Internal Report 10-02 May 2010

Universiteit Leiden

Opleiding Informatica

Exploring High-Performance Regions with Model Induced Metropolis-Hastings

Jasper Abraham Visser

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Global optimization methods tend to focus on exploitation of known optima, often getting stuck in local optima. For problems with costly evaluations, it is more effective to initially identify the high-performance regions. This thesis proposes an algorithm designed to provide good coverage of the high-performance regions of an objective function using few objective function evaluations. The algorithm performs consecutive Metropolis-Hastings random walks on an RBFN meta-model of the objective function. After each walk, it adds the endpoint to the training set, then retrains the RBFN. Experiments show that the algorithm explores good solutions in significantly fewer objective function evaluations than state of the art algorithms, such Niching ES. The efficiency of the algorithm can be significantly increased by raising the acceptance function to some power. The map of the high-performance regions obtained can be used to initialize a more greedy optimization method. Moreover, the MIMH algorithm can be readily used to sample efficiently from a distribution the shape of which is determined by a costly evaluation function.

Acknowledgments

I thank Dr. Michael Emmerich for his very valuable feedback during the research and writing of this thesis.

Contents

Chapter 1

Introduction

The focus of this thesis is in conceiving a sampling method that emphasizes sampling from high-performance regions without performing a large number of objective function evaluations. This should be of particular interest to those interested in discovering the rough optima of an extremely costly objective function. The algorithm that is designed in this thesis is intended as a means of obtaining good starting points for more greedy and precise search.

This shall be accomplished by performing Metropolis-Hastings walks on a gradually improving model of the objective function. Each sampled point is added to the training set, which is then used to enhance the model. As a generic type of model, the radial basis function network (RBFN) is employed. The algorithm shall be referred to as Model Induced Metropolis-Hastings (MIMH).

In order to be successful, the conceived method must satisfy these three criteria:

- The algorithm emphasizes sampling from high-performance regions.
- The algorithm minimizes the number of objective function evaluations to be performed.

- The algorithm must be capable of finding all global optima of the objective function with some predefined accuracy ϵ.

It has been proven by [Has70] that the Metropolis-Hastings algorithm can be used to mimic sampling from an arbitrary distribution function. In this paper, it will be shown that MIMH converges on the same sampling distribution after a reasonable number of objective function evaluations.

1.1 Overview

Chapter 2 begins by describing the problem of global optimization. The remainder of the chapter is devoted to introducing the basic techniques on which the MIMH algorithm builds: the Metropolis-Hastings algorithm, and Radial Basis Function networks.

Chapter 3 introduces the MIMH algorithm and details the parameters of the algorithm and how they affect the behavior of the algorithm. Afterwards, it is proven that the sample set will converge on the target distribution. The computational cost of the algorithm is offset against the cost of objective function evaluations.

Experiments to test the convergence of the algorithm, to find the ideal value of one of the parameters, and to compare the algorithm to other approaches are described in Chapter 4. The results of these experiments are analyzed in Chapter 5.

Chapter 6 concludes this thesis by examining the niche the MIMH algorithm can occupy in the global optimization, and provides pointers for future research.

Chapter 2

Preliminaries and Basic Techniques

2.1 Global Optimization

Optimization algorithms typically have to choose between either exploring much of the landscape or having a good resolution on one or some local optima, which may not include the global optima. The former quality is often referred to as exploration, the latter quality as exploitation.

Most popular optimization algorithms use a method akin to local iterative optimization, and boast high performance on the exploitative scale. The drawback of this is that they can easily get trapped in local optima. Many such algorithms deal with this problem by means of a restart mechanism.

The MH algorithm, while not strictly an optimization algorithm, can be viewed as one extreme of the exploration vs. exploitation dimension. It is highly explorative, and offers little exploitation of known good areas.

The main drawback of MH as an optimization algorithm is the number of objective function evaluations required to obtain a proper sample. MIMH attempts to offset this drawback by using a model to estimate the value of the objective function for all points in the burn-in phase of the walk.

In many engineering problems, the cost of objective function evaluation is prohibitively high. MIMH allows these problems to be approached with the explorative qualities of the MH algorithm, offering a more divergent range of potential solutions.

Two other approaches to global optimization are:

- *Evolutionary Strategies* (ES) [BFM97] are a popular class of population-based optimization algorithms. Niching methods [Shi08] are an extension of ES, which offer parallel exploitation of optima by maintaining greater diversity in the population.

- *Simulated Annealing* (SA) [KGV83] is an adaptation of the MH algorithm, inspired by annealing in metallurgy. It employs a global parameter T, which is steadily decreased during the algorithm, and which controls the explorativity of the algorithm. Thus, SA can be highly explorative initially, and gradually become more exploitative.

2.2 High-Performance Regions

To measure the effectiveness of global optimization algorithm, usually a metric is used that focuses on the fitness function value of the best solution encountered so far, or on the distance to the nearest global optimum. Some metrics, such as the Mean Peak Ratio (Section 4.1.2), consider the possibility of multiple optima of interest.

In addition to the MPR metric, this thesis suggests examining the thoroughness of an algorithm in exploring the regions of the sample domain with low fitness value. These regions are dubbed the high-performance regions, and are defined as any regions of the sample domain with $f(\cdot) \leq f_{threshold}$ for and arbitrary $f_{threshold}$.

The High-Performance Region Coverage (Section 4.1.1) metric measures how well the set of points sampled by an algorithm cover the these regions.

2.3 Metropolis-Hastings Algorithm

Metropolis-Hastings (MH) is a Markov Chain Monte-Carlo algorithm. This class of algorithms can be used to sample from an arbitrary distribution $\pi(\cdot)$ that cannot be sampled from using a simpler distribution-specific method. The MH algorithm only requires that an unnormalized density function $g(\cdot)$ is known.

$$\pi(\mathbf{x}) = k \cdot g(\mathbf{x}) \tag{2.1}$$

where k is a normalizing constant. Since $\pi(\mathbf{x})$ is a probability distribution, the volume of the integral over the whole sample space S must be 1. Therefore:

$$k = \left(\int_{\mathbf{x} \in S} g(\mathbf{x}) d\mathbf{x} \right)^{-1} \tag{2.2}$$

Given $g(\cdot)$, the MH algorithm provides samples from the target density without requiring the value of k.

The Metropolis-Hastings algorithm was developed by Metropolis et al. [MRR+53] and generalized by Hastings [Has70]. It was used extensively in physics, but did not gain popularity under statisticians until the last decade of the century. The works of Gelfand & Smith [GS90] and Tierney [Tie94] stimulated interest under statisticians. Despite the late bloom, MH has evolved to one of the most influential algorithms introduced in the past century.

Given an d-dimensional sample space $\mathbb{R}^d$, the MH algorithm performs a random walk of length l through the sample space from a random starting point $\mathbf{x}_0$ to arrive at a final point $\mathbf{x}_l$ which is a proper sample of $\pi(\cdot)$. At each step $t \in \{1, \ldots, l\}$, a candidate point $\mathbf{y}$ is generated from the neighborhood of

$\mathbf{x}_t$ by some proposal density function $Q(\mathbf{x})$, usually a Gaussian distribution around $\mathbf{x}_t$, see Section 2.3.2.

$$\mathbf{y} \sim Q(\mathbf{x}_t) \tag{2.3}$$

The candidate point is then accepted or rejected based on the relative density function values:

$$\mathbf{x}_{t+1} = \begin{cases} \mathbf{y}, & \text{if } \alpha(\mathbf{x}_t, \mathbf{y}) \geq \mathrm{U}(0,1) \\ \mathbf{x}_t, & \text{otherwise} \end{cases} \tag{2.4}$$

with

$$\alpha(\mathbf{x}_t, \mathbf{y}) = \frac{g(\mathbf{y}) * q(\mathbf{y}, \mathbf{x}_t)}{g(\mathbf{x}_t) * q(\mathbf{x}_t, \mathbf{y})} \tag{2.5}$$

where $q(\mathbf{x}, \mathbf{y})$ is defined as the chance that the proposal density function will step from $\mathbf{x}$ to $\mathbf{y}$. If $q(\mathbf{x}, \mathbf{y})$ is symmetrical (as is assumed in the original Metropolis algorithm [MRR+53]), the two factors can be eliminated from the equation, yielding the simpler formula:

$$\alpha(\mathbf{x}_t, \mathbf{y}) = \frac{g(\mathbf{y})}{g(\mathbf{x}_t)} \tag{2.6}$$

Initially, the value of $\mathbf{x}_t$ will be largely dependent on the starting point $\mathbf{x}_0$. Over many iterations, however, the effect of initial state will be forgotten. The initial states are called the burn-in period and are discarded. It has been proven [Tie94, 1702] for a sufficiently long MH walk, that the stationary distribution converges to the target distribution $\pi(\cdot)$.

2.3.1 Limitations

In this section a method is described for transforming an arbitrary objective function $f(\mathbf{x})$ into an unnormalized density function $g(\mathbf{x})$ that can be used by the MH algorithm.

A requirement of the MH algorithm is that the integral of the density function is Lebesgue measurable. That is, the integral over the whole sample domain must have a finite value. Many objective functions do not meet this requirement, including those used in Section 4.2.

However, if the sample domain is limited to some finite interval S, for example a hypercube, and if it is assumed that $f(\mathbf{x})$ is locally Lipschitz continuous, then the integral of $f(\mathbf{x})$ over S will always have a finite value.

In this thesis, S is the hypercube defined by:

$$S = [x_{min}, x_{max}]^d \tag{2.7}$$

where both x_{min} and x_{max} are chosen to define the interval of interest for a specific objective function. Most of the equations can easily be adapted to accept a differently shaped bounded sample domain.

As an alternative approach to ensure that the integral of $f(\mathbf{x})$ over S is finite, the objective function can be altered to have zero value outside some bounded domain of interest, e.g.:

$$f'(\mathbf{x}) = \begin{cases} f(\mathbf{x}), & \text{if } \mathbf{x} \in S \\ 0, & \text{otherwise} \end{cases} \tag{2.8}$$

This approach is further examined and rejected in Section 3.3.1.

Since $f(\mathbf{x})$ is locally Lipschitz continuous, it will have finite value for all $\mathbf{x} \in S$, including:

$$f_{min} = \min_{\mathbf{x} \in S} f(\mathbf{x}), \tag{2.9}$$

$$f_{max} = \max_{\mathbf{x} \in S} f(\mathbf{x}) \tag{2.10}$$

An additional requirement of the MH algorithm is that:

$$\forall \mathbf{x} \in S : g(\mathbf{x}) > 0 \tag{2.11}$$

From Equation 2.6, it can be observed that if at any time $g(\mathbf{x}_t) = 0$, the acceptance chance is undefined, and if either $g(\mathbf{x}_t) < 0$ or $g(\mathbf{y}) < 0$, the acceptance chance is smaller than zero, resulting in unreachable regions of the sample space.

Also, since the focus of the MIMH algorithm is on minimization, rather than maximization, it should have a higher chance of sampling from an area with low $f(\mathbf{x})$ and a lower chance of sampling from an area with high $f(\mathbf{x})$.

An unnormalized density function that meets both requirements can be obtained by applying a simple transformative function to the objective function:

$$g(\mathbf{x}) = \begin{cases} \eta, & \text{if } f_{max} - f(\mathbf{x}) \leq \eta \\ f_{max} - f(\mathbf{x}), & \text{otherwise} \end{cases} \tag{2.12}$$

where η is some very small positive constant.

The resulting probability density is simply the normalized version hereof:

$$\pi(\mathbf{x}) = k \cdot g(\mathbf{x}) \tag{2.13}$$

where k is an unknown normalizing constant.

2.3.2 Generate candidate point

The Metropolis-Hastings algorithm generates a candidate point $\mathbf{y}$ by taking a random step from starting point $\mathbf{x}$. There are some possible methods of generating a random step. In this document, a hyperspherical direction generator [Zab03, 131] is used. It has the advantage of generating an independent random direction even in higher dimensions, which can easily be scaled to a random length by an independent random variable.

The hyperspherical direction generator provides vector with a random direction:

$$\mathbf{d} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_d \end{pmatrix}, d_i \sim \mathrm{N}(0,1) \tag{2.14}$$

This vector can subsequently be scaled to a random length with mean value 0 and variation σ:

$$\mathbf{x}' = \mathbf{x} + \mathbf{d} \cdot \frac{l}{|\mathbf{d}|} \tag{2.15}$$

with:

$$l \sim \mathrm{N}(0, \sigma) \tag{2.16}$$

Since the sample domain is bounded, as explained in Section 2.3.1, the proposal density function must be limited to the sample domain. A simple method to achieve this is to wrap $\mathbf{x}'$ around the edges of the domain

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix}, y_i = x_{min} + [(x'_i - x_{min}) \mod (x_{max} - x_{min})] \tag{2.17}$$

2.4 Radial Basis Function Networks

2.4.1 Introduction

Radial basis function networks (RBFN) are a class of neural network models. Specific to RBFN is that the activation of the hidden units is determined by the distance between the input vector $\mathbf{x}$ and some center vectors.

The hidden units in RBFN are represented by basis functions $\phi(x_n)$ with $x_n = |\mathbf{x} - \mathbf{c}_n|$ being the Euclidean distance between the input vector and the center of the n-th basis function $\mathbf{c}_n$. $\phi(x)$ is some non-linear function, usually the Gaussian:

$$\phi(x) = exp(-\frac{x^2}{2r^2}) \tag{2.18}$$

where r is a parameter which controls the smoothness properties of the interpolating function, often referred to as the radius. In this thesis, a single value of r will be used for all basis functions in the network.

The output of the RBFN is the sum of the radial basis function activations, with each basis function's contribution augmented by a weight value w_n:

$$h(\mathbf{x}) = \sum_n w_n \phi(|\mathbf{x} - \mathbf{c}_n|) + \widehat{w} \tag{2.19}$$

where $\widehat{w}$ is an additional global bias value.

Training in an RBFN consists of two steps. First, the center vectors and spread values of the hidden units are chosen. Second, the weights of the final layer are determined. More detail on this will be provided in Section 2.4.2.

2.4.2 Forward Selection

To train an RBFN involves a trade-off between perfectly mapping the training set and providing good interpolation for unknown points. It is easy to construct a network that perfectly maps all the points in the training set, but often, such a network will perform poorly on points outside the training set. Whenever the number of radial basis functions is limited, the interpolation of the target function will be enhanced.

The problem of finding the best subset of centers in a set of size N is usually intractable [Raw88]. There are $2^N - 1$ such subsets. However, there are a number of methods capable of finding a near-optimal subset. One such method is called forward selection.

The forward selection method [Orr96] starts with an empty subset, then iteratively grows the subset with a single basis function at a time. The centers of the basis functions are chosen from the training set X, and $\mathbf{r}$ is a single radius for all basis functions, given by:

$$\mathbf{r} = \zeta \cdot (r_1, \ldots, r_d)^{\mathrm{T}} \tag{2.20}$$

$$r_i = \max x_i - \min x_i \mid \mathbf{x} \in X \tag{2.21}$$

with d being the dimensionality of the input vectors and ζ some constant scaling factor. At each step, the basis function is chosen that would most reduce the sum-squared error. That basis function is added to the subset.

This process is repeated until some chosen criterion stops decreasing, usually the generalized cross-validation (GCV). The GCV is an error prediction metric that penalizes adding more than an effective number of parameters to the network. Although the sum-squared error will never increase as basis functions are added, the GCV begins to increase at some point.

The hidden-to-output weights are not selected. They are determined by solving a set of linear equations after the centers have been selected.

The forward selection method will produce different networks for different values of the scaling factor ζ. Higher values of ζ will result in larger radii for the basis functions and a smoother landscape. Lower values of ζ will result in narrower peaks and more often overfitting of the training data. Orr [Orr96] recommends that a number of possible values for ζ are used. For each value of ζ, an RBFN is constructed by the forward selection method. Finally, the network that minimizes the chosen criterion is chosen.

In the experiments in Chapter 4, the scaling factor ζ is chosen from:

$$Z = \left\{ \frac{d}{2^0}, \frac{d}{2^1}, \ldots, \frac{d}{2^5} \right\} \tag{2.22}$$

Forward selection is a non-linear algorithm, but it has the following advantages:

- There is no need to specify the number of basis functions in advance.
- The model selection criteria are tractable.
- The computational requirements are relatively low.

The implementation of RBFN and the forward selection method used in this document can be found at `http://www.anc.ed.ac.uk/rbf/rbf.html` (August 15th, 2009).

2.4.3 Benefits and Drawbacks

In this section, the benefits and drawbacks of RBFN as a model in the algorithm will be discussed.

Girosi & Poggio [GP89] prove that the set of functions that can be represented by an RBFN with a Gaussian basis function is dense in the space of Lipschitz continuous functions. This means that any continuous function can be approximated arbitrarily well by an RBFN. However, it requires that the training set is a representative subset of the input space. It will be shown in Section 3.4 that the training set provided by the MIMH algorithm converges on a dense subset of the input space.

Also, with supervised learning, the RBFN can be modeled without user intervention, solely based on the points sampled by the algorithm.

Radial basis functions have a local impact. A benefit of this locality is that the global bias value of the network determines the estimate fitness value for points outside the explored regions. Thus, this global bias value can be leveraged to affect the explorativity of the algorithm, as shown in Section 3.2.3.

The information stored in the RBFN can very easily be used to obtain predictions of the probable location of local and global optima.

A common problem to all kernel estimation and machine learning methods is affectionately known as the curse of dimensionality. This refers to the exponential growth of the number of data points required to model the target density as the dimensionality of the sample space increases. In [Sil86], the author reports that the sample size required to estimate the target density with a given accuracy is 4 for a 1-dimensional space, 19 for a 2-dimensional space and $842,000$ for a 10-dimensional space.

Coupled with the $O(n^3)$ cost of training the RBFN, it should be evident that the MIMH algorithm does not lend itself well to finding the minima of high-dimensional problems.

2.5 Generalized Least Squares Estimate

The generalized least squares method of [KO96] can be used to estimate the mean of a set of correlated values. In this case, the fitness values $\mathbf{y} = [y_1 \dots y_N]^{\mathrm{T}}$ are spatially correlated, with the correlation between fitness values y_i and y_j defined as a function of the Euclidean distance between the corresponding points $\mathbf{x}_i$ and $\mathbf{x}_j$:

$$c_\theta(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\theta \cdot |\mathbf{x}_i - \mathbf{x}_j|) \tag{2.23}$$

where θ is the correlation strength. When $\theta = 0$, the field is perfectly correlated. When $\theta \to \infty$, the data points will be assumed to be uncorrelated.

The process of finding a good value of θ for the calculation of the mean estimate is called the calibration phase. This is an expensive process, and it is outside the scope of this thesis to explore the calibration of θ in detail. In Section 3.2.3, where the generalized least-squares mean estimate of the training set is calculated, a best-guess value for θ is used.

Given N points $[\mathbf{x}_1 \dots \mathbf{x}_N]$ with the respective fitness values . The generalized least squares estimate $\widetilde{\beta}$ is defined by:

$$\widetilde{\beta} = \frac{\mathbf{1}^{\mathrm{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{y}}{\mathbf{1}^{\mathrm{T}} \cdot \mathbf{C}^{-1} \cdot \mathbf{1}} \tag{2.24}$$

with

$$\mathbf{C} = \begin{bmatrix} c_\theta(\mathbf{x}_1, \mathbf{x}_1) & \cdots & c_\theta(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ c_\theta(\mathbf{x}_N, \mathbf{x}_1) & \cdots & c_\theta(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}, \mathbf{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \tag{2.25}$$

Chapter 3

Approach

3.1 Model Induced Metropolis-Hastings

The Model Induced Metropolis-Hastings (MIMH) algorithm follows these steps:

1. Construct an initial model of the objective function;
2. Use Metropolis-Hastings to sample from the model;
3. Refine the model using the new samples;
4. Repeat step 2 and 3.

Each iteration of the algorithm, the training set used to update the model is expanded. The effectiveness of the algorithm hinges on two qualities:

- The model is able to learn the objective function well, and provide good interpolation.
- Samples drawn from the model provide new training data that increases the accuracy of the model, both in known good areas and in unexplored regions.

In Section 3.4 it is shown that, in the limit, the training set is a dense subset of the sample space. This provides an RBFN with more than sufficient well distributed points to learn the objective function. Clearly, obtaining a dense subset is undesirable in any real-world case. However, the experiment in Section 4.3.1 demonstrates that a good approximation of the objective function can be obtained in few iterations.

The outline of the algorithm is given in Algorithm 1.

Algorithm 1 Model Induced Metropolis-Hastings

```
for $i = 1$ to $\mu_0$ do
    $\mathbf{x} \leftarrow \mathrm{U}(x_{min}, x_{max})^n$
    $X \leftarrow X \cup \{(\mathbf{x}, f(\mathbf{x}))\}$
end for
repeat
    /* NB: $\bar{f}(\mathbf{x})$ is the scaled objective function value of $\mathbf{x}$, defined in Equation 3.6. */
    train RBFN on $(X, \bar{f}(X))$
    $\mathbf{x}_0 \leftarrow \mathrm{U}(x_{min}, x_{max})^n$
    for $i = 1$ to $l$ do
        $\mathbf{y} \leftarrow Q(\mathbf{x}_{i-1})$
        /* NB: $\widetilde{g}(\mathbf{x})$ is an estimate of the unnormalized target density function value of $\mathbf{x}$, defined in Equation 3.4. */
        if $\widetilde{g}(\mathbf{y})/\widetilde{g}(\mathbf{x}_{i-1}) > \mathrm{U}(0,1)$ then
            $\mathbf{x}_i \leftarrow \mathbf{y}$
        else
            $\mathbf{x}_i \leftarrow \mathbf{x}_{i-1}$
        end if
    end for
    $X \leftarrow X \cup (\mathbf{x}_l, f(\mathbf{x}_l))$
until termination criterion is met
```

3.2 Step by Step Description

3.2.1 Initial Modeling

The initial model of the objective function is obtained by training an RBFN on $\mu_0 > 2$ random points from the sample space. Obviously, the objective function must be evaluated for these points. In addition, any known points can be added to the training set, perhaps known from previous optimization attempts or other outside sources.

3.2.2 Sampling

Each iteration a point is sampled and added to the training set. This sample is obtained by performing a Metropolis-Hastings walk of length l. This would cost l objective function evaluations. To limit the number of actual evaluations that are made, the RBFN is used to estimate the fitness value of each point evaluated during the walk, given by the estimate function $\widetilde{f}(\mathbf{x})$. The actual fitness value is only calculated for the end point of the random walk.

Calculating the output of the meta-model for some point $\mathbf{x}$ costs only $O(m \cdot d)$ time, where m is the number of basis functions in the meta-model. The cost of the algorithm is further explored in Section 3.5.

As described in Section 2.3.1, the objective function, or in this case, the estimate objective function, must be normalized in order to obtain a usable density function. The density function used is an estimate of $g(\mathbf{x})$ (see Equation 2.12), defined by:

$$\widetilde{g}(\mathbf{x}) = \begin{cases} \eta, & \text{if } f_{max} - \widetilde{f}(\mathbf{x}) \leq \eta \\ f_{max} - \widetilde{f}(\mathbf{x}), & \text{otherwise} \end{cases} \tag{3.1}$$

where $\widetilde{f}(\mathbf{x})$ is an estimate of the objective function provided by the RBFN. Thus, the acceptance probability formula is changed to:

$$\alpha(\mathbf{x}_t, \mathbf{y}) = \frac{\widetilde{g}(\mathbf{y})}{\widetilde{g}(\mathbf{x}_t)} \tag{3.2}$$

Usually, the real value of f_{max} is not known beforehand. This, too, can be estimated by using the observed maximum fitness value:

$$\widetilde{f}_{max} = \max_{\mathbf{x} \in X} f(\mathbf{x}) \tag{3.3}$$

By extension, Equation 3.1 is rewritten to include the observed maximum fitness:

$$\widetilde{g}(\mathbf{x}) = \begin{cases} \eta, & \text{if } \widetilde{f}_{max} - \widetilde{f}(\mathbf{x}) \leq \eta \\ \widetilde{f}_{max} - \widetilde{f}(\mathbf{x}), & \text{otherwise} \end{cases} \tag{3.4}$$

In Section 3.4, the convergence of $\widetilde{f}_{max}$ to f_{max} and the drawbacks of using an estimate for f_{max} shall be explored.

When each MH walk is completed, the final point of the walk is evaluated using the objective function $f(\mathbf{x})$. Thereafter that point is added to the training set X.

3.2.3 Scaling

It is possible to leverage the effect of the global bias value, or rather the absence of one, to control the explorativity of the algorithm.

For points in unexplored regions, the effect of the radial basis functions is negligible and $h(\mathbf{x})$ is dominated by the global bias value. Therefore, the global bias value determines how explorative the algorithm will be. If the weight is near $\widetilde{f}_{max}$, the chance of sampling from unexplored regions will be near zero. If the weight is near $\widetilde{f}_{min}$ (see Equation 3.5), unexplored regions will be treated as high-performance regions.

$$\widetilde{f}_{min} = \min_{\mathbf{x} \in X} f(\mathbf{x}) \tag{3.5}$$

Assuming no global bias value is added to the network, $h(\mathbf{x}) \approx 0$ for points in unexplored regions. If, before training, the fitness values of the points in the training set are scaled to an interval around 0, such as $[-\lambda; 1]$ for some positive constant λ, the position of 0 in that interval determines the relative position of the fitness value estimate for unexplored regions.

If $f(\mathbf{x})$ is the fitness value of the $\mathbf{x} \in X$, then the scaled fitness value of that element is defined by:

$$\bar{f}(\mathbf{x}) = -\lambda + \left(\frac{f(\mathbf{x}) - \widetilde{f}_{min}}{\widetilde{f}_{max} - \widetilde{f}_{min}} \right) \cdot (1 + \lambda) \tag{3.6}$$

This equation ensures that

$$\forall \mathbf{x} \in X : f(\mathbf{x}) = \widetilde{f}_{min} \Rightarrow \bar{f}(\mathbf{x}) = -\lambda \tag{3.7}$$

and

$$\forall \mathbf{x} \in X : f(\mathbf{x}) = \widetilde{f}_{max} \Rightarrow \bar{f}(\mathbf{x}) = 1 \tag{3.8}$$

with all intermediate values linearly scaled in the defined interval.

The RBFN is trained on all points $\mathbf{x} \in X$ and the respective scaled fitness values $\bar{f}(\mathbf{x})$.

It is easy to scale the output values of the RBFN back to the original interval of $[\widetilde{f}_{min}, \widetilde{f}_{max}]$:

$$\widetilde{f}(\mathbf{x}) = \widetilde{f}_{min} + \left(\frac{h(\mathbf{x}) + \lambda}{1 + \lambda} \right) \cdot (\widetilde{f}_{max} - \widetilde{f}_{min}) \tag{3.9}$$

In order to provide reasonable interpolation, λ is set to ensure that for unexplored regions $\widetilde{f}(\mathbf{x})$ will be near an estimate of the mean fitness value for the whole landscape. This estimate is obtained by calculating the generalized least-squares mean estimate $\widetilde{\beta}$ (Section 2.5) of the training set.

The calibration of the correlation strength θ used in calculating the generalized least-squares mean estimate can be very expensive. In this thesis, a best guess is used for θ. It is set to the radius of the radial basis functions in the previous training step:

$$\theta_t = \begin{cases} 1, & \text{if } t = 0 \\ \left|\mathbf{r}_{t-1}\right|, & \text{if } t > 0 \end{cases} \tag{3.10}$$

For Equation 3.9 to yield $\widetilde{f}(\mathbf{x}) = \widetilde{\beta}$ when $h(\mathbf{x}) = 0$, it is possible to determine the value of λ by simply solving a linear equation. The resulting value for λ is given by:

$$\lambda = \frac{\widetilde{\beta} - \widetilde{f}_{min}}{\widetilde{\beta} + \widetilde{f}_{max} - 2 \cdot \widetilde{f}_{min}} \tag{3.11}$$

3.2.4 Training RBFN

The RBF network is retrained on the now-expanded training set. The exact training method, target criteria, and values for parameters of the training can be chosen specifically for the objective function. In this document, the forward selection training method [Orr96] is used, as described in Section 2.4.2.

Specifically, the RBFN is trained using the mapping of the points in the training set to the scaled fitness values, $\bar{f} : \mathbf{X} \rightarrow \bar{\mathbf{y}}$. As a result, $h(\mathbf{x})$ will be similarly scaled. The output values of the RBFN can be scaled back to the original interval $[\widetilde{f}_{min}, \widetilde{f}_{max}]$ using Equation 3.9.

3.3 Parameters of the Algorithm

3.3.1 Step Variation

The step variation σ, encountered in Equation 2.15, determines the size of the steps in a MH walk.

If σ is small, most steps generated will be local, and will most likely be accepted. On the other hand, if σ is too large, more candidate points are generated in far-off low probability areas, and the algorithm will end up rejecting more steps. In either case, the autocorrelation in the points in the MH walk will be high.

Much research has been done to the ideal acceptance rate; that is, the rate of accepted samples to rejected samples that offers the best mixture. [RR01] find that the optimal acceptance rate is approximately 0.35 when $d = 2$ and converges to 0.234 when $d \to \infty$. Also, they conclude that acceptance rates between 0.15 and 0.5 do not decrease efficiency significantly.

It should be noted that these conclusions apply to an infinite sample domain with a Lebesgue measurable integral over $g(\cdot)$. In contrast, the focus of this thesis is on bounded domains and all of the test objective functions described in Section 4.2 would have an infinite integral over an infinite domain.

If the proposal density function $Q(\mathbf{x})$ is modified to allow steps outside the bounds of the domain, $g(\cdot)$ can easily be altered to accept an infinite domain:

$$\forall \mathbf{x} \notin [x_{min} \ldots x_{max}]^d : g(\mathbf{x}) = 0 \tag{3.12}$$

This modification does not alter the stationary distribution of the algorithm, since it does not affect the acceptance function $\alpha(\mathbf{x}, \mathbf{y})$ for any $\mathbf{x}$ that can be reached by the random walk. It can only provide additional rejected samples, when the proposal function reaches outside the bounds of the domain.

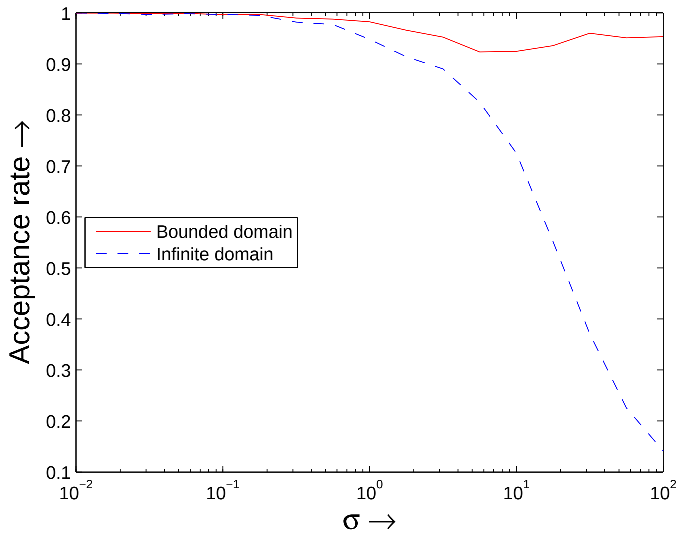


Figure 3.1: Average acceptance rate for different values of σ on both a bounded domain and an infinite domain.

To illustrate the difference of using an infinite or a bounded sample domain, the MIMH algorithm is run on the Moonface function [Jac97], which is defined as:

$$f(\mathbf{x}) = d_1 - \frac{50 \cdot \sin(d_1)}{d_1} + d_2 - \frac{50 \cdot \sin(d_2)}{d_2} + d_3 - \frac{50 \cdot \sin(d_3)}{d_3} - 100$$

with:

$$d_1 = \left| \mathbf{x} - \begin{pmatrix} 11 \\ 9 \end{pmatrix} \right|, d_2 = \left| \mathbf{x} - \begin{pmatrix} -11 \\ -3 \end{pmatrix} \right|, d_3 = \left| \mathbf{x} - \begin{pmatrix} 6 \\ -9 \end{pmatrix} \right|$$

for a 2-dimensional $\mathbf{x}$ bounded by $-20 \leq x_1, x_2 \leq 20$. For the sake of completeness, this function is repeated in Section 4.2.2. The algorithm is run for 100 iterations with different values of σ and a fixed walk length of $l = 100$. In Figure 3.1 the average acceptance rate is plotted against σ.

It is clear from Figure 3.1 that on a bounded domain the average acceptance rate is high for all practical values of σ. On an infinite domain, the acceptance rate drops off at high values of σ. However, all the additionally rejected samples are located outside the bounds of the input domain. This does not imply better mixture of the points in the random walk, since the accepted points are drawn from the same distribution as the bounded domain variant.

Unfortunately, this exercise provides no clues to a proper value for σ. In this thesis, σ is set to a fraction of the diagonal of the sample space:

$$\sigma = \frac{1}{m} \cdot (x_{max} - x_{min}) \cdot \sqrt{d} \tag{3.13}$$

where m is an arbitrary number. Also, in this thesis, $m = 100$ is used.

3.3.2 Walk Length

The length of the Metropolis-Hastings walk l should be large enough that the final point is a sufficiently randomized sample. If l is too low, the final point will be biased toward the start point of the walk. If l is too high, it adds to the cost of the algorithm with little added value.

For the experiments in this document, l is set to the average number of steps needed to move across the sample domain from one corner of the hyperrectangle to its opposite corner. The step variation σ determines the mean size of the steps in a MH walk. The mean step size is equal to the absolute mean deviation of the factor $\mathrm{N}(0, \sigma)$. It can be deduced that the mean step size is $\sigma\sqrt{\frac{2}{\pi}}$.

Thus, the intended walk length is defined by:

$$l = \left\lfloor (x_{max} - x_{min}) \cdot \frac{1}{\sigma} \cdot \sqrt{\frac{\pi d}{2}} \right\rfloor \tag{3.14}$$

This formula can easily be adapted to account for problems where not all dimensions have equal bounds.

If σ is chosen according to Equation 3.13, then Equation 3.14 can be simplified to:

$$l = m \cdot \sqrt{\frac{\pi}{2}} \tag{3.15}$$

3.3.3 Probability Exponent

A variant on the standard MIMH algorithm is presented that uses an exponent parameter κ to adjust the shape of the unnormalized density estimate. This approach simply replaces the unnormalized density estimate function $\widetilde{g}(\mathbf{x})$ used in the MH walks with:

$$\bar{g}(\mathbf{x}) = \widetilde{g}(\mathbf{x})^{\kappa} \tag{3.16}$$

The standard MIMH algorithm can be seen as a specific case of this variant with $\kappa = 1$.

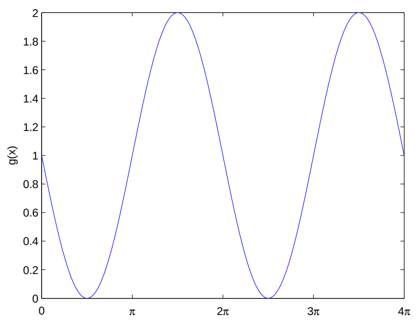
By choosing a value of $\kappa > 1$, the differences in density function values are emphasized, as shown in Figure 3.2, and by choosing a value of $\kappa < 1$, the differences in the density function values are reduced. By looking at the acceptance function in Equation 3.2, it is possible to see that if, for some point $\mathbf{x}$ and a candidate point $\mathbf{y}$, the probability density estimates are related as $\widetilde{g}(\mathbf{y}) = \beta \cdot \widetilde{g}(\mathbf{x})$, then the chance of accepting $\mathbf{y}$ is simply:

$$\alpha(\mathbf{x}, \mathbf{y}) = \frac{\widetilde{g}(\mathbf{y})}{\widetilde{g}(\mathbf{x})} = \beta \tag{3.17}$$

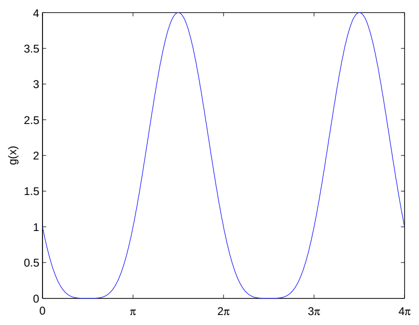
or, if the density function $\bar{g}(\mathbf{x})$ is substituted:

$$\alpha(\mathbf{x}, \mathbf{y}) = \frac{\bar{g}(\mathbf{y})}{\bar{g}(\mathbf{x})} = \beta^{\kappa} \tag{3.18}$$

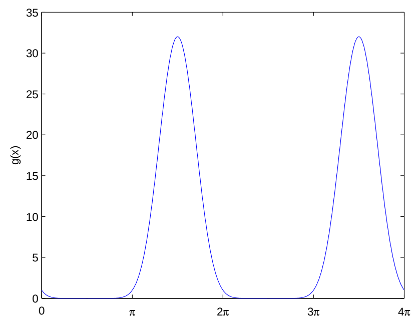
Thus, the κ parameter can be used to make the algorithm more or less



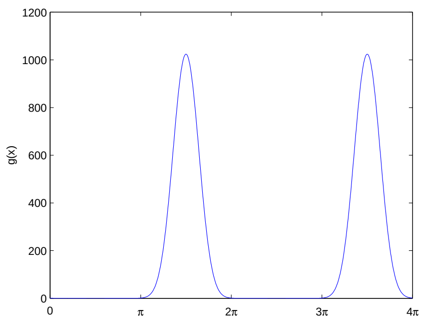
(a) $\kappa = 1$



(b) $\kappa = 2$



(c) $\kappa = 5$



(d) $\kappa = 10$

Figure 3.2: Adjusted unnormalized density function $\bar{g}(\mathbf{x})$ for objective function used is $f(\mathbf{x}) = \sin(\mathbf{x})$ using different values of κ.

greedy. The effects of this parameter on the discovery of the minima and coverage of the high-performance regions are explored in an experiment in Section 4.3.2.

3.4 Convergence

Although the main intention was to design of the MIMH algorithm as an explorative global optimization method, it is also a method of sampling from an arbitrary distribution, the shape of which is defined by a cost expensive evaluation function. Assuming the $f(\mathbf{x})$ is Lipschitz continuous with some constant $L < \infty$ and assuming S is a compact sample space, the samples drawn by the MIMH algorithm will eventually converge on the target distribution. However, there are obstacles that slow down convergence to $\pi(\cdot)$, and until the training set is a dense subset of the sample space, the distribution of the samples drawn will be skewed.

The estimate probability density can be obtained from Equation 3.4 as:

$$\widetilde{\pi}(\mathbf{x}) = \widetilde{k} \cdot \widetilde{g}(\mathbf{x}) \tag{3.19}$$

where $\widetilde{k}$ is a normalizing constant. More specifically, to refer to the estimate probability density at iteration t:

$$\widetilde{\pi}_t(\mathbf{x}) = \widetilde{k}_t \cdot \widetilde{g}_t(\mathbf{x}) \tag{3.20}$$

with

$$\widetilde{g}_t(\mathbf{x}) = \begin{cases} \eta, & \text{if } \widetilde{f}_{t,max} - \widetilde{f}_t(\mathbf{x}) \le \eta \\ \widetilde{f}_{t,max} - \widetilde{f}_t(\mathbf{x}), & \text{otherwise} \end{cases} \tag{3.21}$$

In order to prove convergence, it must be shown that:

$$\forall \mathbf{x} \in S : \lim_{t \to \infty} \widetilde{\pi}_t(\mathbf{x}) = \pi(\mathbf{x}) \tag{3.22}$$

This criterion is satisfied if:

$$\forall \mathbf{x} \in S : \lim_{t \to \infty} \widetilde{g}_t(\mathbf{x}) = g(\mathbf{x}) \tag{3.23}$$

and

$$\lim_{t \to \infty} \widetilde{k}_t = k \tag{3.24}$$

Equation 3.24 can be expanded using Equation 2.2 to:

$$\lim_{t\to\infty}\left(\int_{\mathbf{x}\in S}\widetilde{g}_t(\mathbf{x})d\mathbf{x}\right)^{-1} = \left(\int_{\mathbf{x}\in S} g(\mathbf{x})d\mathbf{x}\right)^{-1} \tag{3.25}$$

Clearly, this is a weaker form of the criterion in Equation 3.23.

Equation 3.23 is met if:

$$\forall\mathbf{x}\in S : \lim_{t\to\infty}\widetilde{f}_t(\mathbf{x}) = f(\mathbf{x}) \tag{3.26}$$

and

$$\lim_{t\to\infty}\widetilde{f}_{t,max} = f_{max} \tag{3.27}$$

Since $\widetilde{f}_{t,max}$ is the maximum value of $\widetilde{f}_t(\mathbf{x})$ for all $\mathbf{x}\in X_t$ and f_{max} is the maximum value of $f(\mathbf{x})$ for all $\mathbf{x}\in S$, the criterion given in Equation 3.27 is covered by the criterion given in Equation 3.26 if additionally:

$$\lim_{t\to\infty} X_t = S \tag{3.28}$$

In other words, it requires that in the limit the training set is a dense subset of the sample space.

It has been proven by Girosi & Poggio [GP89] that an RBFN can approximate any Lipschitz continuous function with arbitrary accuracy when given a training set that is a dense subset of the sample space. Therefore, the requirement in Equation 3.26 is met if Equation 3.28 is true.

The training set converges on a dense subset of the sample space, because, for any ϵ-ball $B_\epsilon \in S$ with $\epsilon > 0$, the chance of sampling in that ϵ-ball is always greater than 0, or:

$$\forall B_\epsilon \subset S : \epsilon > 0 \Rightarrow \pi(B_\epsilon) > 0 \tag{3.29}$$

From Equations 3.21 and 2.2 can be derived that the minimum chance of sampling in B_ϵ is:

$$\widetilde{\pi}_t(B_\epsilon) \geq \frac{\eta \cdot \Lambda(B_\epsilon)}{\int_{\mathbf{x} \in S} \widetilde{g}_t(\mathbf{x}) d\mathbf{x}} \tag{3.30}$$

Since η is defined as a small positive value and $\epsilon > 0$, $\eta \cdot \Lambda(B_\epsilon) > 0$. The integral of $\widetilde{g}_t(\mathbf{x})$ over S has a finite positive value, delimited by the minimum possible volume $\eta \cdot \Lambda(S)$ and the maximum possible volume $\left(\widetilde{f}_{t,max} - \widetilde{f}_{t,min}\right) \cdot \Lambda(S)$.

Because S has been assumed to be a finite sample space and $f(\mathbf{x})$ to be a Lipschitz continuous function, both $\Lambda(S)$ and $\left(\widetilde{f}_{t,max} - \widetilde{f}_{t,min}\right)$ cannot have an infinite value.

The chance of drawing a sample $\mathbf{x} \in B_\epsilon$ is always greater than 0. Therefore, the chance that the training set will contain an element $\mathbf{x} \in B_\epsilon$ converges on 1 as $t \to \infty$. If the training set is a dense subset of the sample space, the RBFN is able to accurately approximate the objective function. By extension, the estimate probability density converges on the target density.

3.5 Cost

The benefit of MIMH is that it decreases the number of objective function evaluations that are required to obtain a good high-performance region sampling of the objective function. However, this benefit comes at a cost: the time spent running the algorithm.

The cost of the algorithm can be subdivided in two parts, expressed in terms of the time cost $T(n)$ of sampling the n-th point:

- The cost of Metropolis-Hastings walks on the objective function estimate.
- The cost of retraining the RBFN.

Even though most parameters of the algorithm are constant, they are kept as part of the equations below to illustrate the effect they have on the time cost of the algorithm.

3.5.1 Cost of Sampling

To sample a single point, the MIMH algorithm performs a walk of length l. At each step of the walk, a candidate point is generated and the objective function value is estimated. The cost of generating a candidate point is constant; the cost of estimating an objective function value is in the order of $O(M \cdot d)$, where M is the number of radial basis functions in the network. By extension, the cost of a single MH walk is in the order of $O(l \cdot M \cdot d)$.

There is no clear relationship between M and the size of the training set. The number of radial basis functions used to model the objective function depends on many factors, including the complexity of the objective function, the parameters of the training method and the distribution of the points in the training set. If the worst case scenario is assumed, the number of radial basis functions is equal to the size of the training set, or $M = n - 1$. In this case, the cost of an MH walk in terms of n is in the order of:

$$T_1(n) \in O(l \cdot n \cdot d) \tag{3.31}$$

3.5.2 Cost of Training

Forward selection, while one of the cheaper methods of training an RBFN, comes at a cost in the order of $O(N^3 \cdot d)$, where N is the size of the training set. For MIMH, $N = n - 1$, thus:

$$T_2(n) \in O(n^3 \cdot d) \tag{3.32}$$

Clearly, as $n \to \infty$, the cost of training the RBFN outweighs the cost of the MH walks.

3.5.3 Comparison

If the algorithm were to sample points from the actual objective function, rather than a model thereof, it would require a MH walk of length l for each sampled point. Given that a single objective function evaluation has a fixed cost of t_{obj}, the time saved by using a model is equal to:

$$T_3 \in O(l \cdot t_{obj}) \tag{3.33}$$

Thus, the MIMH algorithm saves time if:

$$T_3 > T_1(n) + T_2(n) \tag{3.34}$$

Because T_3 is invariant to n and both $T_1(n)$ and $T_2(n)$ are dependent on n, at some point the cost of estimating the objective function will exceed the cost of actually evaluating the objective function.

Chapter 4

Experimental Setup

In this chapter, the setup of 3 experiments is described. In these experiments, the convergence to a target distribution (Section 4.3.1) and the effect of the κ parameter (Section 4.3.2) are examined, and the performance of the MIMH algorithm is compared to that of a Niching CMA-ES and uniform random sampling (Section 4.3.3). First, however, the performance metrics and the objective functions used in these experiments are introduced.

4.1 Performance Metrics

4.1.1 High-Performance Region Coverage

The High-Performance Region Coverage (HPRC) is a new metric proposed to measure how well a sample set X covers the high-performance regions of the sample domain. The set T consists of the points in high-performance regions of the sample domain:

$$T = \{\mathbf{x} \in S | f(\mathbf{x}) \leq f_{threshold}\} \tag{4.1}$$

for some threshold fitness value $f_{threshold}$. In the experiments in Section 4.3, the threshold fitness value is set at:

$$f_{threshold} = f_{min} + 0.1 \cdot (f_{max} - f_{min}) \tag{4.2}$$

The HPRC is defined as the integral over all $\mathbf{x} \in T$ of the distance to the nearest neighbor in X, weighted by the unnormalized density function:

$$\upsilon(X) = \frac{1}{\Lambda(T)} \cdot \int_{\mathbf{x} \in T} (f_{max} - f(\mathbf{x})) \cdot d(\mathbf{x}, X) d\mathbf{x} \tag{4.3}$$

where $d(\mathbf{x}, X)$ is the Euclidean distance between $\mathbf{x}$ and the nearest neighbor in X:

$$d(\mathbf{x}, X) = \left| \mathbf{x} - \arg\min_{\mathbf{x}' \in X} |\mathbf{x} - \mathbf{x}'| \right| \tag{4.4}$$

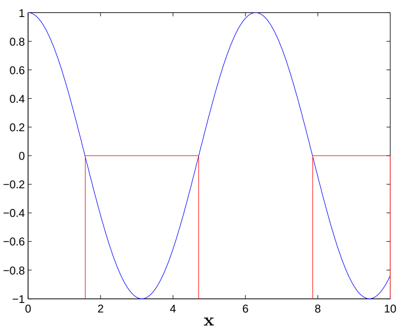
and $\Lambda(T)$ is the Lebesgue measure of the set T. An example of the HPRC calculation is shown in Figure 4.1.

In Equation 4.3 the points with lower fitness value will have more impact on the value of $\upsilon(X)$, and similarly points with a greater distance to the nearest point in X. As the sample set provides better coverage of the high-performance regions, the HPRC value decreases.

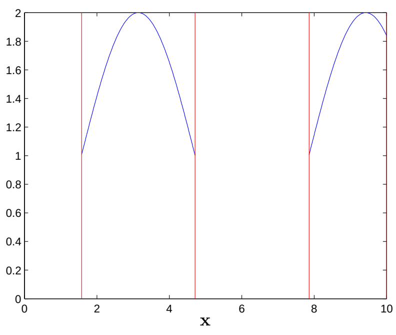
Since the value of the integral and the Lebesgue measure of T cannot be calculated, except for some simple objective functions, an approximation of the $\upsilon(X)$ is obtained by means of a Monte Carlo integration method.

It employs a finite comparison set $\Upsilon \subset T$. The size N of the comparison set is a sufficiently large number that Υ can be considered a well distributed subset of the sample space with high density. The comparison set and an approximation $\widehat{\Lambda}(T)$ of the Lebesgue measure of T are obtained by means of Algorithm 2.

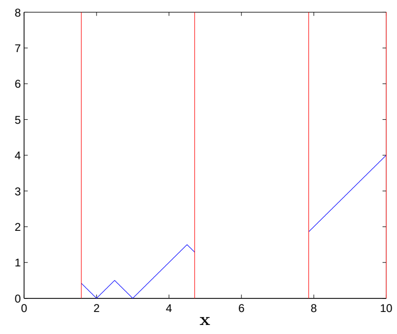
The HRPC is approximated by the sum of the weighted distances of each



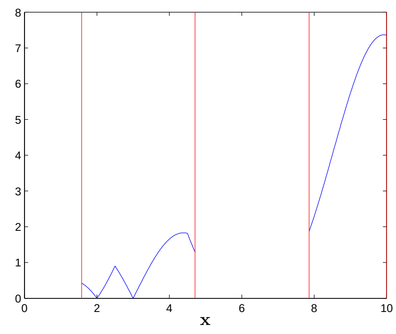
(a) $f(\mathbf{x})$



(b) $f_{max} - f(\mathbf{x})$



(c) $d(\mathbf{x}, X)$



(d) $(f_{max} - f(\mathbf{x})) \cdot d(\mathbf{x}, X)$

Figure 4.1: Example of HPRC calculation for a 1-dimensional sample space with $f(\mathbf{x}) = \cos(\mathbf{x})$, $f_{threshold} = 0$ and $X = \{2, 3, 6\}$. The high-performance set T consists of the areas in the red boxes in Figure 4.1(a). The HPRC value $\upsilon(X)$ is defined as the volume under Figure 4.1(d) divided by the Lebesque measure of T.

point $\mathbf{u} \in \Upsilon$ to the nearest point in X.

$$\widehat{\upsilon}(X) = \frac{1}{\widehat{\Lambda}(T)} \cdot \sum_{\mathbf{u} \in \Upsilon} (f_{max} - f(\mathbf{u})) \cdot d(\mathbf{u}, X) \tag{4.5}$$

In effect, $\widehat{\upsilon}(X)$ is minimized by spreading out the points in X over the high-performance regions.

Algorithm 2 Select Υ

```
$s \leftarrow 0$
**for** $i = 1$ to $N$ **do**
  **repeat**
    $\mathbf{x} \leftarrow \mathrm{U}(x_{min}, x_{max})^n$
    $s \leftarrow s + 1$
  **until** $\mathbf{x} \in T$
  $\Upsilon \leftarrow \Upsilon \cup \mathbf{x}$
**end for**
$\widehat{\Lambda}(T) \leftarrow \Lambda(S) \cdot N/s$
```

4.1.2 Mean Peak Ratio

The Mean Peak Ratio (MPR) is a metric to score a search method based on its ability to find not only the global optimum, but any number of local optima. It was first suggested by [MS96], and also employed by [Shi08].

In [MS96], the optima are defined as the maxima of the objective function. Since the MIMH algorithm focuses on finding the minima, this document shall employ an altered version of the MPR calculation where the fitness value $f(\mathbf{x})$ is replaced by $f_{max} - f(\mathbf{x})$.

Another restriction of the MPR metric is that it requires all fitness values to be positive values. This criterion is met, because $\forall \mathbf{x} \in X : f_{max} \geq f(\mathbf{x})$.

The MPR metric looks for the point in the population with the highest fitness value near each optimum. It considers only points that are no further from the optimum than a certain distance, known as the niche radius or ρ. Additionally, the point must have a fitness value that is at least 80% of the fitness value of the optimum.

In other words, for each optimum $\mathbf{y}_i \in Y$, $\widehat{X}_i$ is the subset of points in X that meet these two criteria:

- $f_{max} - f(\mathbf{x}) \geq 0.8 \cdot (f_{max} - f(\mathbf{y}_i))$
- $|\mathbf{x} - \mathbf{y}_i| < \rho$

The MPR value is defined as:

$$\mathrm{MPR} = \frac{\sum_{i=1}^{N} \begin{cases} 0, & \text{if } \widehat{X}_i = \emptyset \\ \max_{\mathbf{x} \in \widehat{X}_i} (f_{max} - f(\mathbf{x})) & \text{otherwise} \end{cases}}{\sum_{i=1}^{N} f_{max} - f(\mathbf{y}_i)} \tag{4.6}$$

The main drawback of the MPR as a metric is that it can only be calculated properly if the true optima are known in advance, either by direct calculation or numerical approximation. For the test problems employed in this document, this is not a problem. For real-world problems, where neither the location, nor the fitness of the actual optima are known, another metric for the performance of the algorithm should be chosen.

4.1.3 Pearson's Chi-Square Test

In addition to a global optimization method, the MIMH algorithm also serves as a method for sampling from an arbitrary distribution. The convergence of the distribution of the sampled points to the target distribution can be measured using Pearson's chi-square test.

Pearson's chi-square test [Pea92] is used to test the goodness of fit of observed data against an expected distribution. The observed data is a set of samples obtained by the algorithm. For some problems, the expected distribution can be directly calculated from the integral of the target density function; in other cases, it must be estimated by drawing a very large sample set from the target distribution. The null hypothesis is that the observed data matches the expected distribution.

Pearson's chi-square test divides the sample space into a finite number n of non-overlapping cells. For the i-th cell, O_i denotes the number of samples in the observed data in that cell, and E_i is the expected number of samples.

The degree of freedom k is equal to the number of cells minus 1:

$$k = n - 1 \tag{4.7}$$

The chi-square value is given by:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{4.8}$$

The higher the chi-square value, the greater the discrepancy between the observed data and the expected distribution. The null hypothesis is rejected if $p \geq 1 - \epsilon$ for some significance ϵ, where:

$$p = F(\chi^2; k) \tag{4.9}$$

where $F(x; k)$ is the cumulative distribution function of the chi-square distribution with k degrees of freedom, defined by:

$$F(x; k) = \frac{\gamma(k/2, x/2)}{\Gamma(k/2)} \tag{4.10}$$

In this thesis, $\epsilon = 0.05$ will be used.

4.2 Objective Functions

Four different objective functions have been chosen for the evaluation of the MIMH algorithm.

4.2.1 Himmelblau's Function

Himmelblau's function [RRR83] is a two-dimensional multi-modal objective function with three local optima and one global optimum. The function

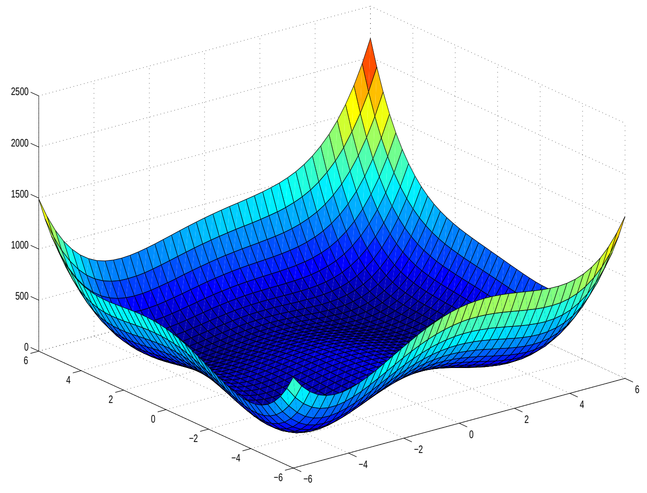


Figure 4.2: Himmelblau's Function

landscape is shown in Figure 4.2. The function is defined by:

$$f(\mathbf{x}) = ({x_1}^2 + x_2 - 11)^2 + (x_1 + {x_2}^2 - 7)^2 \tag{4.11}$$

Usually, the input vector $\mathbf{x}$ is constrained by requiring that:

$$-6 \leq x_1, x_2 \leq 6 \tag{4.12}$$

The global optimum is located at $\mathbf{x} = (3, 2)^{\mathrm{T}}$.

The landscape defined by Himmelblau's function is largely flat, with ridges at the edges of the defined range ($x_1 = -6 \vee x_1 = 6 \vee x_2 = -6 \vee x_2 = 6$) that have very high values. Because the probability density function $\pi(\mathbf{x})$ is obtained by inverting and linearly scaling the objective function, this translates into a large area with very high probability value. In fact, $\sim 60\%$ of the area has a probability value $\pi(\mathbf{x}) > 0.9$, and $\sim 99\%$ of the area has a probability value $\pi(\mathbf{x}) > 0.5$

This property of Himmelblau's function makes it very hard to approximate the global optima of the function by means of importance sampling. The results of the second experiment shall demonstrate that this problem can be overcome by throttling the value of κ.

4.2.2 The Moonface Function

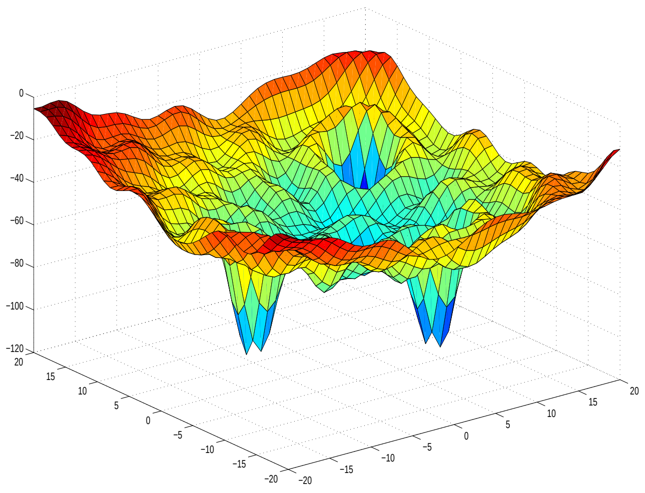


Figure 4.3: The Moonface Function

Jacobs [Jac97] suggested a two-dimensional multi-modal objective function with many local optima. Due to its visual appearance, it will be dubbed the Moonface function in this thesis. Three of the local optima stand out as sharply defined peaks, one of those is the global optimum. The function is defined by:

$$d_1 = \left| \mathbf{x} - \begin{pmatrix} 11 \\ 9 \end{pmatrix} \right|, d_2 = \left| \mathbf{x} - \begin{pmatrix} -11 \\ -3 \end{pmatrix} \right|, d_3 = \left| \mathbf{x} - \begin{pmatrix} 6 \\ -9 \end{pmatrix} \right| \tag{4.13}$$

$$f(\mathbf{x}) = d_1 - \frac{50 \cdot \sin(d_1)}{d_1} + d_2 - \frac{50 \cdot \sin(d_2)}{d_2} + d_3 - \frac{50 \cdot \sin(d_3)}{d_3} - 100 \quad (4.14)$$

The input vector $\mathbf{x}$ is often constrained by:

$$-20 \leq x_1, x_2 \leq 20 \quad (4.15)$$

At the exact positions $(11, 9)^{\mathrm{T}}$, $(-11, -3)^{\mathrm{T}}$, and $(6, -9)^{\mathrm{T}}$ the value of the Moonface function is undefined by grace of division by zero. However, the function value of the neighborhood of these points can be calculated and is Lebesgue measurable to an arbitrarily small distance from these three points. Therefore, these points can be considered to represent the three strongest local optima, and $(6, -9)^{\mathrm{T}}$ can be considered the global optimum.

4.2.3 Multisphere Functions

A Multisphere function is an n-dimensional multi-modal objective function, defined by:

$$f(\mathbf{x}) = \arg\min_{\mathbf{y} \in Y} |\mathbf{x} - \mathbf{y}| \quad (4.16)$$

where Y is the set of global optima.

There are no local optima in a Multisphere function that are not a part of Y.

The input vector $\mathbf{x}$ will be constrained by:

$$\forall x_i : -10 \leq x_i \leq 10 \quad (4.17)$$

The number of global optima and the exact locations of those optima can be chosen randomly or pre-selected. In the experiments in Sections 4.3.2 and 4.3.3, a 2-dimensional and a 4-dimensional Multisphere function are used. For both functions, a fixed set of optima is used, so that the results of this experiment are more easily reproduced.

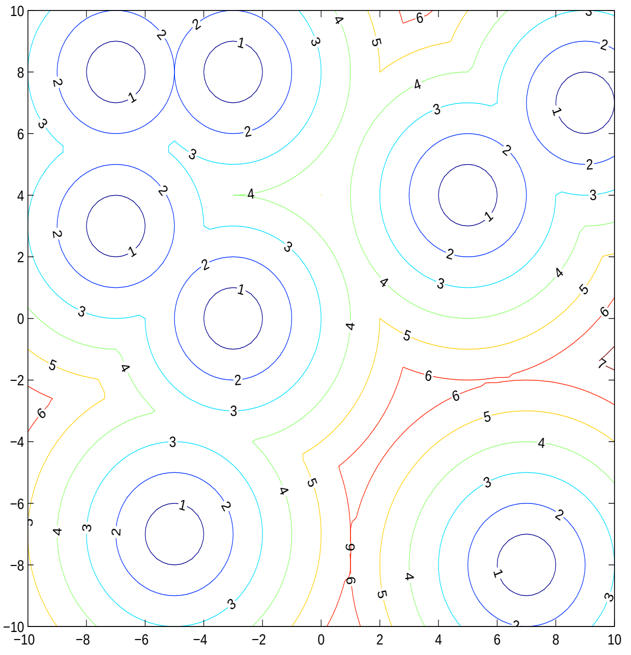


Figure 4.4: A 2-dimensional Multisphere function

These optima were generated by drawing from a uniform random distribution over the domain of the objective function. The optima were re-rolled if the distance between any two optima is less than $\rho = 1$ for the 2-dimensional Multisphere function and $\rho = 3$ for the 4-dimensional Multisphere function, to ensure that the optima are distinguishable from each other to any evaluation metric, specifically the MPR metric.

The optima selected for the 2-dimensional Multisphere function are:

$$Y = \left\{ \begin{pmatrix} -3 \\ 8 \end{pmatrix} \begin{pmatrix} -5 \\ -7 \end{pmatrix} \begin{pmatrix} -7 \\ 8 \end{pmatrix} \begin{pmatrix} 5 \\ 4 \end{pmatrix} \begin{pmatrix} 9 \\ 7 \end{pmatrix} \begin{pmatrix} -7 \\ 3 \end{pmatrix} \begin{pmatrix} -3 \\ 0 \end{pmatrix} \begin{pmatrix} 7 \\ -8 \end{pmatrix} \right\} \tag{4.18}$$

The optima selected for the 4-dimensional Multisphere function are:

$$Y = \left\{ \begin{pmatrix} -4 \\ -9 \\ -8 \\ 6 \end{pmatrix} \begin{pmatrix} 3 \\ -3 \\ 9 \\ -9 \end{pmatrix} \begin{pmatrix} -1 \\ -2 \\ 5 \\ 5 \end{pmatrix} \begin{pmatrix} -6 \\ 0 \\ -1 \\ 2 \end{pmatrix} \begin{pmatrix} 4 \\ 5 \\ -4 \\ 3 \end{pmatrix} \begin{pmatrix} 3 \\ -6 \\ -7 \\ 0 \end{pmatrix} \begin{pmatrix} 9 \\ -3 \\ 1 \\ -5 \end{pmatrix} \begin{pmatrix} 5 \\ -4 \\ 0 \\ 3 \end{pmatrix} \right\} \tag{4.19}$$

An example of a 2-dimensional Multisphere function that uses the optima from Equation 4.18 is shown in Figure 4.4.

4.3 Test Setup

In this section, three experiments will be described that are designed to test the effectiveness of the MIMH algorithm, the effect of the κ parameter, and how the algorithm compares to similar search algorithms.

In the first experiment (Section 4.3.1), it is shown that the distribution of the points sampled by the MIMH algorithm converges to the target distribution. This is done by running the algorithm a large number of times in parallel, and comparing the distribution of the points sampled in each iteration to the calculated distribution of the objective function. Pearson's chi-square test provides an effective means of comparing these distributions.

In the second experiment (Section 4.3.2), the value of κ is used to adjust the greediness of the algorithm, and a self-adaptive scheme for controlling the value of κ is explored. Success is measured in terms of the sum of the distance between each optimum and the nearest sampled point.

In third and final experiment (Section 4.3.3), the MIMH algorithm is compared to an Evolutionary Strategy with Niching [Shi08] and plain uniform random sampling.

These experiments were implemented in MATLAB. The source code of the algorithm and the experiments can be found at:

`http://www.liacs.nl/~jvisser2/` (May 4th, 2010).

4.3.1 Test Convergence to Target Distribution

This experiment serves to explore the relation between the target density $\pi(\cdot)$, and the probability density $\widetilde{\pi}(\cdot)$ obtained by sampling from an estimate of the objective function. The MIMH algorithm formulates a probability density $\widetilde{\pi}_t(\cdot)$ that converges toward $\pi(\cdot)$ as more points are added to the training set.

To measure how quickly the probability distribution of the sampled points $\widetilde{\pi}_t(\cdot)$ converges towards that predicted by the actual objective function $\pi(\cdot)$, it is possible to use Pearson's Chi-Square Test. The values of $\widetilde{\pi}_t(\cdot)$ can be obtained by running the algorithm many times. For selected objective functions $\pi(\cdot)$ can be calculated.

A simple 1-dimensional Sine objective function (see Equation 4.20) will be used, because it is easy to calculate the value of the integral of $\pi(\cdot)$ for that function for any interval.

$$f(x) = 1 + \sin(x) \tag{4.20}$$

The value of x will be limited to $0 \leq x \leq 4\pi$. Pearson's Chi-Square Test requires that the data be divided among cells. In this experiment, the data is divided into 30 cells, with the i-th, cell corresponding to the interval:

$$a_{i-1} \leq x < a_i \tag{4.21}$$

$$a_i = i \cdot 4\pi/30 \tag{4.22}$$

The unnormalized target density associated with this objective function is:

$$g(x) = f_{max} - f(x) = 1 - \sin(x) \tag{4.23}$$

Therefore, the expected distribution of N sampled points among them intervals is given by simple integration of the target density:

$$E_i = c \cdot (a_i + \cos(a_i) - a_{i-1} - \cos(a_{i-1})) \tag{4.24}$$

where $i \in [1, 2 \dots 30]$ and c is a normalizing constant defined by the number of sampled points over all cells divided by the integral of the whole interval:

$$c = \frac{N}{(4\pi + \cos(4\pi) + 1)} \tag{4.25}$$

The MIMH algorithm is run $N = 1000$ times, each run with $t_{max} = 200$. For each iteration t, the sampled points from the N runs are aggregated into a set X_t. In effect, X_1 is composed of the first sampled points from all N runs, and $X_{t_{max}}$ is composed of the last sampled points from all runs.

For this experiment, the parameter κ is kept at the default value of 1, since any other value would skew the observed distribution. Both σ and l are also set at the default values suggested in Sections 3.3.1 and 3.3.2.

For each iteration t, the observed distribution $O_{t,i}$ is the number of sampled points $x \in X_t$ where $a_{i-1} \leq x < a_i$. The distribution among cells found in O_t is compared to the expected distribution E for goodness-of-fit using Pearson's Chi-Square Test. The result is a p-value for each iteration that reflects the chance that the observed distribution meets the null-hypothesis, i.e. the expected distribution.

In Section 5.1 these p-values are plotted.

4.3.2 Test Effect of Probability Exponent

This experiment is intended to explore the effect of the value of κ on the speed at which the algorithm converges on the optima. As discussed in Section 3.3.3, the default value for κ in the MH algorithm is 1. A higher value should make the algorithm more greedy, because it increases the factor difference between any two fitness values.

The MIMH algorithm is run on four objective functions: Himmelblau's function, the Moonface function, a 2-dimensional Multisphere function with the optima given in Equation 4.18 and a 4-dimensional Multisphere function with the optima given in Equation 4.19. These represent three very specific cases, where the value of κ could have a distinct effect on the performance of the algorithm.

- Himmelblau's function has a largely flat landscape with relatively very high ridges at the edges. Since the PDF is a linearly scaled inversion of the objective function, this translates into an almost uniformly high distribution function. A higher value of κ should help accentuate the minima.

- The Moonface function features three sharp valleys with extreme fitness values, and many local minima. Increasing κ will increase the chance of an MH walk stranding in one of the local minima.

- The Multisphere functions have no such extreme features as the previous two objective functions. They consist of very smooth transitions from low to high fitness values. All minima in the Multisphere functions are global optima; therefore an increase in κ should make the algorithm more exploitative, less explorative.

The values of κ that are explored shall be limited to $\kappa \in K$ with:

$$K = (1, 2, \ldots, 100) \tag{4.26}$$

Both σ and l are also set at the default values suggested in Sections 3.3.1 and 3.3.2.

For each $\kappa \in K$, the objective function is exposed to the MIMH algorithm for $t_{max} = 100$ iterations. At the end of each run, the HRPC and MPR values are calculated for the sample set. The former value shall serve as an indicator of how well the algorithm explores the high-performance regions of the objective function, the latter as an indicator of the effectiveness of the algorithm to find the global minima.

The niche radius required by the MPR calculation is set at 0.5 for Himmelblau's function, 1.0 for the Moonface and the 2-dimensional Multisphere function, and 3.0 for the 4-dimensional Multisphere function.

The results are plotted in Section 5.2.

4.3.3 Comparison to Other Approaches

Here the performance of the MIMH algorithm is compared to that of a Niching CMA-ES algorithm and plain uniform random sampling. The algorithms are applied to three objective functions, Himmelblau's function, the Moonface function and the 4-dimensional Multisphere function with the optima given in Equation 4.19.

Each is allowed $t_{max} = 200$ objective function evaluations. As an illustration of the effect of the κ parameter, the MIMH algorithm is included with the default value of $\kappa = 1$, as well as the ideal value of κ obtained in Section 5.2.

The four algorithms applied are:

1. The Niching CMA-ES using a $(1 + 10)$ approach with $q = 4$ expected

number of peaks and $q + p = 8$ D-sets; the individuals in the $(q + 1)$-th $\ldots (q + p)$-th D-set are randomly regenerated every κ-th generation with $\kappa = 10$. The specific implementation of the algorithm used is that of Shir [Shi08], which can be found at:

`http://www.liacs.nl/~oshir/code/` (February 28th, 2010).

Shir [Shi08] recommends using a niche radius ρ such that the combined volume of q hyperspheres of radius ρ is equal to the volume of the hypersphere that contains the sample space, or:

$$\rho = \sqrt{\frac{d}{2}} \cdot (x_{max} - x_{min}) \cdot q^{-1/d} \tag{4.27}$$

with the caveat that the minimum distance between two minima must be at least 2ρ.

For all objective functions used, this caveat is more restrictive than the recommended value for ρ. Thus, $\rho = 3.8$ is used for the Moonface function, $\rho = 1.9$ is used for Himmelblau's function, and $\rho = 2.6$ is used for the 4-dimensional Multisphere function.

2. Uniform random samples obtained by independent draws from:

$$\mathbf{x} = \mathrm{U}(x_{min}, x_{max})^d \tag{4.28}$$

3. The MIMH algorithm using the standard value of $\kappa = 1$. Both σ and l are also set at the default values suggested in Sections 3.3.1 and 3.3.2.

4. The MIMH algorithm using the ideal value of κ obtained from the results in Section 5.2. The ideal value of κ is specific both to the objective function and to the metric used. Both σ and l are set at the default values suggested in Sections 3.3.1 and 3.3.2.

Each algorithm is run independently $N = 200$ times until it has sampled $t_{max} = 200$ samples. $X_{i,t}$ is the result set of the i-th run after sampling exactly t points. The MPR and HPRC values are calculated for all $X_{i,t}$ and the average over all runs is obtained by:

$$\text{M}\bar{\text{P}}\text{R}_t = \frac{1}{N} \sum_{i \leq N} \text{MPR}(X_{i,t}) \tag{4.29}$$

$$\text{H}\bar{\text{P}}\text{RC}_t = \frac{1}{N} \sum_{i \leq N} \text{HPRC}(X_{i,t}) \tag{4.30}$$

These results are plotted in Section 5.3.

Chapter 5

Results

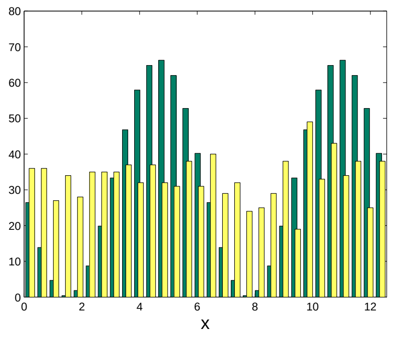
5.1 Test Convergence to Target Distribution

Figures 5.1(a) through 5.1(d) illustrate the evolution of the distribution of sampled points over 100 iterations. It is evident from the figures that the distribution shifts from a largely uniform distribution to the target sinusoïd distribution.

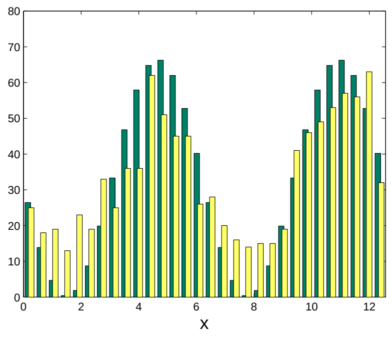
This shift can be explained by the increase in information that the model contains about the shape of the objective function. At $t = 1$, the model will have been trained on an initial population of $\mu_0 = 2$ randomly selected points. As more points are added to the training set, the shape of the estimate distribution will converge on the target distribution.

Figure 5.2 further illustrates this convergence. It shows the χ^2 value for the points sampled at $t = [1 \ldots 200]$, when compared to the target distribution. There is a clear downward trend in the χ^2 values, despite the large fluctuations. This trend indicates a convergence to the target distribution.

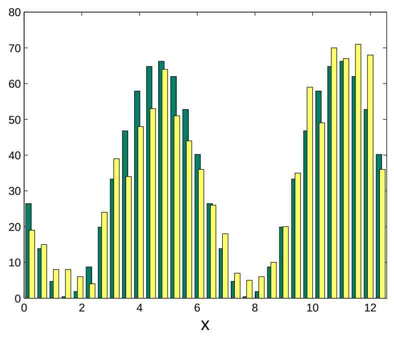
For a test with $k = 29$ degrees of freedom, the χ^2-value that corresponds with $p \leq 0.95$ can be numerically approximated to $\chi^2_{threshold} \approx 42.5570$. This value is represented in Figure 5.2 as a threshold line. The null hypothesis



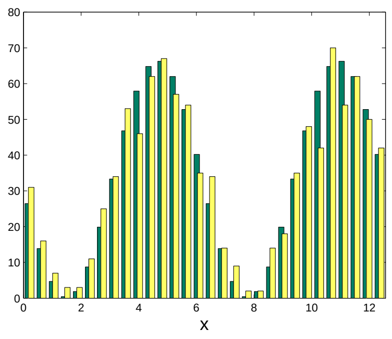
(a) At $t = 1$



(b) At $t = 5$



(c) At $t = 10$



(d) At $t = 100$

Figure 5.1: Observed and expected distributions of sampled points at iterations $t = [1, 5, 10, 100]$. The yellow bars represent the observed distributions; the green bars represent the expected distributions.

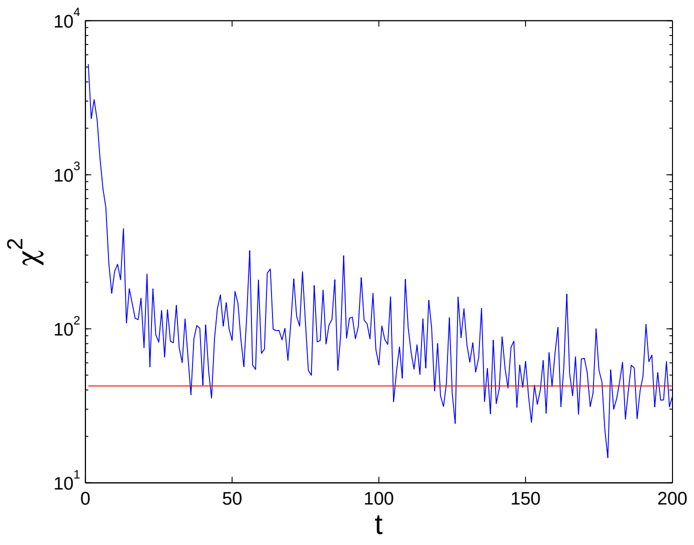
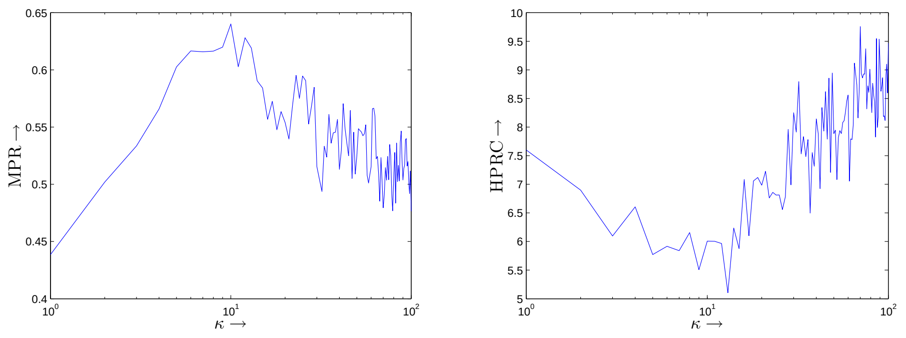


Figure 5.2: χ^2 value for observed points sampled at each iteration, when compared to the expected distribution. The red line indicates $\chi^2_{threshold}$.

that the observed data X_t is consistent with the expected distribution is accepted only when $\chi^2 \leq \chi^2_{threshold}$.

Figure 5.2 reveals a downward trend in the χ^2-value. In the first 100 iterations, all but 2 observed distributions are rejected; in the next 100 iterations, 37 observed distributions are accepted.



(a) MPR (b) HPRC

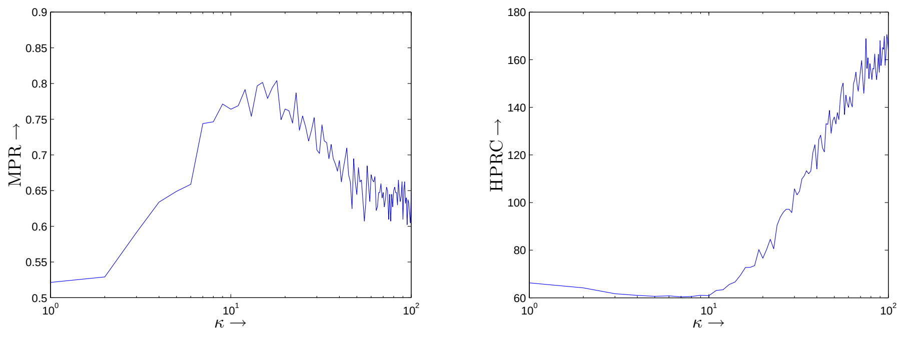
Figure 5.3: Average MPR & HPRC value of 100 points sampled from the Moonface function by the MIMH algorithm using different values of κ.

5.2 Test Effect of Probability Exponent

The MIMH algorithm was run on the Moonface function, Himmelblau's function and two Multisphere functions with $\kappa \in [1 \ldots 100]$, all other parameters being equal. This was repeated 100 times. The average resulting HPRC and MPR values are plotted in Figures 5.3, 5.4, 5.5, and 5.6.

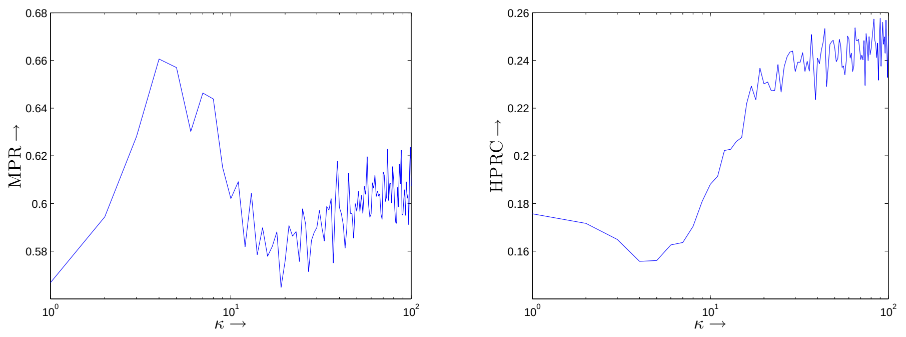
Despite the large fluctuations inherent in small samples, it is clear that there is a peak in the average MPR value after 100 iterations at $\kappa \approx 9$ for the Moonface function, $\kappa \approx 15$ for Himmelblau's function, $\kappa \approx 4$ for the 2-dimensional Multisphere function, and $\kappa \approx 5$ for the 4-dimensional Multisphere function. Higher values of κ inhibit the algorithm from exploring the sample space and finding the optima, whereas at lower values of κ the algorithm is not greedy enough.

Similarly, the average HPRC value is minimized at $\kappa \approx 12$ for the Moonface function, $\kappa \approx 9$ for Himmelblau's function $\kappa \approx 6$ for the 2-dimensional Multisphere function, and $\kappa \approx 5$ for the 4-dimensional Multisphere function. However, lower values of κ do not cost much high-performance region coverage on Himmelblau's function. Since a very large part of Himmelblau's



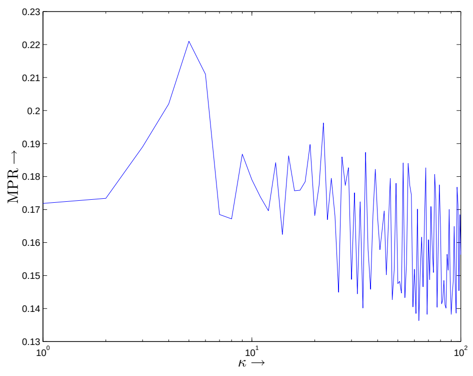
(a) MPR (b) HPRC

Figure 5.4: Average MPR & HPRC value of 100 points sampled from Himmelblau's function by the MIMH algorithm using different values of κ.

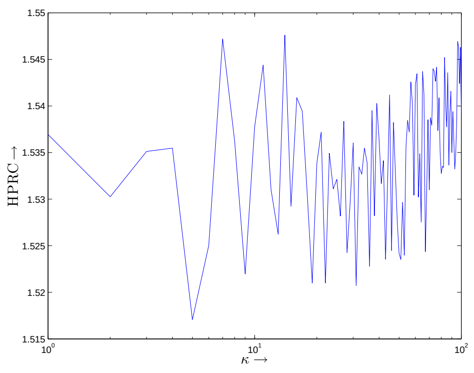


(a) MPR (b) HPRC

Figure 5.5: Average MPR & HPRC value of 100 points sampled from the Multisphere function by the MIMH algorithm using different values of κ.



(a) MPR



(b) HPRC

Figure 5.6: Average MPR & HPRC value of 100 points sampled from the 4-dimensional Multisphere function by the MIMH algorithm using different values of κ.

function consists of high-performance regions, any explorative approach will perform well on the HPRC.

There is an optimal region of κ for sampling 100 points from the objective functions used in this experiment, which is specific to that objective function and the metric to be optimized.

5.3 Comparison to Other Approaches

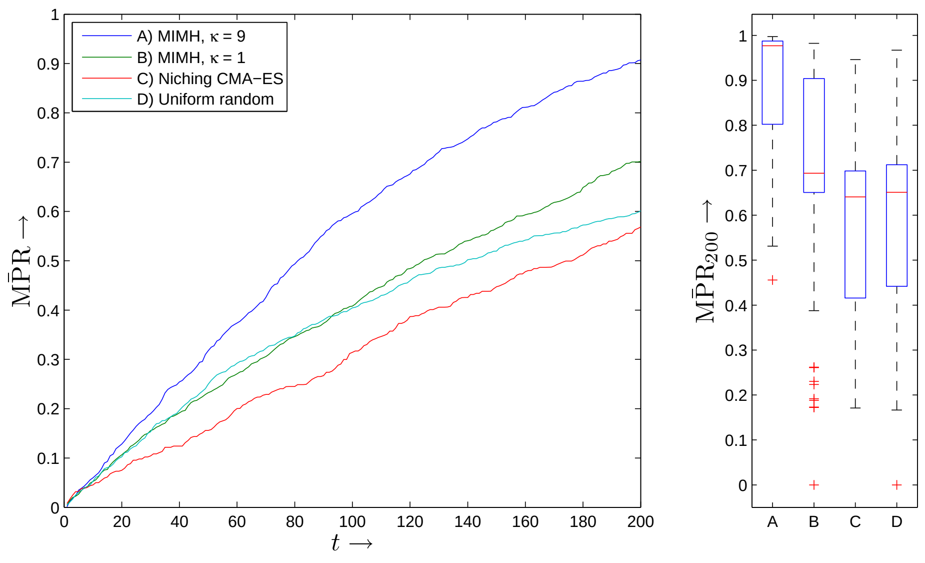


Figure 5.7: $\bar{\text{MPR}}_t$ value of the first 200 points sampled from the Moonface function and the distribution of $\bar{\text{MPR}}_{200}$ using 4 different approaches.

Figures 5.7, 5.9 and 5.11 show the evolution of the average MPR value as more points are sampled from the Moonface function, Himmelblau's function and the 4-dimensional Multisphere function using 4 different approaches.

It is clear from these figures that the MIMH algorithm approaches the minima of the objective functions significantly faster than either uniform random sampling or a Niching CMA-ES. If the ideal value of the κ parameter is used, the performance of the algorithm is enhanced greatly. However, determining the ideal value of κ may be a problem that requires much more computational power and more objective function evaluations than the MIMH algorithm itself.

The $(1 + 10)$ Niching CMA-ES with $(q + p) = 8$ D-sets that was used generates 10 initial parents and 80 offspring per generation. Therefore, less

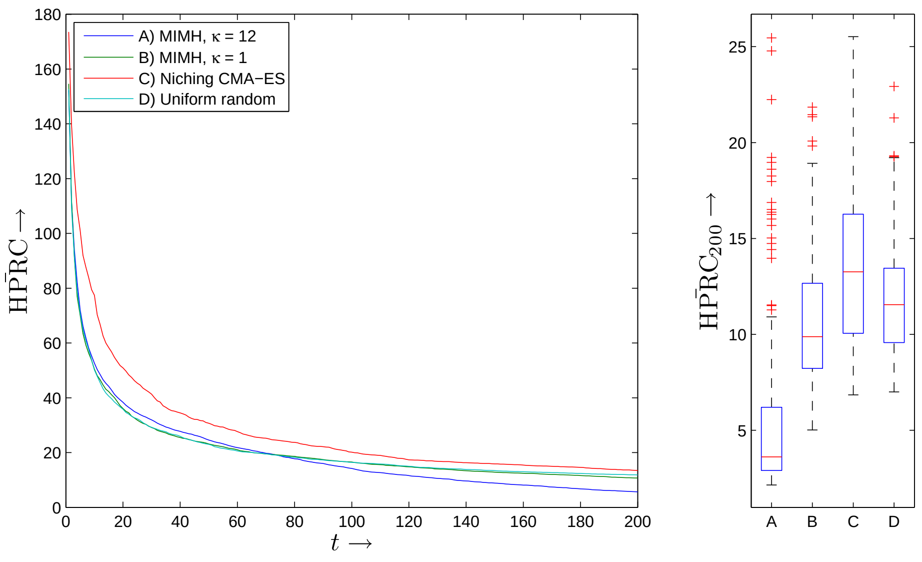


Figure 5.8: HP$\bar{\text{R}}$C$_t$ value of the first 200 points sampled from the Moonface function and the distribution of HP$\bar{\text{R}}$C$_{200}$ using 4 different approaches.

than 3 full generations of offspring can be created in this experiment to satisfy the maximum number of 200 objective function evaluations.

Similarly, Figures 5.8, 5.10 and 5.12 show the evolution of the average HPRC value as points are sampled from the Moonface and Himmelblau's function.

Here the MIMH algorithm performs only slightly better than uniform random sampling. On the Moonface function, only the approach with the ideal value of κ gives a significantly better coverage of the high-performance regions than the other three approaches. On the other two objective function, uniform random sampling and the MIMH algorithm perform roughly equal, while the Niching CMA-ES lags behind.

The average execution time of these approaches is measured and shown in table 5.1.

Even for as few as 200 samples, the MIMH algorithm is roughly 4000

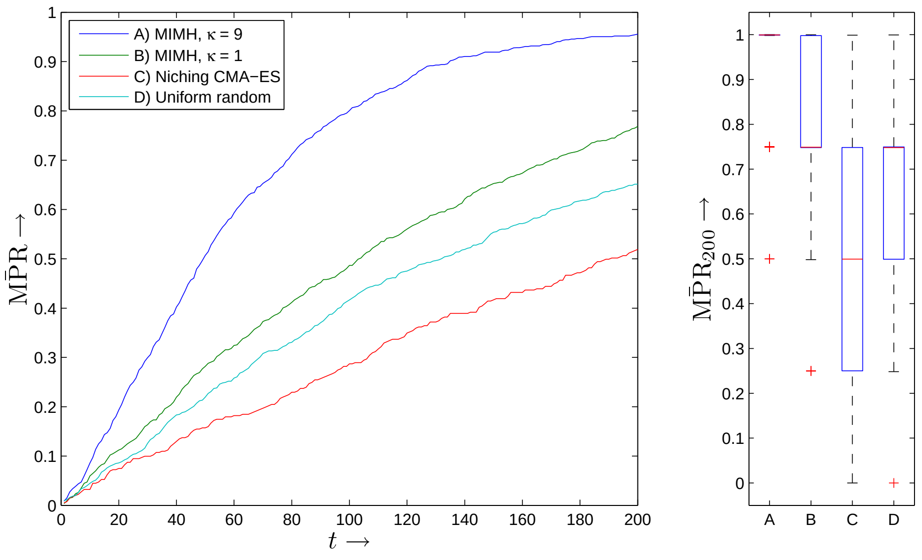


Figure 5.9: $\bar{\text{MPR}}_t$ value of the first 200 points sampled from Himmelblau's function and the distribution of $\bar{\text{MPR}}_{200}$ using 4 different approaches.

times more expensive than a Niching CMA-ES. However, this is offset by the faster convergence to the minima of the objective function and the greater coverage of the high-performance regions.

At $t = 200$ on the Moonface function, the MIMH algorithm achieves $\bar{\text{MPR}} \approx 0.7014$ if $\kappa = 1$ and $\bar{\text{MPR}} \approx 0.9072$ if $\kappa = 9$. When the experiment is extended to $t_{max} = 1000$, it is possible to observe the number of iterations until the Niching CMA-ES or uniform random sampling to arrive at similar $\bar{\text{MPR}}$ values.

If, for a given algorithm and objective function,

$$t(x) = \arg\min_{t>0} |\bar{\text{MPR}}_t \geq x \tag{5.1}$$

then $t(0.7014)$ and $t(0.9072)$ for the Niching CMA-ES and for uniform random sampling are shown in Table 5.2.

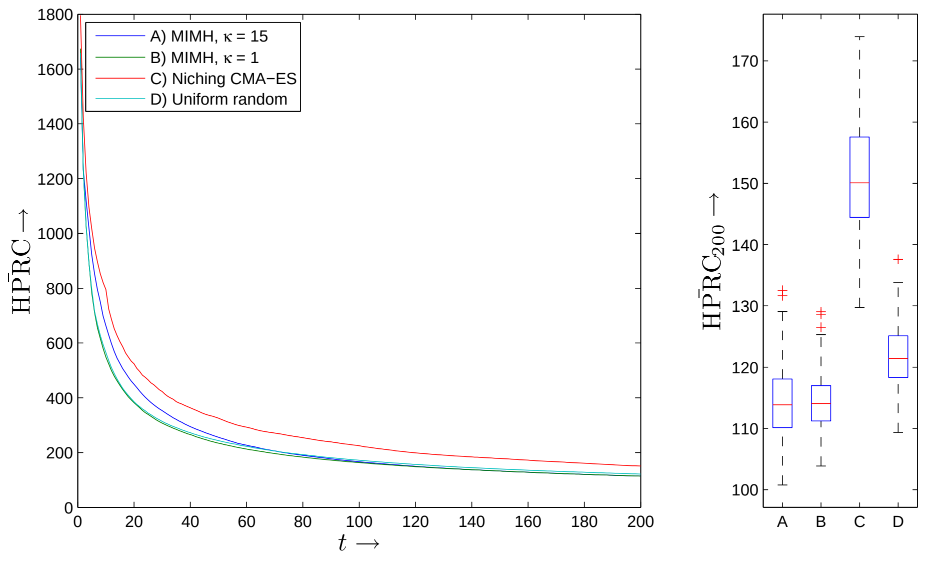


Figure 5.10: $\mathrm{HP\bar{R}C}_t$ value of the first 200 points sampled from Himmelblau's function and the distribution of $\mathrm{HP\bar{R}C}_{200}$ using 4 different approaches.

Similarly, at $t = 200$ on Himmelblau's function, the MIMH algorithm achieves $\mathrm{M\bar{P}R} \approx 0.7687$ if $\kappa = 1$ and $\mathrm{M\bar{P}R} \approx 0.9556$ if $\kappa = 15$. Table 5.3 shows the number of iterations required by the other approaches to arrive at the same $\mathrm{M\bar{P}R}$ value.

Finally, at $t = 200$ on the 4-dimensional Multisphere function, the MIMH algorithm achieves $\mathrm{M\bar{P}R} \approx 0.3586$ if $\kappa = 1$ and $\mathrm{M\bar{P}R} \approx 0.3731$ if $\kappa = 5$. Table 5.4 shows the number of iterations required by the other approaches to arrive at the same $\mathrm{M\bar{P}R}$ value.

Using the MIMH algorithm may be worthwhile if the cost of the additional objective function evaluations is greater than the execution cost of the MIMH algorithm.

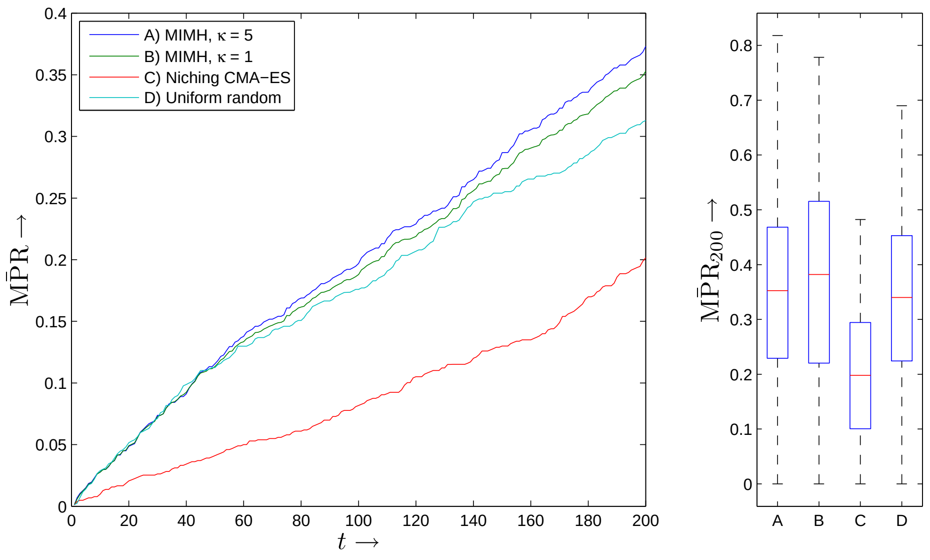


Figure 5.11: $\bar{\text{MPR}}_t$ value of the first 200 points sampled from the 4-dimensional Multisphere function and the distribution of $\bar{\text{MPR}}_{200}$ using 4 different approaches.

	Moonface	Himmelblau	Multisphere 4d
MIMH, $\kappa = 1$	107.2 s	116.9 s	
MIMH, $\kappa = 5$			133.1 s
MIMH, $\kappa = 9$	114.4 s	120.7 s	
MIMH, $\kappa = 12$	113.4 s		
MIMH, $\kappa = 15$		116.9 s	
Niching CMA-ES	0.029 s	0.028 s	0.038 s
Uniform random sampling	0.001 s	0.001 s	0.001 s

Table 5.1: Average execution time of the algorithms used

x	$t(x)$ for Niching CMA-ES	$t(x)$ for Uniform Random
0.7014	320	309
0.9072	743	574

Table 5.2: Number of iterations required to arrive at a given average MPR value on the Moonface function.

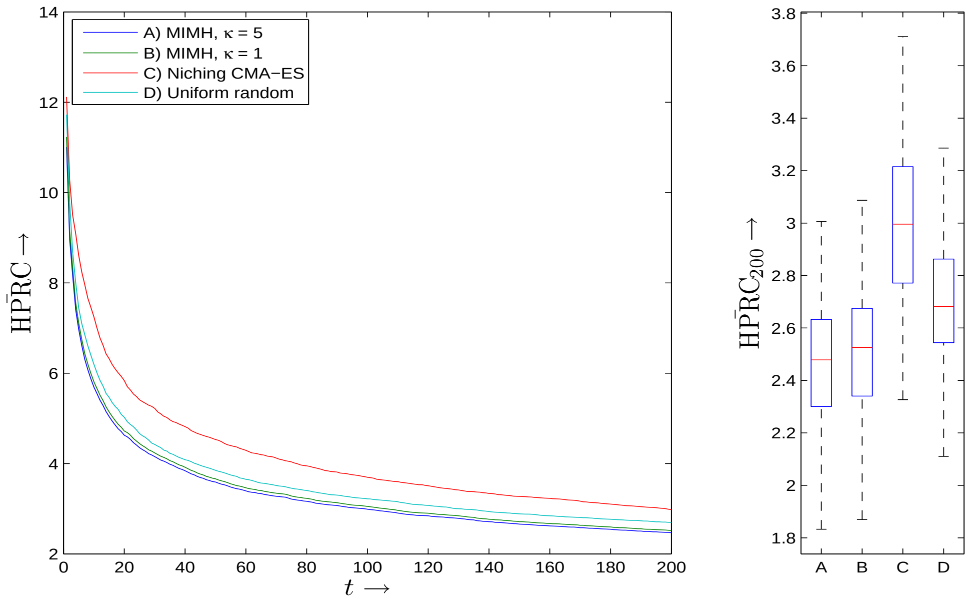


Figure 5.12: $\text{HP}\bar{\text{R}}\text{C}_t$ value of the first 200 points sampled from 4-dimensional Multisphere function and the distribution of $\text{HP}\bar{\text{R}}\text{C}_{200}$ using 4 different approaches.

x	$t(x)$ for Niching CMA-ES	$t(x)$ for Uniform Random
0.7687	270	244
0.9556	634	626

Table 5.3: Number of iterations required to arrive at a given average MPR value on Himmelblau's function.

x	$t(x)$ for Niching CMA-ES	$t(x)$ for Uniform Random
0.3586	377	238
0.3731	398	261

Table 5.4: Number of iterations required to arrive at a given average MPR value on the 4-dimensional Multisphere function.

Chapter 6

Main Conclusion and Outlook

The MIMH algorithm can be an efficient method for obtaining a small sample set that provides good coverage of the high-performance regions of the sample space, and offers good starting points for more greedy local searches for the optima of an objective function. The experiment in Section 5.3 shows that the MIMH algorithm performs significantly better in these respects than a standard Niching CMA-ES or plain uniform random sampling on three test cases.

The efficiency of the MIMH algorithm is compounded by choosing the optimal value of the probability exponent parameter κ. Section 5.2 reveals that there is an optimal region of values for κ, which is specific to the objective function and the target metric. However, finding this optimal region by trial-and-error is a very costly process. Future research might be able to discover a cheap method for estimating the optimal value of κ.

The execution time of the MIMH algorithm is high, and scales to the order of $O(n^3)$ where n is the number of data points sampled. This restricts it usefulness to costly objective functions, where the cost of the objective function evaluations saved outweighs the cost of the algorithm.

In addition to global optimization, the MIMH algorithm can also serve as a method for sampling from an arbitrary distribution, similar to the MH algorithm. Section 3.4 proves that the samples drawn from the estimate probability density converge on the target density. Section 5.1 demonstrates that convergence can occur in relatively few iterations.

It would be interesting to further examine the coverage of high-performance regions as an alternate goal for global optimization methods, providing a different perspective than other, more optimum-centric metrics. There are many other ways to define the concept of high-performance regions than the one employed in this thesis. Ultimately, the definition of high-performance regions for a given problem would have to closely follow the preference of the experts in its field.

As an alternative approach, rather than employing a global meta-model of the objective function, the MIMH algorithm can be adapted to use a meta-model local to each sample for which the objective function value is estimated. This approach could significantly reduce the cost of training the RBFN at higher iterations.

Nomenclature

$\Lambda(S)$	Lebesgue measure of set S, page 33
$\mathrm{N}(\mu, \sigma)$	Normal distribution with mean μ and variance σ^2, page 10
$\mathrm{U}(a, b)^d$	Sample from uniform random distribution over $[a \dots b]^d$; $d = 1$ unless otherwise specified, page 7
σ	Step variation, page 10
$\widetilde{\beta}$	Generalized least squares estimate, page 15
d	Dimensionality of the sample space, page 6
f_{max}	Maximum objective function value, page 8
f_{min}	Minimum objective function value, page 8
S	Sample domain, page 8
x_{max}	Upper bound of sample domain, page 8
x_{min}	Lower bound of sample domain, page 8

Metropolis-Hastings Algorithm

$\alpha(\mathbf{x}_t, \mathbf{y})$	Proposal acceptance function, page 7
$\mathbf{x}_0$	Starting point for MH walk, page 6
$\mathbf{x}_t$	t-th point in MH walk, page 6
$\mathbf{y}$	Candidate point, page 6
π	Target distribution, page 6
$g(\mathbf{x})$	Unnormalized target density function, page 6
k	Normalizing constant, page 6
l	Length of MH walk, page 6

$Q(\mathbf{x}_t)$	Proposal density function, page 6

Radial Basis Function Network

$\mathbf{c}_n$	Center of the n-th basis function, page 10
$\mathbf{r}$	Radius of basis functions, page 12
$\phi(x)$	Basis function, page 10
$\widehat{w}$	Global bias value, page 11
$h(\mathbf{x})$	Output of the RBFN, page 11
w_n	Weight value for the n-th basis function, page 11

MIMH Algorithm

$\bar{f}(\mathbf{x})$	Scaled fitness value, page 20
κ	Probability exponent, page 25
μ_0	Size of initial training set, page 18
$\widetilde{f}(\mathbf{x})$	Objective function estimate, page 18
$\widetilde{f}_{max}$	Observed maximum fitness, page 19
$\widetilde{f}_{min}$	Observed minimum fitness, page 20
$\widetilde{g}(\mathbf{x})$	Unnormalized density function estimate, page 18

High-Performance Region Coverage

Υ	Comparison set, page 33
$\upsilon(X)$	High-performance region coverage, page 33
$\widehat{\Lambda}(S)$	Approximated Lebesgue measure of S, page 33
$\widehat{\upsilon}(X)$	Approximated high-performance region coverage, page 34
$d(\mathbf{x}, X)$	Euclidean distance to nearest neighbor in X, page 33
$f_{threshold}$	Threshold fitness value, page 33
T	High-performance set, page 32

Pearson's Chi-Square Test

χ^2	Chi-square value, page 37
E	Expected distribution, page 44

E_i	Expected number of samples in i-th cell, page 36
$F(x;k)$	Cumulative chi-square distribution function, page 37
k	Degrees of freedom, page 37
O	Observed distribution, page 44
O_i	Observed number of samples in i-th cell, page 36
p	Chance that null hypothesis is correct, page 37
X_t	Aggregated sampled point for iteration t, page 44

References

[BFM97] Thomas Bäck, D. B. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. CRC Press, 1997.

[GP89] Federico Girosi and Tomaso Poggio. Networks and the best approximation property. 1989.

[GS90] Alan E. Gelfand and Adrian F.M. Smith. Sampling based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85:398–409, 1990.

[Has70] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57:97–109, 1970.

[Jac97] Christian Jacob. *Principia Evolvica: Simulierte Evolution mit Mathematica*. dpunkt-Verlag, 1997.

[KGV83] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science, New Series*, 220(4598):671–680, May 1983.

[KO96] J.R. Koehler and A.B. Owen. Computer experiments. *Handbook of Statistics*, 13:261–308, 1996.

[MRR$^+$53] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[MS96] Brad L. Miller and Michael J. Shaw. Genetic algorithms with dynamic niche sharing for multimodal function optimization. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC'96)*, pages 786–791, New York, NY, USA, 1996.

[Orr96] Mark J. L. Orr. *Introduction to Radial Basis Function Networks.* Centre for Cognitive Science, University of Edinburgh, 2, Buccleuch Place, Edinburgh EH8 9LW, Scotland, 1996.

[Pea92] Karl Pearson. The grammar of science. *Nature*, 46:247–, 1892.

[Raw88] John O. Rawlings. *Applied Regression Analysis: a Research Tool.* Wadsworth & Brooks/Cole Advanced Books and Software, 1988.

[RR01] Gareth O. Roberts and Jeffrey S. Rosenthal. Optimal scaling for various metropolis-hastings algorithms. *Statistical Science*, pages 351–367, 2001.

[RRR83] G.V. Reklaitis, A. Ravindran, and K.M. Ragsdell. *Engineering Optimization: Methods and Applications.* Wiley-Interscience, New York, 1983.

[Shi08] Ofer Shir. *Niching in Derandomized Evolution Strategies and its Applications in Quantum Control; A Journey from Organic Diversity to Conceptual Quantum Designs.* PhD thesis, Universiteit Leiden, 2008.

[Sil86] B. W. Silverman. *Density Estimation for Statistics and Data Analysis, Monographs on Statistics and Applied Probability 26.* CRC Press, 1986.

[Tie94] Luke Tierney. Markov chains for exploring posterior distributions. *Annals of Statistics*, 22:1701–1762, 1994.

[Zab03] Zelda B. Zabinsky. *Stochastic Adaptive Search for Global Optimization.* Springer, 2003.