioner intent, encoded as a set of probability distributions. The distributions $\rho(f)$, $\rho(\mathbf{Y} \mid f, \mathbf{X})$, and $\rho(f_t \mid f_{t-1})$ define the function class; $\rho(\mathbf{x}^\star \mid f)$ defines the overall goal of search (e.g., optimization, although other goals may be expressed); and $\rho(\mathbf{X}_t \mid \mathbf{x}^\star_{t-1})$ and $\rho(\mathbf{X}_0)$ define the operation of population evolution.

These last distributions are less easily motivated than the others. While the other distributions have demonstrably clear meanings, the significance of these two distributions and the consequences of their specification are less clear; they represent such indirect and fuzzy notions as “scale reduction”, “greediness”, or “locality”. These ideas are interesting, and useful things can be said about them, but it is difficult to apply rigor to their definition.

[3]Note that this does not imply that all meanings of *multi-objective* can be easily incorporated. In particular, social welfare problems continue to be difficult regardless of the applied algorithm because of the fundamental impossibility of fairly assigning global utility given multiple individual utilities (Arrow, 1950). In these cases algorithms are often adapted to search for the Pareto Front, which is itself a problem of tracking infinitely many minima simultaneously.

While evolution itself is easily motivated by considering dynamic functions or time-shifting uncertainty, the use of $\rho(\mathbf{X}_t \,|\, \mathbf{x}^\star_{t-1})$ as the primary vehicle for defining the mechanics of evolution is suspect. Ongoing work in this area suggests that this mechanism hides an implicit definition of utility, one that assumes that near-optimal locations also provide maximal information. More explicit utilities provide opportunities to improve the model and will appear in future work.

9 Conclusions

The GMEO and the static models of which it is composed form a powerful framework for thinking about the problem of continuous optimization. The models provide a clear and direct way to encode the function class, they help to clarify the role of NFL in continuous problem complexity, and (with the exception of particle filter parameters and the choice of population size) the corresponding inference algorithm requires only a specific distribution-based declaration of intent for its operation.

While useful on these merits alone, the GMEO also points to some interesting avenues for future research, including a more thorough treatment of problem complexity (particularly sample complexity), the creation of a toolbox of useful function class definitions, the study of natural stopping criteria, and the addition of explicit utility specifications to allow for more principled population evolution.

Optimization problems with discrete valued outputs present another interesting area for future work. Such problems are even more directly tied to the original discrete proof of NFL and present some unique theoretical and algorithmic opportunities.

References

Arrow, K. (1950). A difficulty in the concept of social welfare. *The Journal of Political Economy*, 58(4):328–346.

Baluja, S. (1994). Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA.

Baluja, S. and Caruana, R. (1995). Removing the genetics from the standard genetic algorithm. In *Proceedings of the International Conference on Machine Learning (ICML 1995)*, pages 38–46, Lake Tahoe, California.

Bartlett, P. L., Long, P. M., and Williamson, R. C. (1996). Fat-shattering and the learnability of real-valued functions. *Journal of Computer and System Sciences*, 52(3):434–452.

Boggs, P. T. and Tolle, J. W. (1995). Sequential quadratic programming. In *Acta Numerica*, pages 1–51. Cambridge University Press, Cambrige.

Christianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press.

de Bonet, J. S., Charles L. Isbell, J., and Viola, P. (1997). MIMIC: Finding optima by estimating probability densities. *Advances in Neural Information Processing Systems*, 9:424–431.

De Freitas, J. F. G., Niranjan, M. A., Gee, A. H., and Doucet, A. (2000). Sequential monte carlo methods to train neural network models. *Neural Computation*, 12(4):955–993.

DeGroot, M. H. (1970). *Optimal Statistical Decisions*. McGraw-Hill, New York.

Doucet, A. (1998). On sequential simulation-based methods for bayesian filtering. Technical Report CUED/F-INFENG/TR.310, Cambridge University.

Emmerich, M., Giotis, A., Ozdemir, M., Back, T., and Giannakoglou, K. (2002). Metamodel-assisted evolution strategies.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

González, C., Lozano, J., and Larranaga, P. (2002). Mathematical modelling of umdac algorithm with tournament selection.

Grahl, J., Minner, S., and Rothlauf, F. (2005). Behaviour of umda/sub c/ with truncation selection on monotonous functions. In *The 2005 IEEE Congress on Evolutionary Computation*, volume 3, pages 2553– 2559.

Grassmann, W. K. and Tremblay, J.-P. (1996). *Logic and Discrete Mathematics*. Prentice Hall, Upper Saddle River, New Jersey.

Hastings, W. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press.

Jones, D., Schonlau, M., and Welch, W. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492.

Jordan, M. I., Ghahramani, Z., Jaakkola, T., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233.

Kearns, M. J. and Schapire, R. E. (1994). Efficient distribution-free learning of probabilistic concepts. *Journal of Computer and System Sciences*, 48(3):464–497.

Kennedy, J. and Eberhart, R. C. (1995). Particle swarm optimization. In *International Conference on Neural Networks IV*, pages 1942–1948, Piscataway, NJ. IEEE Service Center.

Krige, D. G. (1951). A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Chem., Metal. and Mining Soc. of South Africa*, 52(6):119–139.

Larrañaga, P. and Lozano, J. A., editors (2001). *Estimation of Distribution Algorithms : A New Tool for Evolutionary Computation*, volume 2. Springer.

Liu, J. and West, M. (2001). Combined parameter and state estimation in simulation-based filtering. In Doucet, A., De Freitas, J. F. G., and Gordon, N. J., editors, *Sequential Monte Carlo Methods in Practice. New York*. Springer-Verlag, New York.

MacKay, D. J. (1998). Introduction to gaussian processes. In Bishop, C. M., editor, *Neural Networks and Machine Learning, vol. 168 of NATO Asi Series. Series F, Computer and Systems Sciences*. Springer Verlag.

Macready, W. G. and Wolpert, D. H. (1996). What makes an optimization problem hard? *Complexity*, 5.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092.

Mitchell, T. M. (1997). *Machine Learning*. WCB/McGraw-Hill.

Mockus, J. (1989). *Bayesian Approach to Global Optimization*. Kluwer Academic Publishers, New York.

Mockus, J., Eddy, W., Mockus, A., Mockus, L., and Reklaitis, G. (1996). *Bayesian Heuritic Approach to Discrete and Global Optimization*. Kluwer Academic Publishers, Dordrecht, Netherlands.

Monson, C. K. and Seppi, K. D. (2004). The Kalman swarm. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004)*, volume 1, pages 140–150, Seattle, Washington.

Monson, C. K. and Seppi, K. D. (2005). Bayesian optimization models for particle swarms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2005)*, volume 1, pages 193–200, Washington, D.C.

Muhlenbein, H. and Mahnig, T. (1999). Convergence theory and applications of the factorized distribution algorithm. *JCIT: Journal of Computing and Information Technology*, 7.

Pearl, J. (1987). Evidential reasoning using stochastic simulation of causal models. *Artif. Intell.*, 32(2):245–257.

Pelikan, M. and Goldberg, D. E. (2000a). Hierarchical problem solving by the bayesian optimization algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2000)*, pages 267–274, Las Vegas, Nevada, USA. Morgan Kaufmann.

Pelikan, M. and Goldberg, D. E. (2000b). Research on the bayesian optimization algorithm. In *Optimization By Building and Using Probabilistic Models*, pages 216–219, Las Vegas, Nevada, USA.

Pelikan, M., Goldberg, D. E., and Cantú-Paz, E. (1999). BOA: The Bayesian optimization algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, volume I, pages 525–532, Orlando, FL. Morgan Kaufmann Publishers, San Fransisco, CA.

Pelikan, M., Goldberg, D. E., and Lobo, F. (2002). A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1):5–20.

Russel, S. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, New Jersey, second edition.

Sasena, M. J. (2002). *Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations*. PhD thesis, University of Michigan, Department of Mchanical Engineering.

Stuckman, B. E. and Easom, E. E. (1992). A comparison of bayesian/sampling global optimization techniques. *IEEE Transactions on Systems, Man and Cybernetics*, 22(5):1024–1032.

Syswerda, G. (1993). Simulated crossover in genetic algorithms. In *Proceedings of the Second Workshop on Foundations of Genetic Algorithms*, pages 239–255, Vail, Colorado. Morgan Kauffman.

Törn, A. and Žilinskas, A. (1989). *Global Optimization*. Springer, Berlin.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142.

Whitley, D. and Watson, J. P. (2006). Complexity theory and the no free lunch theorem. In Burke, E. K. and Kendall, G., editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Springer.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.