DESIGN OF A SURROGATE ASSISTED $(1 + 1)$-ES

by

Arash Kayhani

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
December 2018

Table of Contents

List of Tables

List of Figures

Abstract

The information gained from previous iterations of an evolution strategy (ES) can be used to create a surrogate model based on the real objective function. While surrogate models are not as accurate as objective functions, they could distinguish more promising candidate solutions. To develop a better understanding of surrogate-assisted ESs, we simulate the behavior of a surrogate-assisted $(1 + 1)$-ES on the quadratic sphere. These simulations are made using a noisy objective function as the surrogate model. We introduce some measures to quantify the trade-off of saving expensive objective function evaluations at the cost of taking poorer steps. Using these findings, we present a mechanism to adapt the step-size based on model accuracy. We empirically evaluate the performance of this step-size adaptation mechanism in surrogate-assisted $(1 + 1)$-ES and compare it to that of the canonical $(1 + 1)$-ES on several simple test functions.

List of Abbreviations and Symbols Used

D	$\sqrt{n+1}$
C	Covariance matrix
I	Identity matrix
K	Covariance function
N	Probability density function of a normal distribution
Φ	Cumulative distribution function of the standard normal distribution
η	The average fitness improvement in each iteration of the original fitness evaluation in SA-ES
$\hat{f}$	Model's prediction
$\mathbb{1}$	The indicator function $\mathbb{1}$ is equal to one if the condition of the indicator is true and is equal to zero if the condition is false.
σ	Step-size
σ^*	Normalized step-size
σ^*_ϵ	Normalized noise strength
$\mathbf{x}$	Input space location
$\mathbf{x}_1$, $\mathbf{x}'$	Space location of offspring
$c_1,\ c_2,\ c_3$	Coefficients of SA-ES step-size adaptation mechanism
exp	The exponential function
f	Objective function
l	Length-scale
m	Mean function
p	Probability density function
p_{succ}	Success rate
$p_{truepositive}$	Probability of an offspring being superior to its parent based on the true fitness function
q^*	Normalized fitness gain

q_ϵ^*	Noisy fitness gain
t	Time or iteration index
A	Least square error function
R	Distance between parent and the location of the optimizer
$\boldsymbol{\beta}$	Parameters of a model
ϵ	Error
κ	Noise-to-signal ratio
$\tilde{\mathbf{x}}$	Vandermonde matrix
∇f	The gradient of f
$\sigma^{(0)}$	The initial value of the step-size
σ_ϵ	Noise strength
$\mathbf{z},\ \mathbf{z}_A,\ \mathbf{z}_B$	A vector of n normally distributed elements. See page 10.
$\tilde{C}$	Covariance matrix of the training data and the parameter space location
ξ_i^+, ξ_i^-	See page 26
d	Distance between two points
f	Objective function value
f_{stop}	Stop criteria
h	The distance to move in input space before the function value changes significantly
p_{eval}	Success probability of the model evaluation
$p_{q^*}^{(acc)}$	The probability of an individual offspring being selected by the surrogate model
p_{step}	The probability of an offspring being better than its parent
q	Fitness gain
r	Distance between offspring and location of the optimizer
$x^{(0)}$	The initial value of the step-size
z_1	Standard normally distributed variable

$(\mu/\rho \overset{+}{,} \lambda)$	See page 2
k	Covariance between the parameter space location **x** and the training data
ANN	Artificial neural network
BBOB	Black-Box Optimization Benchmarking
BFGS	Broyden-Fletcher-Goldfarb-Shanno
CMA	Covariance Matrix Adaptation
COBRA	Constrained Optimization By RAdial basis function interpolation
CPU	Central Processing Unit
CSA	Cumulative Step-size Adaptation
EA	Evolutionary Algorithms
ES	Evolution Strategy
GP	Gaussian Process
GPR	Gaussian Process Regression
KDE	Kernel Density Estimation
LWR	Locally Weighted Regression
MLE	Maximum Likelihood Estimation
MM-ES	Meta-Model assisted ES
n	Number of dimensions
P	Training set

RBF	Radial Basis Function
SA-ES	Surrogate-Assisted Evolution Strategy
SVM	Support Vector Machine
SVR	Support Vector Regression

Acknowledgements

I would like to express my sincere gratitude to Prof. Dirk V. Arnold for his support throughout the research. His insightful guidance and contributions to this study are gratefully acknowledged.

Chapter 1

Introduction

1.1 Motivation

Evolution strategies are a class of evolutionary algorithms introduced by Ingo Rechenberg [43] and Hans-Paul Schwefel [47] that have been widely studied for optimization in continuous search spaces. Notably, they are known for their use in the optimization of *black-box problems*, where there is no information about derivatives or differentiability of the problem. One drawback of these algorithms is that they need a high number of evaluations to find the optimal solution and, therefore, they are considered expensive for some real-world problems with expensive objective functions, where a function evaluation might cost much money, time or human resources.

A class of powerful approaches that were employed to enhance the performance of evolution strategies are *surrogate-assisted evolution strategies* (SA-ES). In these approaches, using information gained from previous iterations of the optimization, and by incorporating a modeling technique, the objective function (*fitness function*) can be approximated. This approximated model is cheaper but has lower accuracy compared to the real fitness function. A study by Grefenstette and Fitzpatrick [18], was one of the first where information gained from past iterations was used to benefit the algorithm.

Loshchilov [34] reports a continuous increase in the number of publications in the field of surrogate-assisted evolution strategies since 1990. A survey by Jin [27] highlights various approaches that recently have been suggested to improve the surrogate-assisted evolutionary algorithms. These approaches combined multiple heuristics that reportedly enhanced the performance of the surrogate-assisted evolutionary algorithms, but the precise effects of each of these strategies are not well understood. One aspect of surrogate-assisted evolution strategies that has been relatively underexamined is the finding of the trade-off between using the expensive but accurate fitness function, and cheap but inaccurate surrogate model. This issue plays a crucial

role in the overall performance of the surrogate-assisted evolution strategies as the surrogate modeling is beneficial only if the benefit of the reduced cost of using the model evaluation is more than the drawback from taking poorer steps due to the inaccuracy of the model evaluation.

Additionally, the suggested surrogate-assisted evolution strategies in the literature employ step-size adaption mechanisms that were initially designed for evolution strategies without surrogate model assistance. These step-size adaption mechanisms might work perfectly well when no surrogate model is used, but they may be less than optimal if models are incorporated in the algorithm. Therefore, it is also desirable to develop a new step-size adaptation mechanism designed explicitly for surrogate-assisted evolution strategies, and in correlation with the model uncertainty.

This study reflects our opinion that the research on surrogate-assisted evolution strategies needs to go more in-depth on understanding the behavior of these algorithms and on designing a step-size adaption interface in support of the uncertainty of the surrogate model. In the next sections, we suggest a new approach to develop a better understanding of surrogate-assisted evolution strategies by simulating the behavior of the surrogate model. We use these simulations in an attempt to answer some of the unanswered questions in this field like how much the use of surrogate models benefits the evolution strategies. We also use this approach to improve the design configurations, particularly the step-size adaptation mechanism, for a simple surrogate-assisted evolution strategy.

1.2 Context

1.2.1 Evolution Strategy

An evolution strategy is a generational algorithm that solves the optimization problems by implementing a repeated process of taking consecutive stochastic variations (mutation) and selection procedures. These algorithms are usually recognized in the form of $(\mu/\rho \stackrel{+}{,} \lambda)$- ES, where μ, ρ and λ are, respectively, number of parents, number of parents used in the recombination and number of offspring produced in each generation (in some cases ρ might be unspecified). *'Plus'* and *'comma'* (+ and ,) define the type of selection procedure, where in 'plus'-selection, μ best of parents and

offspring are selected, and in 'comma'-selection, μ best of the offspring are selected for the next generation.

1.2.2 Step-size Adaptions

The parameters of the mutation operator play a key role in the convergence speed and efficiency of the optimization process. The parameters of the mutation operator can be adapted using the following three classes of parameter adaptation mechanisms.

- **The 1/5th Success Rule** Defining success probability as the average probability of having an offspring being superior to its parent, for a symmetric mutation operation on a linear function the success probability of the mutation operator is 50%. Based on the Taylor series approximation of smooth functions, as the step-size decreases, fitness difference becomes more linear and, therefore, the success probability becomes closer to 50%. Step-size is the variance of the mutation. On most functions if step-size goes to very large values, the success rate decreases to zero. As a result, for an isotropic mutation, we can use the success probability to control the relative step-size. Rechenberg [43], by studying two simple test functions (corridor and sphere) showed that the optimal success rates for a $(1+1)$-ES with the isotropic mutation is approximately 1/5. He further used the 20% success probability as a switching point between decreasing and increasing the value of the step-size.

- **Self-Adaptation** In this strategy the mutation parameters are also included in the optimization process and therefore, they also mutate and evolve during the optimization process.

- **Derandomized Self-Adaptation** This strategy, also known as the cumulative step-size adaptation (CSA), employs a weighted sum of successful steps to adapt the current step-size.

1.2.3 Regression and Gaussian Process

Surrogate-assisted evolution strategies are employed using various machine learning algorithms. Loshchilov et al. [35] used Support Vector Regression (SVR). Bajer [8]

used Radial Basis Function (RBF) networks and Gaussian Process. Other modeling techniques like Artificial Neural Networks (ANNs) and Polynomial Regression are also used in [33], [25] and [30]. They were also implemented in discrete platforms like discrete genetic algorithms [46] [9]. These machine learning methods are thoroughly explained in chapter 2.

Based on the definition provided by Rasmussen and Williams [42], a Gaussian Process (GP) is a collection of random variables, where every finite number of those variables has a normal joint distribution. A Gaussian Process Regression (GPR) is a machine learning method that uses the concept of GP to describe a function. In this research, we use a Gaussian process regression for SA-ES. Bajer [8] described GP as "a model with a decent number of parameters" that can regress a variety of different continuous fitness landscapes. GPR is commonly used for surrogate modeling, as they do not have some of the disadvantages of their counterparts. ANNs, for instance, are popular surrogate modeling approaches, but they are more difficult to work with. Specially determining a proper structure for ANNs has been a challenging issue for researchers who used ANNs for optimization. Cross-validation and Maximum Likelihood Estimation (MLE) are the most popular approaches for tuning the main hyperparameters of GP. In the next chapters, we further investigate the process of Maximum Likelihood Estimate as well as its accuracy in tuning the hyperparameters of the surrogate-assisted $(1 + 1)$-ES.

Perhaps closest studies to surrogate-assisted $(1 + 1)$-ES are studies by Ulmer et al. [49] and Oliver Kramer [32], in which they use modeling techniques on the platform of, respectively, $(1 + 1)$-ES and steady-state $(\mu + 1)$-ES. Kramer explores a broad range of possible ways that machine learning algorithms can improve $(1 + 1)$-ES, including dimension reduction and problem visualization, but does not study parameter settings, particularly step-size adaption mechanism of a surrogate-assisted $(1 + 1)$-evolution strategy. In this research, we use a similar approach to Kramer's to implement surrogate modeling. Ulmer et al. [49], implements a surrogate-assisted $(\mu + 1)$-ES with a median selection step-size mechanism, suggested by Wakunda and Zell [50]. Then they compare it with steady-state $(\mu + 1)$-ES with and without median selection, but these step-size adaption mechanisms are not designed to work with surrogate modeling, and it is not clear whether they can find the optimum value of

the step-size.

1.2.4 Contribution and Outline

Contributions

In this research by studying the benefits and limitations of surrogate-assisted evolution strategies (SA-ES) compared to normal evolution strategies, we introduce new approaches to investigate the behavior of SA-ES. We study the parameters that affect the performance of SA-ES such as step-size, training data and the hyper-parameters of the modeling technique. Choosing a proper approach for selecting the training dataset of the modeling technique affects the accuracy and efficiency of the model. We will further discuss these parameters in the next chapter. Our research in SA-$(1+1)$-ES consists of expanding the analysis of the surrogate-assisted evolution strategies and simulating the modeling techniques in SA-ES. For the former, we use $(1+1)$-ES which is much easier to be interpreted compared to other evolution strategies. For the latter, we use a normalized Gaussian error to simulate the behavior of an inaccurate model.

To limit the possible unforeseen side effects of the interactions between the evolution strategy and the machine learning method, in this study, we choose the simplest evolution strategy: $(1+1)$-ES. One potential pitfall for using evolutionary strategies with larger population size is that if λ is larger than optimal, the surrogate modeling can improve the performance of the algorithm by simply selecting a smaller portion of the solutions for the next generation. In this case, the improvement in the performance is because of reducing the population size instead of the surrogate modeling. Furthermore, we use the approach that was suggested by Arnold and Beyer [3] and Arnold [1] to analyze noisy fitness functions. In [2], they assumed that the fitness error is Gaussian. They showed that this assumption has a limited effect on results when the noise function is not Gaussian. These analyses are further explained in chapter 3.

We use the analysis from simulations to find mechanisms for adapting the parameters of the algorithm with regard to the problem dimension. One of the mechanisms that we are most focused on is the step-size adaptation mechanism. We use an approach similar to the 1/5th rule which uses the success probability of the algorithm

to adapt the step-size.

Furthermore, we evaluate the performance of the new designed surrogate-assisted evolution strategy on some simple test functions. These test functions are all unconstrained, non-noisy and uni-modal. That is because we want to investigate only the effects of surrogate modeling on evolution strategies by removing the side effects that any complementary extension can have on these algorithms. For instance, multi-modal and anisotropic problems can be better optimized, if we use evolution strategies with bigger values of μ and λ, or if we use covariance matrix adaptation in addition to the surrogate modeling. These complementary features are further explained in chapter 2.

This research expands our work [29] in surrogate assisted $(1+1)$ evolution strategies, where we used a new mechanism for step-size adaption of surrogate assisted $(1+1)$-ES. In this research we were able to reach a similar degree of speed-up compared to our research [29] on most of the benchmark problems. However, on more ill-conditioned problems such as Quartic function, we were able to improve the algorithm's performance.

The contributions of this thesis are as follows:

- A modification of surrogate-assisted $(1+1)$-ES, is introduced. This modification is based on the Kramer's meta-model assisted ES [32] in which the real function only evaluates solutions that are likely to have a better fitness value based on the surrogate model evaluation. We modify the step-size adaption mechanism and the modeling technique to improve the performance of the surrogate-assisted $(1+1)$-ES on the benchmark problems. We follow an approach similar to 1/5th rule to develop a step-size adaptation mechanism based on the characteristics of the proposed surrogate-assisted ES.

- Theoretical analyses of the behavior of the surrogate-assisted $(1+1)$-ES on a quadratic sphere are derived from simulating the behavior of the inaccurate surrogate model, using the objective function with additive Gaussian noise. These analyses include quality measurements of the optimization algorithm such as fitness gain and success probability (which are defined in chapter 2) in light of the inaccuracy of the surrogate model.

- The proposed strategy is evaluated on a set of simple uni-modal test functions to optimize the strategy's parameters such as the hyper-parameters of the surrogate model and the number of training points, and also to evaluate the quality of the new step-size adaptation mechanism. By comparing the performance of the proposed strategy with a $(1 + 1)$-ES, this research reports to what extent surrogate modeling can improve the performance of $(1 + 1)$-evolution strategies on the benchmark problems.

Thesis Outline

In chapter 2, we further describe related work and the necessary background for this research. Chapter 3 describes my methodology, including analysis to explore surrogate-assisted evolution strategies. This chapter also designs and implements a new step-size adaptation mechanism. In chapter 4 we study the performance of the new surrogate-assisted $(1 + 1)$-ES on some simple uni-modal test functions. Finally, chapter 5 gives an overview of the whole study and future work.

Chapter 2

Background and Related Work

In this chapter, we review the previous studies about evolution strategies, especially the $(1+1)$-ES. Then, we compare the different approaches that have been suggested for surrogate-assisted evolution strategies based on their type of *evolution control* and their type of modeling technique. Evolution control is a mechanism that determines when the model should evaluate the offspring and when the real objective function should evaluate them. We also review surrogate-assisted algorithms that were designed for general cases, where test functions are not spherical and contain certain types of difficulties: multi-modal, constrained and ill-conditioned problems. We do not consider these difficulties for evaluation of the surrogate-assisted $(1+1)$-ES. However, we will review some strategies that were used in surrogate assisted EAs to deal with the difficulties, as in future studies these strategies can be implemented in the surrogate-assisted $(1+1)$-ES to generalize the algorithm for real-world problems.

2.1 Evolution Strategies

One of our main aims in this study is to analyze the behavior of surrogate-assisted ES, and as much as possible eliminate any possible side effects that other features of these algorithms, other than the surrogate modeling, might have on the optimization process. Therefore, we use a simple evolution strategy (ES) on simple optimization problems. The optimization problems can be defined as minimization of an objective function (fitness function) $f : \mathbb{R}^n \to \mathbb{R}$ for any input parameter space location $\mathbf{x} \in \mathbb{R}^n$. As was discussed earlier in the introduction, evolution strategies contain two main processes: selection and variation (mutation). In this section, we introduce different mutation operators, and then we analyze the performance of the $(1+1)$-ES on test functions with spherical fitness countours. These analyses will be further used in chapter 3 to investigate the behavior of a surrogate-assisted $(1+1)$-ES.

2.1.1 Mutation Operators

As a part of every evolution strategy procedure, unbiased variations known as mutations are added to solutions of the parent generation. These variations are derived from a multivariate normal distribution $N(0, C)$, where zero is the mean and $C \in \mathbb{R}^{n*n}$ is the covariance matrix of the mutation. Hansen et al. [21], categorize the mutation operators into three categories.

- **Isotropic** where the covariance matrix is proportional to the identity matrix. Therefore, the mutation can be denoted as $\mathbf{x} + \sigma N(0, I)$, where $\mathbf{x}$ is a solution and $\sigma \in \mathbb{R}^+$ is the step-size.
- **Axis-parallel** where the principal axes of the mutation distribution are parallel to the axes of the coordinate system.
- **General** where there is a mutation operator with a symmetric and positive definite covariance matrix.

In this study, we only use the isotropic mutation operator. That is because isotropic mutation operators are easier to analyze and they perform efficiently on simple test functions like the quadratic sphere. Techniques such as covariance matrix adaptation (CMA) can be used in future studies to transform other test functions to isotropic and near-isotropic test functions and generalize the results of this study to real-world test functions.

2.1.2 Quadratic Sphere Function

The quadratic sphere is a well-known optimization problem whose optimization results have significant importance, while it is simple enough to be analyzed. As simple as this optimization problem is, any optimization algorithm that cannot efficiently solve the quadratic sphere, it most likely cannot solve the other optimization problems as well. Arguably, we can further generalize this test problem using covariance matrix adaptation (which will be discussed in the next section) as CMA in some cases might be able to transform other functions into the sphere. The quadratic sphere can be defined as

$$f(\mathbf{x}) = \sum_{i=1}^{n} (x_i - x_{o_i})^2 \tag{2.1}$$

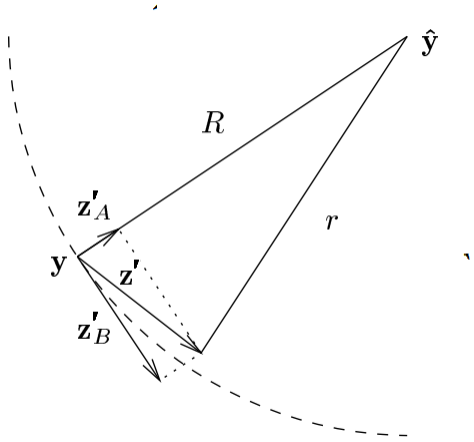

Figure 2.1: This figure from [2] shows $\mathbf{z}'_A = \mathbf{z}_A * \sigma$ and $\mathbf{z}'_B = \mathbf{z}_B * \sigma$ as the two components of the mutation vector $\mathbf{z}'$.

where $\mathbf{x} \in \mathbb{R}^n$ is an input space location and $\mathbf{x}_o \in \mathbb{R}^n$ is the optimizer space location.

2.1.3 The $(1+1)$-ES

In this section, we will review previous studies and theoretical analysis around the $(1+1)$-ES. The $(1+1)$-ES is a simple evolution strategy that unless weighted recombination is considered, is more efficient for the sphere than any other $(\mu/\rho \stackrel{+}{,} \lambda)$-ES. In $(1+1)$-ES, in each iteration, a new offspring is created using a mutation operator. The mutation $\mathbf{x} + \mathbf{z}\sigma$ includes a vector of n normally distributed elements $\mathbf{z}$ and a step-size of σ (also known as mutation strength). In a minimization problem using this definition, the new offspring replaces its parent only if it has a lower objective function value than its parent (Equation 2.2).

$$f(\mathbf{x}) > f(\mathbf{x} + \mathbf{z}\sigma) \tag{2.2}$$

The performance of $(1+1)$-ES can be evaluated using *fitness gain*. Fitness gain q of an input space location $\mathbf{x} \in \mathbb{R}^n$ with mutation vector of $\mathbf{z}\sigma$ is

$$q(\mathbf{z}, \sigma) = f(\mathbf{x}) - f(\mathbf{x} + \mathbf{z}\sigma) \tag{2.3}$$

Using the approach used by Rechenberg [43] the mutation vector can break into two vectors $\mathbf{z}_A$ and $\mathbf{z}_B$, where $\mathbf{z}_A$ is the component of $\mathbf{z}$ in the direction of the negative

gradient $-\nabla f$ at the parent space location, and $\mathbf{z}_B$ is orthogonal to the direction of $\mathbf{z}_A$. We can use the isotropic nature of the mutation to, without loss of generality, transform the vectors in a way that $\mathbf{z}_A = (z_1, 0, ..., 0)^T$ and $\mathbf{z}_B = (0, z_2, ..., z_n)^T$ (Figure 2.1). Using the distance between parent and the location of the optimizer (R) and the distance between offspring and location of the optimizer (r) , in a quadratic sphere we can conclude

$$\begin{aligned} r^2 &= (R - \sigma z_1)^2 + \sigma^2 \|\mathbf{z}_B\|^2 \\ &= R^2 - 2R\sigma z_1 + \sigma^2 z_1^2 + \sigma^2 \|\mathbf{z}_B\|^2 \end{aligned} \tag{2.4}$$

Since $\|\mathbf{z}_B\|^2$ is the sum of squares of $n-1$ standard normally distributed random variables, $\|\mathbf{z}_B\|^2$ has a mean of $n-1$ and a variance of $2(n-1)$. Therefore, when $n \to \infty$, $\frac{\|\mathbf{z}_B\|^2}{n} \to 1$. We can also state that when $n \to \infty$ the value of z_1^2 can be neglected compared to the value of $\|\mathbf{z}_B\|^2$. Therefore, the fitness gain can be written as

$$\begin{aligned} q(\mathbf{z}, \sigma) &= R^2 - r^2 \\ &= 2R\sigma z_1 - n\sigma^2 \end{aligned} \tag{2.5}$$

Furthermore, we can use the normalized mutation strength and normalized fitness gain definitions as it was introduced by Rechenberg [43]

$$\sigma^* = \sigma \frac{n}{R} \tag{2.6}$$

$$q^* = q \frac{n}{2R^2} \tag{2.7}$$

From Equations (2.5), (2.6) and (2.7), it can be concluded that the potential fitness gain of a mutation is

$$q^* = \sigma^* z_1 - \frac{\sigma^{*2}}{2} \tag{2.8}$$

Therefore, the distribution of normalized fitness gain of mutation on a quadratic sphere has the mean of $-\sigma^{*2}/2$ and variance of σ^{*2}, and the probability density function of

$$p_{q^*}(\mathbf{x}) = \frac{1}{\sqrt{2\pi}\sigma^*} \exp\left(-\frac{1}{2} \left(\frac{\mathbf{x} + \sigma^{*2}/2}{\sigma^*} \right)^2 \right) \tag{2.9}$$

Another measurement that can be used to describe the behavior of the $(1+1)$-ES

is the *success probability* of the algorithm. In $(1+1)$-ES, success probability can be defined as the probability of an offspring being superior to its parent. From equation (2.8), the $(1+1)$-ES is only successful if $\sigma^* z_1 - \frac{\sigma^{*2}}{2} > 0$, and therefore $z_1 > \frac{\sigma^*}{2}$. Since z_1 is a standard normal random variable, the success probability of $(1+1)$-ES on a quadratic sphere is

$$\begin{aligned} p_{succ} &= Prob\left[\sigma^* z_1 - \frac{\sigma^{*2}}{2} > 0\right] \\ &= Prob\left[z_1 > \frac{\sigma^*}{2}\right] \\ &= 1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\frac{\sigma^*}{2}} e^{-\frac{1}{2}t^2} dt \\ &= 1 - \Phi\left(\frac{\sigma^*}{2}\right) \end{aligned} \tag{2.10}$$

where Φ is the cumulative distribution function of the standard normal distribution.

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}t^2} dt \tag{2.11}$$

In each iteration the $(1+1)$-ES with the probability of $(p_{succ} = Prob[z_1 > \frac{\sigma^*}{2}])$ has a normalized fitness gain of $\sigma^* z_1 - \frac{\sigma^{*2}}{2}$ and with the probability of $1 - p_{succ}$ has a normalized fitness gain of zero. Therefore, we can find the average normalized fitness gain using the following formula

$$\begin{aligned} E\left[q_+^*\right] &= \frac{1}{\sqrt{2\pi}} \int_{\frac{\sigma^*}{2}}^{\infty} \left(\sigma^* z_1 - \frac{\sigma^{*2}}{2}\right) e^{-\frac{1}{2}z^2} dz_1 \\ &= \frac{\sigma^*}{\sqrt{2\pi}} e^{\frac{-1}{8}\sigma^{*2}} - \frac{\sigma^{*2}}{2}\left[1 + \Phi\left(\frac{\sigma^*}{2}\right)\right] \end{aligned} \tag{2.12}$$

These equations are made for a quadratic sphere, where $n \to \infty$. In the next step, we examine the accuracy of Equations 2.10 and 2.12 for the success probability and the normalized fitness gain on quadratic functions with a finite number of dimensions. In this experiment, we use quadratic spheres with 10 and 100 dimensions. Then, we find the fitness gain for different values of the normalized step-size between zero and ten. We sample 10,000 offspring for each value of σ^*, then, we measure the average

fitness gain and success probability of the samples. The step-size is calculated using the distance of the parent to the optimum R and the normalized step-size definition $\sigma^* = \sigma\frac{n}{R}$. The average normalized fitness gain can be calculated by averaging over the differences between the function values of the successful offspring and their parent. The success probability can be measured by dividing the number of samples that offspring are superior to their parent by the total number of samples of the algorithm.

Figure 2.2 shows the values of the average normalized fitness gain and success probability both for the finite dimensions and the theoretical formulas provided in Equations 2.12 and 2.10. Equations 2.12 and 2.10 represent the fitness gain and success probability for infinite dimension. Figure 2.2 shows that the results of the theoretical and experimental analysis of $(1 + 1)$-ES approximately match and therefore, we can use Equations 2.12 and 2.10 to study the behavior of $(1 + 1)$-ES on the quadratic spheres that n is not ∞.

2.1.4 The 1/5th Rule

In Figure 2.2, the values of the normalized fitness gain for various values of the normalized step-size suggests that the maximum normalized fitness gain for a quadratic sphere is 0.202. This performance can be achieved when the normalized step-size is 1.224 and the success probability of the algorithm is 27%. Rechenberg [43], used the optimum success probability of the quadratic sphere alongside the optimum success probability of the *corridor function*, which is 0.184, to develop a step-size adaption mechanism for $(1 + 1)$-ES. These two functions are both simple but very different from each other. Therefore they represent a range of problems. Corridor function can be defined as

$$f(\mathbf{x}) = \begin{cases} x_1 & if\ |x_i| < 1\ for\ i = 1, 2, ..., n \\ \infty & otherwise \end{cases} \tag{2.13}$$

He argued that $(1 + 1)$-ES reaches its optimum step-size for those test functions when the success probability of the optimization is approximately 1/5. In this approach, known as the 1/5th rule, the step-size is controlled in a way that the success probability remains approximately 20%. Algorithm 1 is an implication of the 1/5th

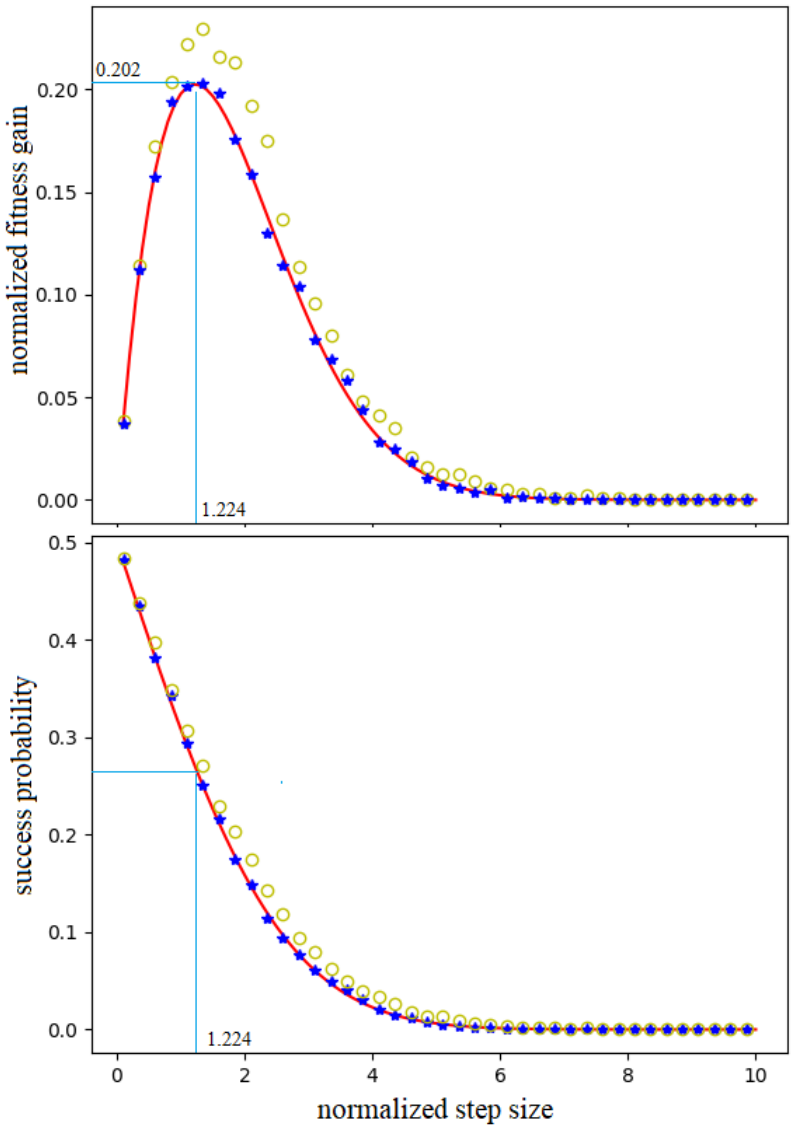

Figure 2.2: Expected fitness gain and success probability of $(1 + 1)$-ES based on theoretical analysis (solid line) and experimental results for $n = 10$ (circles) and $n = 100$ (stars).

rule for $(1+1)$-ES. We introduce this implementation in order to use it in the next chapters to find a similar approach for adapting the step-size of the surrogate assisted evolution strategies. This implementation is based on an approach suggested by Kern et al. [30] where they used an innovative way for the algorithm to reach the 20% success rate. In each iteration of the algorithm , the step-size is increased by the factor of $e^{0.8/D}$ when the offspring is superior to its parent. Otherwise, the step-size is decreased by the factor of $e^{-0.2/D}$. D controls the speed of changes in the step-size value. Hansen et al. [21], suggested that D should be $\sqrt{n+1}$. The indicator function $\mathbb{1}$ is equal to one if the condition of the indicator is true and is equal to zero if the condition is false.

Algorithm 1 $(1+1)$-ES with 1/5th rule

1: initialize $\sigma \in \mathbb{R}^+, \mathbf{x} \in \mathbb{R}^n$
2: **while** not happy **do**
3: $\quad \mathbf{x}_1 \leftarrow \mathbf{x} + \sigma * N(0, I)$
4: $\quad \sigma \leftarrow \sigma \exp^{1/D}(\mathbb{1}_{f(\mathbf{x}_1) \leq f(\mathbf{x})} - 1/5)$
5: $\quad$ **if** $f(\mathbf{x}_1) \leq f(\mathbf{x})$ **then**
6: $\quad\quad \mathbf{x} \leftarrow \mathbf{x}_1$
7: $\quad$ **end if**
8: **end while**

2.2 Surrogate-Assisted Evolution Strategy (SA-ES)

2.2.1 Quality Measurement

Compared to other evolutionary algorithms, surrogate-assisted evolutionary algorithms have the benefit of using the simulated cheap fitness functions instead of the original expensive function. The efficiency of surrogate-assisted evolution strategies can be measured based on the number of fitness evaluations that the algorithm needs to find the optimal solution with a certain accuracy. Based on this definition of efficiency, Büche et al. [13] and Kern et al. [30], respectively reported speed-up by the factor of four to five and two to eight, when they compared their surrogate-assisted evolution strategies with the evolution strategies that do not use the surrogate modeling on uni-modal test functions like quadratic sphere, Schwefel's function. However, for other test functions such as Rosenbrock, Büche et al. reported a smaller factor

of speed-up, and for Rastrigin's function, they stated that the algorithm performs either similar or with less efficiency than its counterparts without the surrogate model. It can be assumed that for some optimization problems model assisted evolutionary algorithms need less number of objective function evaluations to reach the optimal solution. That is because they have an extra selection procedure which eliminates offspring that are not promising based on the picture that fitness prediction develops from the search space.

Unlike the number of function evaluations, CPU cost of the surrogate-assisted evolutionary algorithms might be higher than other evolutionary algorithms. CPU cost of the surrogate-assisted evolutionary algorithms depends on the CPU cost required for building a surrogate model and the number of model evaluations. Although model assisted evolutionary algorithms need less number of function evaluations than normal EAs, model approximation procedure is computationally expensive. Emmerich et al. [16] assert that the CPU cost required for building a surrogate model is not significant compared to the cost for an objective function evaluation in real-world problems.

The main focus of this research is to improve the efficiency of the surrogate-assisted EAs based on their number of fitness evaluations. However, in chapter 3, we redefine and use measures like fitness gain and success rate to study the behavior of surrogate-assisted evolution strategies. In chapter 4, in addition to the number of objective function evaluations, we reduce the number of model evaluations that algorithm needs to select an offspring.

2.2.2 Evolution Control

Evolution control is used to find a trade-off between using the expensive but accurate objective function and cheap but inaccurate model. It determines the number of model evaluations that need to be done, before using the real fitness function. Jin [26] categorized the SA-ESs based on their evolution control technique into three categories: *no evolution control*, *fixed evolution control*, and *adaptive evolution control*.

- **No Evolution Control** No evolution control is when the surrogate model is made from the information that is given at the beginning of the optimization

process. After that, the entire optimization is only based on the information that the surrogate model provides.

- **Fixed Evolution Control** In fixed evolution control, the real objective function is used in every iteration or after a fixed number of iterations that the surrogate model is used to evaluate the offspring. The number of model evaluation per real objective function evaluation in a surrogate-assisted ES is known as *life-length*. Bajer et al. in [10] and [11] used fixed values of 1 to 5 for the life-length.

- **Adaptive Evolution Control** Since the fidelity of the model changes during the optimization process, adaptive evolution control was introduced to tune the life-length. Loshchilov [37] used the model error to adjust the life-length. If the error is 0, the life-length gets the maximum value. Otherwise, life-length decreases as the model error increases.

In this study, we follow a simple approach for evolution control. We evaluate each offspring with the model; if the offspring is successful based on the model evaluation, then we use the real objective function for evaluation of the offspring, if not we choose another point for model evaluation. This approach was implemented by Kramer. Kramer [32] implemented a surrogate-assisted $(1+1)$-ES, in which in every iteration an offspring is evaluated by the real fitness function, only if, it is better than tth best solution in the training set based on the prediction of the model. Algorithm 2 shows the Kramer's Meta-Model ES (MM-ES) where $\mathbf{x}_t$ is the tth last solution and $\hat{f}(\mathbf{x})$ is the model prediction of a location $\mathbf{x}$. Depending on the type of the test function and the value of step-size, the objective function can be used for every offspring (life-length of one), or it might be used after a considerable number of model evaluations. $\mathbf{x}_1$ is the most recent solution and has the highest objective function value in the training set. Therefore, setting $t = 1$ is the highest standard that the algorithm can have for determining if an offspring is promising. However, Kramer uses different values of t for different test functions. In Kramer's approach the value of t sometimes should be set to values from 10 to 50. That is because in some cases the model might overestimate the function value of the offspring and therefore, reject all solutions. In our research we fixed this problem using an approach known as normalization that

removes the bias in the training data. Moreover, Kramer used the 1/5th rule for the step-size adaptation, in future chapters we will investigate whether this mechanism, derived by Rechenberg [43], provides the optimal step-size for surrogate-assisted ES.

Algorithm 2 Kramer [32] MM-ES

initialize $\sigma \in \mathbb{R}^+, \mathbf{x} \in \mathbb{R}^n$
while not happy **do**
  adapt σ with Rechenberg
  $\mathbf{x}' \leftarrow \mathbf{x} + \sigma N(0, I)$
  if $\hat{f}(\mathbf{x}') \leq f(\mathbf{x}_t)$ **then**
    update $\hat{f} \leftarrow (\mathbf{x}', f(\mathbf{x}'))$
    if $f(\mathbf{x}') \leq f(\mathbf{x})$ **then**
      $\mathbf{x} \leftarrow \mathbf{x}'$
    end if
  end if
end while

2.3 Modeling Techniques

In this section, we discuss the four most popular modeling techniques that have been used in surrogate-assisted ES: Gaussian process regression, polynomial regression, Artificial neural networks, and finally support vector regression .

2.3.1 Gaussian Process Regression

Gaussian process (GP) is a powerful machine learning method with a limited number of hyper-parameters which can be tuned using the Maximum Likelihood Estimate (MLE) technique. This machine learning method was employed in many studies of surrogate-assisted evolutionary algorithms such as [16]. Another characteristic of GP is that it is based on probabilistic (Gaussian) prediction; it can provide confidence intervals for prediction of every point. This quality can be further used to tune the parameters of the model and to make a balance between using the model evaluation and the fitness evaluation (*evolution control*). This feature has also been used to control the exploitation vs. exploration of the optimization process. For instance, Emmerich et al. [16] used GP for modeling, and they added the mean squared error of the estimation to the model function to give higher priority to the points that have

higher uncertainty. In what follows we further discuss the GP as well as a strategy to tune the parameters of the GP.

In probability theory, *random processes* are used to describe functions. In a specific type of random process, known as Gaussian Process, each point $\mathbf{x}$ in the search space is assigned to a random variable, where the joint distribution of any finite subset of the random variables is Gaussian. Therefore, a Gaussian Process can be defined using a mean function $m(\mathbf{x})$ and a covariance function K.

$$\begin{pmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ ... \\ f(\mathbf{x}_n) \end{pmatrix} \sim N \left(\begin{pmatrix} m(\mathbf{x}_1) \\ m(\mathbf{x}_2) \\ ... \\ m(\mathbf{x}_n) \end{pmatrix}, \begin{pmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & ... & K(\mathbf{x}_1, \mathbf{x}_n) \\ K(\mathbf{x}_2, \mathbf{x}_1) & ... & K(\mathbf{x}_2, \mathbf{x}_n) \\ ... & & \\ K(\mathbf{x}_n, \mathbf{x}_1) & ... & K(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} \right) \tag{2.14}$$

Random processes are categorized into stationary and non-stationary random processes. The distribution of the random variables in a stationary process is invariant to translation in the search space. Therefore, the average of the distribution of a stationary Gaussian Process is a constant value ($m(\mathbf{x}) = m, \forall \mathbf{x} \in \mathbb{R}$), and the covariance function should only depend on the distance between the data points.

Covariance functions, also known as *kernels*, are the core functions in the Gaussian Process that define and measure the relationship of the input data in the search space and in correlation with other data points. This relationship in stationary random processes can be measured by a *stationary kernel*, which measures the distance between inputs invariant to translations in the search space, e.g., the Euclidean distance function. Quality of the kernels mostly depends on the type of function that is being approximated. Certain kernels can measure particular types of relationships, like *isotropic kernels* which are invariant to the rotation in the search space and therefore they are suitable for spherical test functions.

Bajer et al. [11] used three types of kernels: $k_{SquaredExponential}$, $k_{Matern_{v=3/2}}$, $k_{Matern_{v=5/2}}$, which are all stationary kernels (Table 2.1). One reason that we use these kernels is that the accuracy of the regression relies mostly on the tuning of one

main hyper-parameter — *length-scale*. Rasmussen and Williams [42] informally define Length-scale as "the distance you have to move in input space before the function value can change significantly". In this research, we use $k_{SquaredExponential}$ because it is isotropic and one of the most common kernels for GP.

Name	Function
Squared Exponential	$k_{SquaredExponential} = exp\left(-\frac{r^2}{2l^2}\right)$
Matern ($v = 3/2$)	$k_{Matern_{v=3/2}} = (1 + \frac{\sqrt{3}r}{l})exp(-\frac{\sqrt{3}r}{l})$
Matern ($v = 5/2$)	$k_{Matern_{v=5/2}} = (1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{l^2})exp(-\frac{\sqrt{5}r}{l})$

Table 2.1: Well-known stationary covariance functions that can be used as kernel functions in Equation 2.14.

In this research we use the GP to create a model based on the objective function. We want to approximate the objective function f using the training data, and a stationary Gaussian process. Gaussian process describes f as a probability density function

$$\begin{aligned} p(f(\mathbf{x})|f(\mathbf{x}_1), ..., f(\mathbf{x}_n)) &= \frac{p(f(\mathbf{x}), f(\mathbf{x}_1), ..., f(\mathbf{x}_n))}{p(f(\mathbf{x}_1), ..., f(\mathbf{x}_n))} \\ &= \frac{N(f(\mathbf{x}), f(\mathbf{x}_1), ..., f(\mathbf{x}_n))|m, \tilde{C})}{N(f(\mathbf{x}_1), ..., f(\mathbf{x}_n)|m, C)} \end{aligned} \tag{2.15}$$

where N is the probability density function of a normal distribution. Based on Gaussian process theorem any finite subset of the random variables in a Gaussian Process is normally distributed. Therefore, as Equation (2.15) suggests both $p(f(\mathbf{x}_1), ..., f(\mathbf{x}_n))$ and $p(f(\mathbf{x}), f(\mathbf{x}_1), ..., f(\mathbf{x}_n))$ can be defined as normal distributions. C is the covariance matrix of the training data.

$$C = \begin{pmatrix} K(0) & K(|\mathbf{x}_1 - \mathbf{x}_2|) & ... & K(|\mathbf{x}_1 - \mathbf{x}_n|) \\ K(|\mathbf{x}_2 - \mathbf{x}_1|) & K(0) & ... & K(|\mathbf{x}_2 - \mathbf{x}_n|) \\ ... & ... & ... & ... \\ K(|\mathbf{x}_n - \mathbf{x}_1|) & K(|\mathbf{x}_n - \mathbf{x}_2|) & ... & K(0) \end{pmatrix} \tag{2.16}$$

and $\tilde{C}$ is the covariance matrix of the training data and the parameter space location

$\mathbf{x}$

$$\tilde{C} = \begin{pmatrix} K(0) & K(|\mathbf{x}-\mathbf{x}_1|) & K(|\mathbf{x}-\mathbf{x}_2|) & ... & K(|\mathbf{x}-\mathbf{x}_n|) \\ K(|\mathbf{x}_1-\mathbf{x}|) & K(0) & K(|\mathbf{x}_1-\mathbf{x}_2|) & ... & K(|\mathbf{x}_1-\mathbf{x}_n|) \\ K(|\mathbf{x}_2-\mathbf{x}|) & K(|\mathbf{x}_2-\mathbf{x}_1|) & K(0) & ... & K(|\mathbf{x}_2-\mathbf{x}_n|) \\ ... & ... & ... & ... & ... \\ K(|\mathbf{x}_n-\mathbf{x}—) & K(|\mathbf{x}_n-\mathbf{x}_1|) & K(|\mathbf{x}_n-\mathbf{x}_2|) & ... & K(0) \end{pmatrix} \tag{2.17}$$

This Equation can be simplified as

$$\tilde{C} = \begin{pmatrix} K(0) & \mathbf{k}^T \\ \mathbf{k} & C \end{pmatrix} \tag{2.18}$$

where $\mathbf{k}$ is the covariance between the parameter space location $\mathbf{x}$ and the training data

$$\mathbf{k} = \begin{pmatrix} K(|\mathbf{x}-\mathbf{x}_1|) \\ K(|\mathbf{x}-\mathbf{x}_2|) \\ ... \\ K(|\mathbf{x}-\mathbf{x}_n|) \end{pmatrix} \tag{2.19}$$

Lemma 1 (Conditional Normal Distribution [15]) *If a set of normally distributed random variables X is divided to two normally distributed sets* $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$ *with means* $\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}$ *and covariance between the two sets defined by* $Cov(X) = \begin{pmatrix} \Sigma_{11} & \Sigma_{21} \\ \Sigma_{12} & \Sigma_{22} \end{pmatrix}$*, the probability distribution of the the conditional distribution of X_1 given $X_2 = x_2$ is equal to*

$$p(X_1|X_2 = x_2) = N(\mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}) \tag{2.20}$$

Equation 2.15 can be interpreted as a conditional distribution of a normally distributed vector containing the parameter space location $\mathbf{x} \in \mathbb{R}^n$ and the training data. For $\mu_1 = \mu_2 = m$, and the covariance matrix equal to $\tilde{C}$ and by using Lemma 1, it can be concluded that the probability density of the function value of the parameter space location $\mathbf{x}$ (Equation 2.15) using GP can be approximated as

$$N(f(\mathbf{x})|\mu, \sigma^2) \tag{2.21}$$

where μ and σ^2 are the mean and the variance of the distribution, and can be denoted as

$$\begin{aligned} \mu &= \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2) \\ &= m + \mathbf{k}^T C^{-1} \begin{bmatrix} f(\mathbf{x}_1) - m \\ f(\mathbf{x}_2) - m \\ ... \\ f(\mathbf{x}_n) - m \end{bmatrix} \\ \sigma^2 &= \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} \\ &= K(0) - \mathbf{k}^T C^{-1}\mathbf{k} \end{aligned} \tag{2.22}$$

A detailed proof of the lemma 1 is available in [15].

In the next step, we have to tune the hyper-parameters of the kernel. Maximum Likelihood Estimate (MLE) is a strategy that can be used for setting the GP's hyper-parameters. In this strategy, we use an optimization algorithm to find the best value for the length-scale l that maximizes the following function

$$\max_l N(f(\mathbf{x}_1), ..., f(\mathbf{x}_n)|m, C)) \tag{2.23}$$

In this paper, we use the Broyden-Fletcher-Goldfarb-Shanno algorithm, also known as BFGS [14], to optimize Equation 2.23. The main problem when using MLE is that, as Mackay [39] described it, the optimization landscape of length-scale is multimodal. Büche et al. [13] use a combination of different search methods to find the global optimum of the hyperparameters. First, they used CMA-ES with 3000 iterations. Then, they used BFGS ten times. After that, they used the CMA-ES again to avoid local optima of the search space of the length-scale. These searches add a considerable amount of computational cost to the optimization process. They also reported numerical difficulties when they used the CMA-ES for optimizing the likelihood function. We further discuss the maximum likelihood estimate and the hyper-parameters of the GP in the next chapters.

In a GP model, the fitness estimation of points that are too far from the training points by default converges into the average function value of the training points (known as normalization). However, the behavior of surrogate-assisted (1 + 1)-ES suggests that offspring are most likely to be selected in the vicinity of the parent. Moreover, in Figure 2.3 we can see that in some cases the average function value of training points might be much higher than the new offspring and therefore it might result in a model that rejects all offspring. Therefore, in this study, we use the function value of the parent for normalization. Figure 2.3 compares the two approaches for the normalization with models created from three and four training points on a one dimension quadratic sphere. The training points are generated by (1 + 1)-ES algorithm. At some arbitrary iterations, we stop the optimization and make two models from the two approaches of normalization of the Gaussian Process. Figure 2.3 compares the models with the real objective function. Based on the figure, the model that is normalized based on the value of parent predicts smaller and most likely better values for the offspring in the close vicinity of the parent.

2.3.2 Polynomial Regression

Polynomial regression is a modeling technique in which the relationship between the input variable $\mathbf{x} \in \mathbb{R}^n$ and the fitness function $f : \mathbb{R}^n \to \mathbb{R}$ is modelled to a polynomial model function $\hat{f} = \tilde{\mathbf{x}}^{\mathbf{T}}\, \boldsymbol{\beta}$, where $\boldsymbol{\beta}$ contains the coefficients of the model. The value of the $\tilde{\mathbf{x}}$ depends on the type of model. For a linear model $\tilde{\mathbf{x}}$ is $(x_1, x_2, ..., x_n, 1)^T$ and for a quadratic model $\tilde{\mathbf{x}} = (x_1^2, x_2^2, ..., x_n^2, x_1, x_2, ..., x_n, 1)^T$, where $\mathbf{x} = [x_1, x_2, ..., x_n]^T$.

The parameters of the model $\boldsymbol{\beta}$ can be selected by minimizing the least square error function A

$$A = \sum_{i=1}^{l} \left[(\hat{f}(\mathbf{x}_i, \boldsymbol{\beta}) - y_i)^2 \right] \tag{2.24}$$

where $\mathbf{x}_i$ and y_i are, respectively, space locations and function values of the training set ($\mathbf{x}_i \in \mathbb{R}^n,\ y_i \in \mathbb{R},\ i = 1, ..., l$) and l is the number of training points.

In [30], Kern et al. used a locally weighted regression technique (LWR) for modeling. This technique makes the regression more locally accurate by weighting the

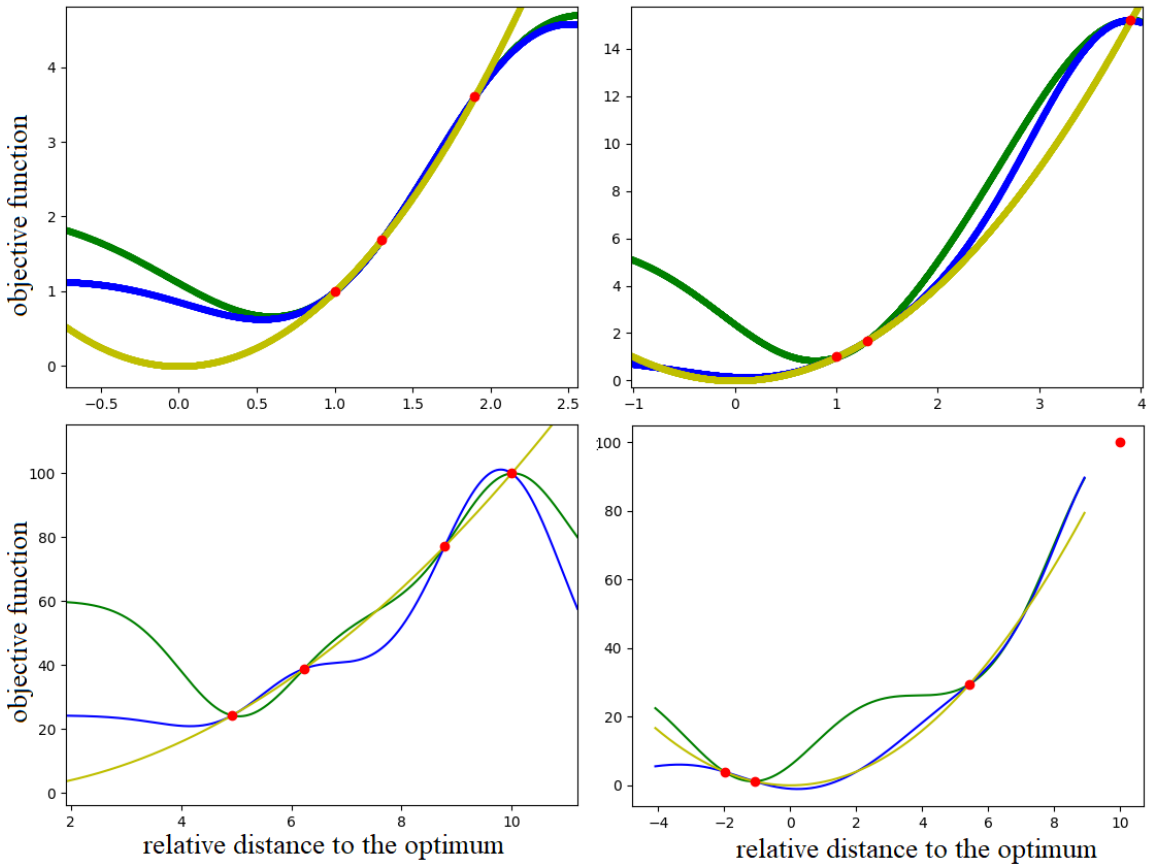

Figure 2.3: Yellow curve is the real objective function (quadratic sphere). The blue curve uses a GP with normalization based on the function value of the parent. The green curve is GP with normalization over the mean of training data.

training set based on their distance to the offspring.

$$A(\mathbf{q}) = \sum_{i=1}^{l} \left[(\hat{f}(\mathbf{x}_i, \boldsymbol{\beta}) - y_i)^2 K \left(\frac{d(\mathbf{x}_i, \mathbf{q})}{h} \right) \right] \tag{2.25}$$

where $\mathbf{q} \in \mathbb{R}^n$ is a query point, and K is a kernel function which has the same role as the kernel functions in GPR, and h (like length-scale for GPR) is the distance to move in input space before the function value changes significantly, and d finds the distance between two points.

2.3.3 Artificial Neural Network

Artificial neural networks (ANNs) inspired by the biological neural networks are probably the most popular class of machine learning algorithms [34]. They were popularized after Rumelhart, Hinton and Williams [45] introduced the error-backpropagation as a technique to adjust the parameters of a multilayer perceptron. Choosing the structure of the ANNs including the number of hidden layers, the number of neurons in each layer and the number of connections between neurons is one of the drawbacks of using ANNs for surrogate modeling. Ulmer et al. [49] use a three-layered ANN with a RBF (radial basis function) activation kernel to implement a surrogate-assisted evolution strategy, but they do not provide any evidence that the values they used for the number of layers and the number of nodes in each layer are optimized and whether we should use different structures for higher dimensions.

2.3.4 Support Vector Regression

Support Vector Regression (SVR) is a class of machine learning algorithms. They are designed to find the optimal hyperplane which is the model that estimates the values of the objective function for all training points with the margin of error of ϵ. A linear model with a weight vector $\mathbf{w}$ can be defined as

$$f(\mathbf{x}) = w_0 + \mathbf{w}^T\mathbf{x} \tag{2.26}$$

where w_0 is the offset and $\mathbf{x} \in \mathbb{R}^n$ is an input data location. The objective function can be approximated by minimizing the $\|\mathbf{w}\|_2^2$ while the training points are within the ϵ distance from the values of the model.

$$\begin{aligned} &minimize: \ \frac{1}{2}\|\mathbf{w}\|_2^2 \\ &subject\ to \begin{cases} y_i - \mathbf{w}^T\mathbf{x}_i - w_0 < \epsilon \\ -y_i + \mathbf{w}^T\mathbf{x}_i + w_0 < \epsilon \end{cases} \end{aligned} \tag{2.27}$$

where $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \mathbb{R}$ are the training data space location and objective function value. This type of optimization problem can be solved using quadratic programming. In this approach we assumed that there is a function that can approximate all training

points with ϵ accuracy. However, this might not be feasible. A more advanced version of SVR, known as soft SVR, uses two more factors $\xi_i^+ \in \mathbb{R}^+$ and $\xi_i^- \in \mathbb{R}^+$, that stores the amount of deviation that the model has from each training data. This capability makes the regression more flexible and capable of modeling complex optimization problems.

$$\begin{aligned} &minimize: \ \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{1}{l}C \ \sum_{i=1}^{l}(\xi_i^+ + \xi_i^-) \\ &subject\ to \begin{cases} y_i - \mathbf{w}^T\mathbf{x}_i - w_0 < \epsilon + \xi_i^+ \\ -y_i + \mathbf{w}^T\mathbf{x}_i + w_0 < \epsilon + \xi_i^- \end{cases} \end{aligned} \tag{2.28}$$

where C controls the effect of $\mathbf{w}$ vs. $\sum_{i=1}^{l}(\xi_i^+ + \xi_i^-)$. We can also use a kernel function to make the machine learning algorithm more suitable for modeling of non-linear data. The classification version of SVR, known as Support Vector Machine (SVM), has been implemented for surrogate modeling. Loshchilov [34] used a modification of SVM, known as Ranking SVM, as the surrogate modeling alongside CMA-ES. The Ranking SVM is trained to estimate the ranking of the points, which suits better with the ranking based selection procedure of the CMA-ES.

2.4 General Test Functions

As was mentioned earlier, in this study, we only consider well-conditioned test functions to evaluate the performance of the surrogate-assisted ES. However, in this section, we review other types of test functions with certain types of difficulties and the strategies that have been proposed to address these difficulties. These strategies can be used in future work to generalize the results of this study to other test functions and real-world problems.

2.4.1 Ill-Conditioned Functions

Optimization of ill-conditioned problems is difficult because the value of the step-size in different directions differs significantly. In some ill-conditioned problems, this difference is too big to be handled without the use of Covariance Matrix Adaptation.

Covariance Matrix Adaptation (CMA), initially introduced in [24], has the potential to transform some optimization problems to near spherical problems. CMA uses the cumulative successful mutations from past iterations to adapt the mutation covariance matrix. CMA-ES showed the best performance in the Black-Box Optimization Benchmarking (BBOB) 2009 [22], and many real-world problems [19] compared to other algorithms used in those papers. CMA based surrogate-assisted evolutionary algorithms were used in several papers including [37] and [34]. Loshchilov et al. [37] improved the performance of the support vector machine (SVM) by using the cumulative information recorded in the CMA from the past iterations a priori to create a better kernel for the modeling technique. Orekhov [41] follows the same approach to use the CMA to adapt the kernel of the Gaussian process. In this way, kernels are updated based on the type of problem that is being optimized.

2.4.2 Multi-Modal Functions

A location $\tilde{\mathbf{x}} \in \mathbb{R}^n$ is a *local minimizer* of function f if there is an $\epsilon > 0$ that for every point $\tilde{\mathbf{x}}$ that satisfies $||\mathbf{x} - \tilde{\mathbf{x}}|| < \epsilon$,

$$f(\tilde{\mathbf{x}}) \leq f(\mathbf{x}) \tag{2.29}$$

The function value of the local minimizer is called *local minimum*. A *global minimizer* of function f is a location $\tilde{\mathbf{x}} \in \mathbb{R}^n$ where for every point $\mathbf{x} \in \mathbb{R}^n$,

$$f(\tilde{\mathbf{x}}) \leq f(\mathbf{x}) \tag{2.30}$$

The function value of the global minimizer is called *global minimum*.

Multi-modal functions are functions that have more than one local optimum. The main difficulty when optimizing these functions is that the optimization algorithm might converge into a local optimum instead of the global optimum. In [4], to handle multi-modal functions a restart strategy for the CMA-ES, called IPOP-CMA-ES, was introduced. This algorithm contains multiple independent restarts of the optimization process, where the population size of offspring increases in each restart. Later another multi-start algorithm known as BIPOP-CMA-ES [20] used two strategies: one with growing population size where the population is doubled in each restart

and the other with a small but varying population size where population size varies in $[\lambda_{default}, \lambda_{default}/2]$. $\lambda_{default}$ is a relatively small population size that is adapted for uni-modal functions. Loschilov et al. used both strategies in [36] and [38] to reduce the chance of getting stuck in local optima in surrogate-assisted evolution strategies. Based on the evaluation of these algorithms on COCO benchmark [23], these algorithms were able to provide, the best result compared to other evolutionary algorithms on some of the tests cases used in the study [38].

2.4.3 Constrained Functions

Surrogate-assisted optimization algorithms were also implemented for solving constrained optimization. COBRA (Constrained Optimization By RAdial basis function interpolation), introduced by Regis [44], uses radial basis functions (RBF) to approximate both the objective and the constraint functions. In [7] an implementation of COBRA with R was able to outperform a non-model based optimization strategy on a subset of well-known constrained problems, known as G-problems [40]. Bagheri [6] further investigates the behaviour of COBRA with different settings and a parameter adaptation mechanism which lead to better performance of the algorithm in higher dimensions and especially on a benchmark of high dimensional constrained industrial problems known as MOPTA08 [28].

Chapter 3

Analysis

3.1 Overview

As was mentioned in previous chapters, in this study, we design a surrogate-assisted (1+1)-ES (SA-(1+1)-ES) based on the approach suggested by Kramer [32]. Kramer's approach used the last tth solution as a threshold to determine which offspring should be evaluated by the objective function and which offspring should be neglected. The choice of t seemed to be arbitrary. Kramer doesn't suggest any strategy for choosing the t value. In this research, we only consider $t = 1$, which means that if the offspring is better than its parent based on the model evaluation, it is promising enough to be selected for objective function evaluation. $t = 1$ is easier to analyze and compared to $t > 1$, requires a higher standard for the offspring. In each iteration of the SA-(1 + 1)-ES, an offspring is evaluated by the real objective function, only if, it is promising based on the predicted fitness value of the surrogate model. We also use a new approach for step-size adaptation. Kramer's step-size adaptation is based on the 1/5th rule. He uses the success rate of the objective function and the model to adapt the step-size. In other words, one out five generations should result in an offspring that is better than its parent based on both model and objective function evaluation. However, in this study, we argue that the step-size in a surrogate-assisted algorithm should be based on the accuracy of the model instead of the success rate of the algorithm and the success rate should be only used when we want to optimize the number of model evaluation. In this chapter, we use a noisy fitness function to simulate the behaviour of the surrogate model, and then we use the simulation to set the parameters of the step-size adaptation technique and to tune the hyperparameters of the modeling algorithm.

3.2 Simulation of SA-$(1+1)$-ES on a Quadratic Sphere

In this section, we simulate the performance of the SA-$(1+1)$-ES on a Quadratic Sphere. We follow the approach that was suggested by Beyer [12] to analyze the performance of $(1+1)$-ES on noisy fitness functions. We use this approach to simulate the modeling error in the form of Gaussian noise. Then, we investigate how this error can affect the quality of the SA-$(1+1)$-ES. The quality of the SA-$(1+1)$-ES is measured based on its fitness gain, true positive rate, and evaluation rate. Fitness gain was defined in chapter 2; in this chapter, we define the evaluation rate and true positive rate as measures to study the behaviour of surrogate-assisted evolution strategies.

For a quadratic sphere, we can denote the predicted objective function $\tilde{f}$ as

$$\tilde{f}(\mathbf{x}+\sigma\mathbf{z}) = f(\mathbf{x}+\sigma\mathbf{z}) + \xi \tag{3.1}$$

where ξ is the error and f is the objective function. We assume that our approximation of the objective function contains a Gaussian error. This Gaussian error is a normally distributed scalar with a standard deviation of σ_ϵ (known as *noise strength*). More accurate surrogate models have smaller values of σ_ϵ. We also assume that our modeling techniques are capable of removing the systematic error in the approximation and therefore the mean of the Gaussian error in the simulations are set to zero. Therefore, the error can be described by the following probability density function

$$p_\xi(x) = \frac{1}{\sqrt{2\pi}\sigma_\epsilon}\exp(-\frac{1}{2}(\frac{x}{\sigma_\epsilon})^2) \tag{3.2}$$

We normalize the noise strength in the same way that we did for fitness gain in the previous chapter

$$\sigma_\epsilon^* = \sigma_\epsilon\frac{n}{2R^2} \tag{3.3}$$

Where R is the distance between the parent and the optimizer.

We argue that the SA-$(1+1)$-ES can be simulated in the form of Algorithm 3. In Algorithm 3, first, a new offspring $\mathbf{x}_1 \in \mathbb{R}^n$ is evaluated using a noisy function that represents the model. If successful, it will be evaluated by the real objective function.

Otherwise, another offspring will be selected for evaluation. Throughout this process, step-size is adapted using a mechanism that keeps the mutation strength value σ constant with respect to distance to the optimum. This is not a real optimization algorithm but rather a simulation of the performance of SA-$(1+1)$-ES on a quadratic sphere.

Algorithm 3 Simulated SA-$(1+1)$-ES on a quadratic sphere

1: initialize $\sigma_\epsilon^* \in \mathbb{R}, \mathbf{x} \in \mathbb{R}^n$
2: **while** not happy **do**
3: $\quad \sigma \leftarrow \frac{\sigma^* \sqrt{\sum_{i=1}^{n} (x_i - x_{o_i})^2}}{n}$
4: $\quad \sigma_\epsilon \leftarrow \frac{\sigma_\epsilon^* \sum_{i=1}^{n} (x_i - x_{o_i})^2}{n}$
5: $\quad$ **Do**
6: $\quad\quad \mathbf{x}_1 \leftarrow \mathbf{x} + \sigma N(0, I)$
7: $\quad$ **while** $f(\mathbf{x}_1) + \sigma_\epsilon N(0,1) > f(\mathbf{x})$
8: $\quad$ evaluate $f(\mathbf{x}) - f(\mathbf{x}_1)$ to measure the fitness gain q
9: **end while**

In Algorithm 3, the fitness value of the parent is always accurate as it is measured by the real objective function, but the offspring is evaluated by a noisy objective function (surrogate model). Using the notation of noise strength, the model error can be denoted as $\xi = \sigma_\epsilon z_\epsilon$ where z_ϵ is a standard normally distributed random variable. Therefore, following the approach used in [1], the noisy fitness gain of the algorithm when $n \to \infty$ can be calculated as

$$\begin{aligned} q_\epsilon^* &= \sigma^* z_1 - \frac{\sigma^{*2}}{2} + \sigma_\epsilon^* z_\epsilon \\ &= q^* + \sigma_\epsilon^* z_\epsilon \end{aligned} \tag{3.4}$$

Therefore, the *success probability of the model evaluation* p_{eval} is equal to

$$\begin{aligned} p_{eval} &= Prob\left[q_{\epsilon^*} > 0\right] \\ &= Prob\left[\sigma^* z_1 - \frac{\sigma^{*2}}{2} + \sigma_\epsilon^* z_\epsilon > 0\right] \\ &= Prob\left[z_2\sqrt{\sigma^{*2} + \sigma_\epsilon^{*2}} - \frac{\sigma^{*2}}{2} > 0\right] \\ &= \Phi\left(\frac{-\sigma^{*2}/2}{\sqrt{\sigma^{*2} + \sigma_\epsilon^{*2}}}\right) \end{aligned} \tag{3.5}$$

where $z_2\sqrt{\sigma^{*2} + \sigma_\epsilon^{*2}}$ is the sum of the two normally distributed random variables $\sigma^* z_1$ and $\sigma_\epsilon^* z_\epsilon$. $1/p_{eval}$ represents the average number of model evaluations in each iteration of the outer loop of Algorithm 3.

The offspring that are superior to their parents, based on the noisy fitness function, are evaluated by the real objective function. The probability of an offspring being better than its parent p_{step}, based on both the model evaluation and the objective function is

$$\begin{aligned} p_{step} &= Prob[q_{\epsilon^*} > 0 \wedge q^* > 0] \\ &= \int_{\frac{\sigma^*}{2}}^{\infty} p_{q^*}(z) p_{q^*}^{(acc)}(z) dz \\ &= \int_{\frac{\sigma^*}{2}}^{\infty} p_{q^*}(z) \Phi\left(\frac{z}{\sigma_\epsilon^*}\right) dz \\ &= \int_{\frac{\sigma^*}{2}}^{\infty} \frac{1}{\sqrt{2\pi}\sigma^*} \exp\left(-\frac{1}{2}\left(\frac{z + \sigma^{*2}/2}{\sigma^*}\right)^2\right) \Phi\left(\frac{z}{\sigma_\epsilon^*}\right) dz \\ &= \frac{1}{\sqrt{2\pi}} \int_{\frac{\sigma^*}{2}}^{\infty} e^{-\frac{1}{2}y^2} \Phi\left(\frac{2\sigma^* y - \sigma^{*2}}{2\sigma_\epsilon^*}\right) dy \end{aligned} \tag{3.6}$$

where substitution $z = (y\sigma^* - \sigma^{*2}/2)$ has been used. $p_{q^*}^{(acc)}$ is the probability of an individual offspring being selected by the inner loop in Algorithm 3 (model approximation). $p_{q^*}^{(acc)}$ depends both on the accuracy of the model and the value of fitness

gain. Therefore,

$$\begin{aligned} p_{q^*}^{(acc)}(y) &= Prob[r^2 - \xi < R^2] \\ &= Prob[-y < \xi^*] \\ &= \Phi\left(\frac{y}{\sigma_\epsilon^*}\right) \end{aligned} \tag{3.7}$$

Now, based on these two probabilities p_{step} and p_{eval}, we can evaluate the quality of the approximation of the value of objective function. We examine the noisy objective function based on the number offspring that are correctly selected by the simulation. The *true positive rate* $p_{truepositive}$ of a surrogate-assisted ES can be defined as the probability of an offspring being superior to its parent based on the true fitness function, presuming that it is already better than the parent based on the model evaluation. Therefore, using the aforementioned probabilities,

$$\begin{aligned} p_{truepositive} &= Prob[q^* > 0 | q_{\epsilon^*} > 0] \\ &= \frac{p_{step}}{p_{eval}} \\ &= \frac{\int_{\frac{\sigma^*}{2}}^{\infty} e^{-\frac{1}{2}y^2} \Phi\left(\frac{2\sigma^* y - \sigma^{*2}}{2\sigma_\epsilon^*}\right) dy}{\sqrt{2\pi}\Phi\left(\frac{-\sigma^{*2}}{2\sqrt{\sigma^{*2}+\sigma_\epsilon^{*2}}}\right)} \end{aligned} \tag{3.8}$$

Since the cost of model evaluation was assumed to be cheap, true positive rate can be used for step-size adaption of surrogate-assisted ESs. We can also measure the quality of a SA-ES using the fitness gain alongside the success probability. The fitness gain of SA-$(1+1)$-ES is the average fitness improvement in each iteration of the original fitness evaluation, where the offspring are superior to their parent based

on the model evaluation (fitness gain cannot have a negative value).

$$\begin{aligned}\eta &= \frac{\int_{\frac{\sigma^*}{2}}^{\infty} z p_{q^*}(z) p_{q^*}^{(acc)}(z) dz}{p_{eval}} \\ &= \frac{\int_{\frac{\sigma^*}{2}}^{\infty} \left(\sigma^* y - {\sigma^*}^2/2\right) e^{-\frac{1}{2}y^2} \Phi\left(\frac{2\sigma^* y - {\sigma^*}^2}{2\sigma_\epsilon^*}\right) dy}{\sqrt{2\pi}\Phi\left(\frac{-{\sigma^*}^2}{2\sqrt{{\sigma^*}^2+{\sigma_\epsilon^*}^2}}\right)}\end{aligned} \tag{3.9}$$

where substitution $z = (y\sigma^* - {\sigma^*}^2/2)$ has been used.

In this research, we assume that on a quadratic sphere the noise strength decreases as the optimal solution is approached. That is because the model cannot accurately compare two points if their fitness difference is smaller than the value of the noise. In the next sections, we provide experimental results to support this assumption. Based on this assumption if the distribution of normalized step-size and normalized nose remain independent of the iteration number, we can expect the algorithm to converge linearly, with the rate of fitness gain when $n \to \infty$.

3.3 Simulation of SA-$(1+1)$-ES with Respect to Noise-to-Signal Ratio

In the next step, we compare the theoretical quality measurements with experimental results of using Algorithm 3 on quadratic spheres with finite dimensions. We perform multiple optimization tasks with different values of normalized noise-strength and normalized step-size, and then we compare the real fitness gain of Algorithm 3 with the fitness gain resulted in Equation 3.9. The experimental results come from performing 10000 iterations of the algorithm 3, in each iteration starting from the same arbitrary point, on quadratic spheres with 10 and 100 dimensions. From Figure 3.1 it can be concluded that for small values of σ^* results are very close, but as σ^* increases, the theoretical measurements show much better fitness gain. Figure 3.1 also shows that when $\sigma_\epsilon^* = 0$, fitness gain increases with respect to σ^*. However, when $\sigma_\epsilon^* > 0$, the fitness gain first increases and then decreases. Choosing the right value

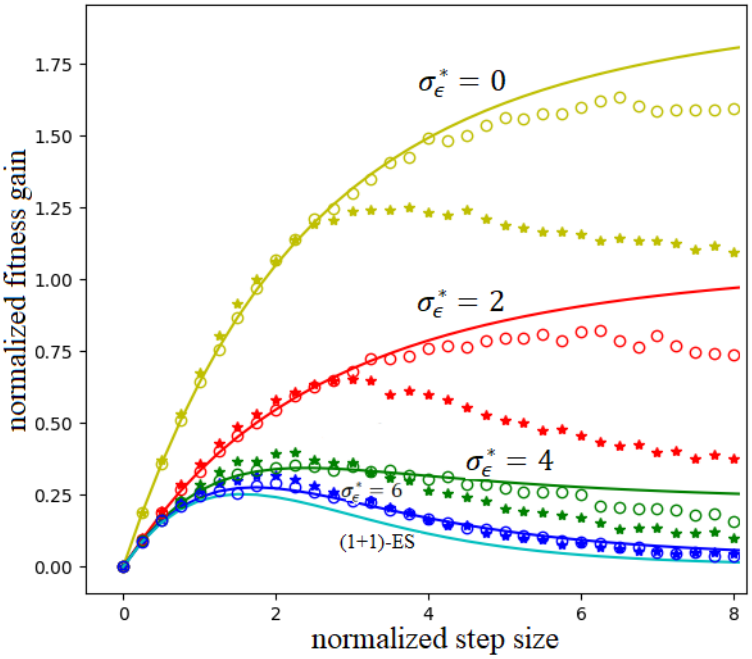

Figure 3.1: Expected fitness gain of simulated surrogate-assisted $(1+1)$-ES based on theoretical analysis $n \to \infty$ (solid line) and experimental results for $n = 10$ (stars) and $n = 100$ (circles). The algorithm was tested for five values of normalized noise strength: $\sigma_\epsilon^* = 0$ (yellow), $\sigma_\epsilon^* = 2$ (red), $\sigma_\epsilon^* = 4$ (green), $\sigma_\epsilon^* = 6$ (dark blue).

of σ^* in each iteration should be addressed by a proper step-size adaptation mechanism. Figure 3.2 compares the values of evaluation rate and true positive rate of the experimental results of the finite dimension quadratic sphere to theoretical results of using Equations 3.8 and 3.5 for an infinite dimension quadratic sphere. The results are very close for small values of noise and step-size. However for bigger values of step-size and noise the results are starting to diverge from each other.

It can be argued that for real modeling algorithms, increasing the step-size itself can increase the error of the model. Therefore, we look at the behaviour of the algorithm with respect to the relationship between the noise strength and the step-size. This relationship is measured using a parameter called *noise-to-signal ratio* or κ.

$$\kappa = \frac{\sigma_\epsilon^*}{\sigma^*} \tag{3.10}$$

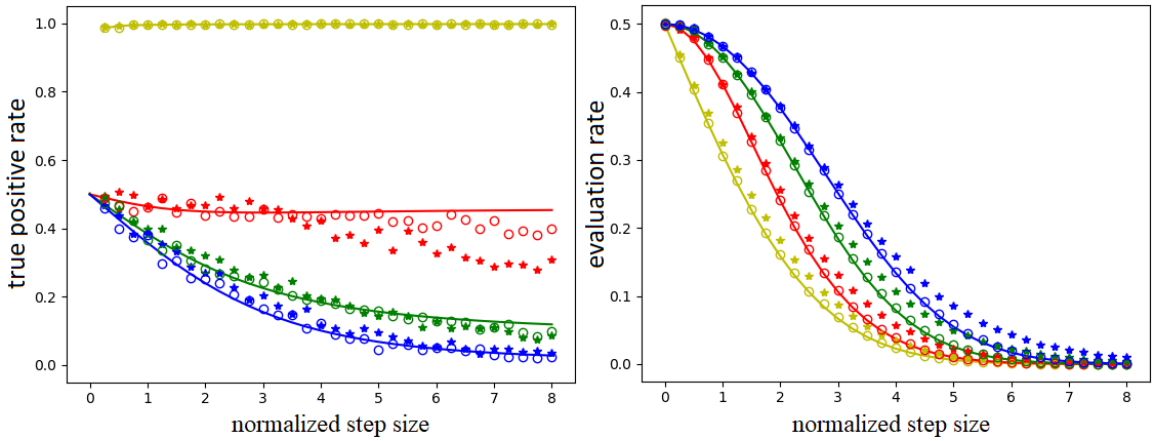

Figure 3.2: True positive rate and evaluation rate of surrogate-assisted $(1 + 1)$-ES based on theoretical analysis $n \to \infty$ (solid line) and experimental results for $n = 10$ (stars) and $n = 100$ (circles). The algorithm was tested for five values of normalized noise strength: $\sigma_\epsilon^* = 0$ (yellow), $\sigma_\epsilon^* = 2$ (red), $\sigma_\epsilon^* = 4$ (green), $\sigma_\epsilon^* = 6$ (dark blue).

To further investigate the performance of the algorithm, we look at a case where step-size is optimal. We use Equation 3.9 to find the optimum step-size for different values of noise-to-signal ratio: for different values of κ, by changing the value of the normalized step-size between 0 to 15, we find the step-size that maximizes the fitness gain. Figure 3.3 displays the fitness gain and normalized step-size as a function of κ, where step-size is optimum. As κ increases the normalized step-size reaches the value of 1.224 which is the optimum step-size of $(1 + 1)$-ES found by Rechenberg [43]. In addition to step-size, the values of other measures, such as normalized fitness gain and true positive rate, respectively get closer to the values of normalized fitness gain and success rate derived by Rechenberg for $(1 + 1)$-ES (Figures 3.3 and 3.4). These plots 3.3 and 3.4 also compare the values of theoretical analysis of the fitness gain, evaluation rate, step-size and true positive rate of the Equations 3.5 and 3.9 to experimental results.

3.4 Step-size Adaptation of the Simulated SA-$(1 + 1)$-ES

In this section, we study the effects of the accuracy of the model on the performance of the simulated SA-$(1 + 1)$-ES. We use the true positive rate as a measure for the accuracy of the model. This study follows the Rechenberg's approach to adapt the step-size. As was explained in the previous chapter, Rechenberg [43] found out that

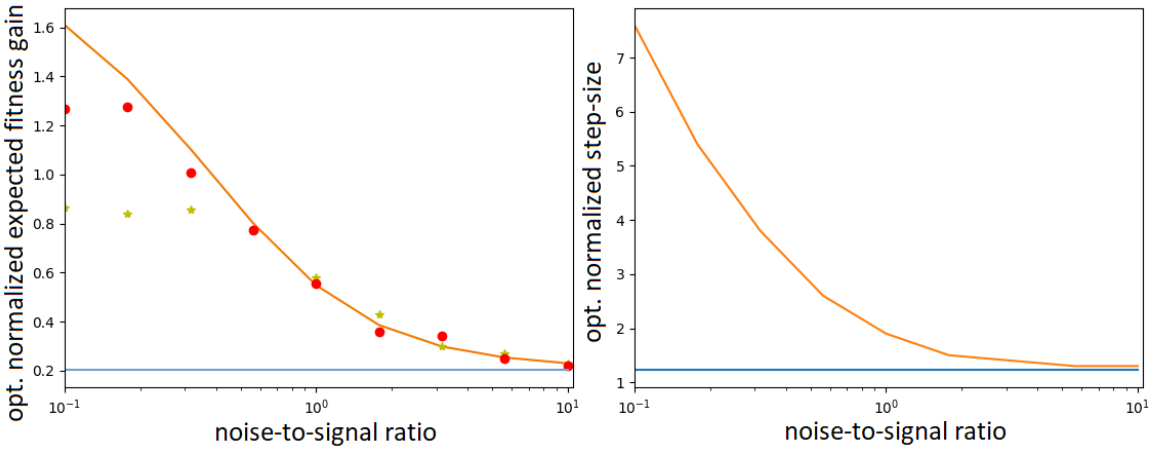

Figure 3.3: Optimal fitness gain and optimal normalized step-size as a function of noise-to-signal ratio when the step-size has the optimal value on quadratic spheres. Experimental results are for quadratic spheres with 10 (yellow stars), and 100 (red circles) dimensions. The theoretical result are for a quadratic sphere, where $n \to \infty$ (orange). The solid blue line represents the $(1+1)$-ES without surrogate modeling.

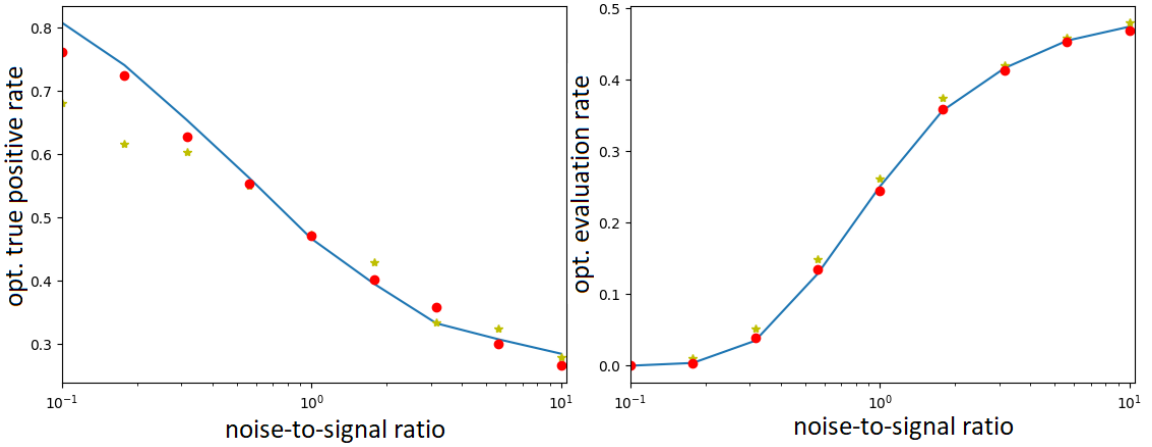

Figure 3.4: Optimal evaluation rate and optimal true positive rate as a function of noise-to-signal ratio when the step-size has the optimal value. Experimental results are for quadratic spheres with 10 (yellow stars), and 100 (red circles) dimensions. The theoretical result are for a quadratic sphere, where $n \to \infty$ (blue).

for $(1+1)$-ES, on certain test functions, the optimal value of the success probability is about 20% (the 1/5th rule). In this study, we are looking for a similar approach to adapt the step-size based on the true positive rate, where the step-size increases if the model can correctly select a superior offspring. Otherwise, the step-size should decrease till the model becomes accurate enough to predict the objective function.

For different values of noise-to-signal ratio, we use Equation 3.8 to find the step-sizes that result in arbitrary values of true positive rate. Then, we use those step-sizes to calculate other quality measures. Figure 3.5 illustrates the values of fitness gain and step-size as a function of noise-to-signal ratio. The solid blue lines are the values of the measures when the fitness gain is at its highest. Figure 3.5 shows for the values of noise-to-signal ratio higher than one, the optimum fitness gain is close to the fitness gain of the case where true positive rate is 30% and for the values of noise-to-signal ratio higher than two the values of the optimum fitness gain and the fitness gain of 30% true positive rate, almost match. In the next sections, we will observe that for a quadratic sphere the values of the noise-to-signal ratio are typically more than one or very close to one and therefore using the objective true positive rate of 30% leads to near-optimal performance. This conclusion can be made from Figure 3.6, which contains the relative fitness gain (normalized fitness gain/optimum fitness gain) as a function of κ. In the next chapter, we design and examine a step-size adaptation based on the true positive rate. Using this value also means that theoretically for any value of κ the algorithm arguably does not perform worse than $(1+1)$-ES. That is because when the model is inaccurate the algorithm works very similar to the simple $(1+1)$-ES and the true positive rate has the same effect as the success probability. Therefore, 30% true positive rate is like using 30% for the success rate.

3.5 Evaluation of the Model Error

In this section, we implement a Gaussian process regression (GPR). Then, we compare the behaviour of the surrogate model created by GPR with some of the assumptions that we made earlier in this chapter.

We start by testing Equation 3.3, where it was assumed that the noise strength is proportionate to the value of the objective function and therefore they were normalized in the same way. We use algorithm 4, which unlike Algorithm 3, uses a real

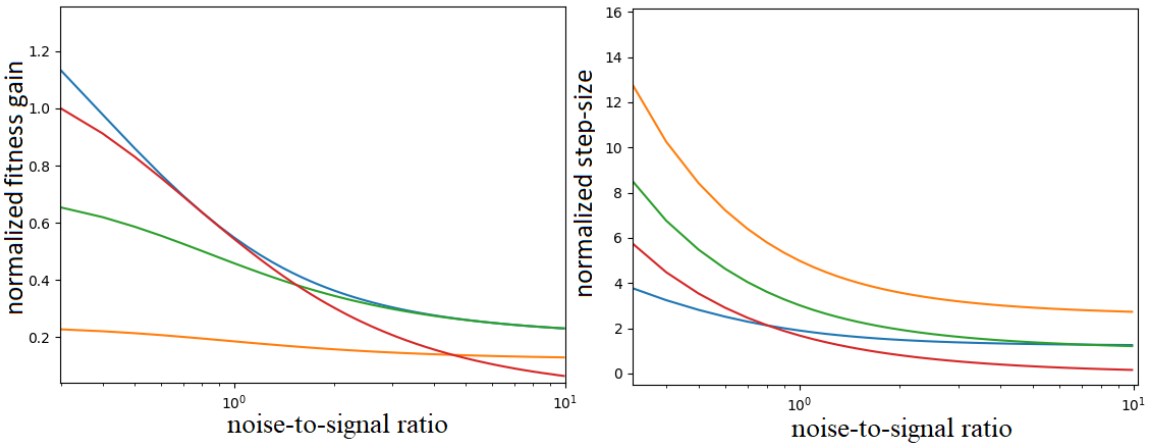

Figure 3.5: Normalized expected fitness gain and normalized step-size as a function of noise-to-signal ratio when $n \to \infty$ for different values of true positive rate. The solid blue line displays the case where the step-size has the optimum value. Other lines are the cases where the true positive rates of the algorithm are 50% (red), 30% (green) and 10% (orange).

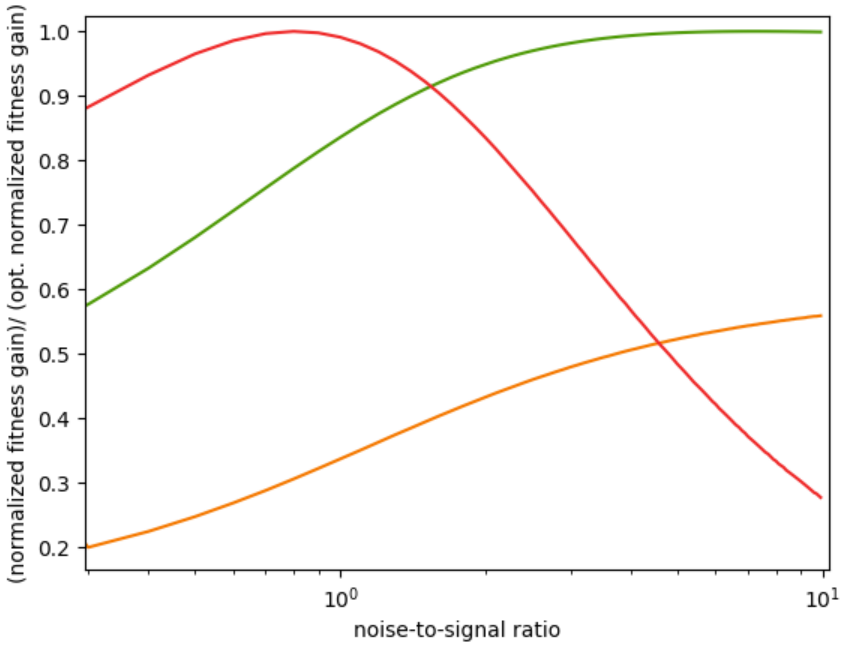

Figure 3.6: Normalized expected fitness gain divided by the optimum fitness gain, for different values of true positive rate. Lines display cases where the true positive rate of the algorithm has the values of 50% (red), 30% (green) and 10% (orange).

Algorithm 4 Surrogate-assisted $(1 + 1)$-ES with simulated step-size adaptation for quadratic sphere

initialize $\mathbf{x} \in \mathbb{R}^n, P \leftarrow \emptyset, \sigma^*, h$
while not happy **do**
  $\sigma \leftarrow \frac{\sigma^* \sqrt{\sum_{i=1}^{n} (x_i - x_{o_i})^2}}{n}$
  for $k \leftarrow 1$ to h **do**
    $\mathbf{x}_1 \leftarrow \mathbf{x} + \sigma N(0, I)$
    Evaluate $\mathbf{x}_1$ using the model, yielding $f_\epsilon(\mathbf{x}_1)$
  end for
  $P \leftarrow P \cup [\mathbf{x}_1, f(\mathbf{x}_1)]$
  Update the Model M
  if $f(\mathbf{x}_1) < f(\mathbf{x})$ **then**
    $\mathbf{x} \leftarrow \mathbf{x}_1$
  end if
end while

modeling algorithm. However, the step-size is chosen manually. The algorithm starts at a random point with zero mean and unit covariance matrix. The test function is a quadratic sphere with 10 dimensions. In each iteration of the outer loop of the Algorithm, we use the points that were evaluated by the objective function in the past 40 iterations to create a model of the objective function. The model is built by a Gaussian process regression algorithm with the length-scale of $3.5\sigma\sqrt{n}$. As was stated in the previous chapter the model is normalized based on the minimum value in the training set (parent). The values of the hyperparameters will be discussed in the next sections. After making the model, the algorithm generates 1000 ($h = 1000$) offspring with the step-size of σ^*. These 1000 offspring does not affect the progress of the algorithm. They are only generated to evaluate the accuracy of the model. We measure the variance of the difference between the objective function and the model prediction. This value is the noise strength of the model. We calculate the normalized values of the error using Equation 3.3. Figure 3.7 shows the noise strength and the normalized noise strength for the first 200 iterations of the algorithm for $\sigma^* = 0.8$, 1.2 and 1.6. The value of the noise strength decrease as the algorithm gets closer to the optimum. The plot of normalized noise strength shows that error values appear to be from a stable distribution and the error in the modeling seems to be proportionate to the value of the objective function.

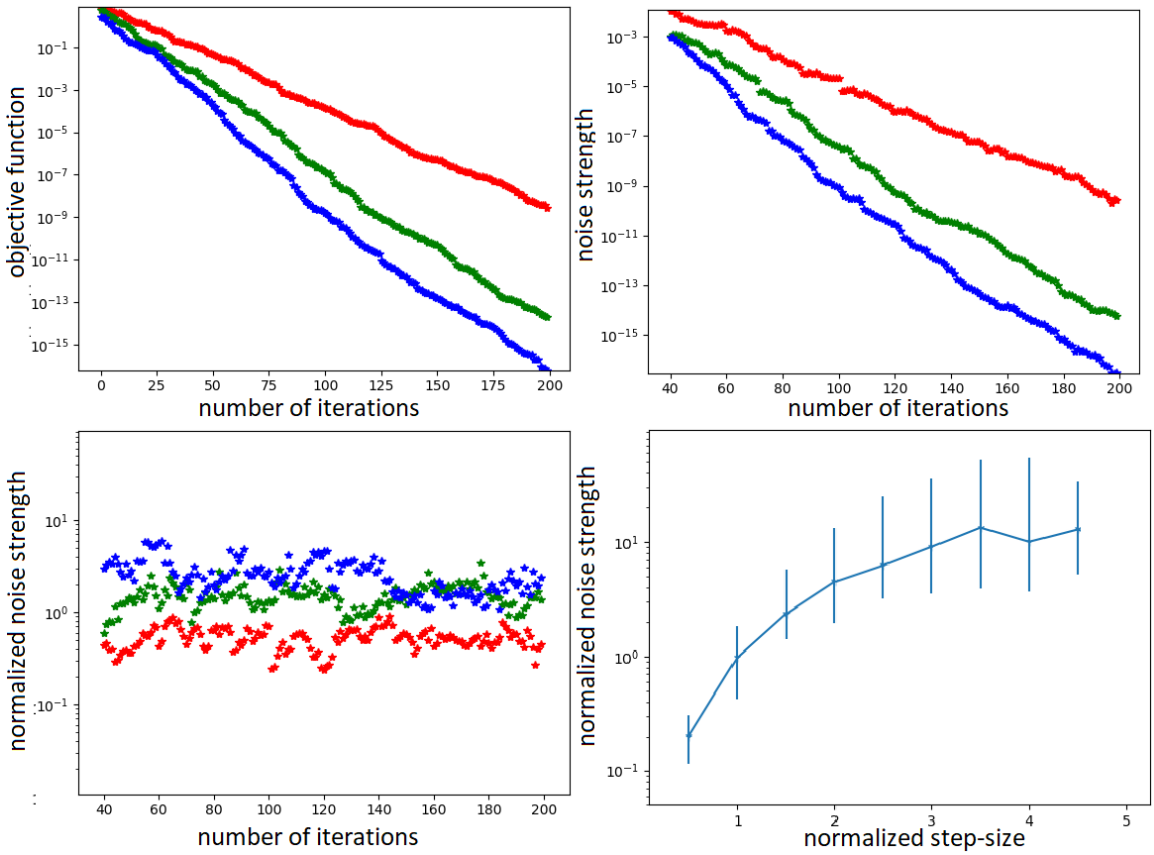

Figure 3.7: σ^*_ϵ Normalized noise strength, noise strength and objective function. The values of normalized step-size in the experiments are $\sigma^* = 0.8$ (red), $\sigma^* = 1.2$ (green), $\sigma^* = 1.6$ (blue).

Another parameter that needs to be investigated is the relationship between step-size and noise strength also known as κ. We measure the value of the noise strength of the model for normalized step-size between 0.5 and 4.5. Figure 3.7 shows the values of normalized noise strength and the distribution of the normalized error as a function of step-size after 10 trials of the algorithm. According to Figure 3.7, noise strength increases with respect to the value of step-size. It also shows that for the values of normalized step-size higher than one and in the range of the experiment the value of normalized noise strength is higher than normalized step-size. In other words, κ is bigger than one. This is important because we assumed in the previous sections that the value of κ is most likely either higher than one or very close to one and then based on that we suggested that 30% true positive rate gives us the near optimal performance on most cases.

Next, we look at the probability density of the model error, and we compare that with a Gaussian function with the same variance. In this comparison, we use the histogram and the kernel density estimation (KDE) of the error. KDE and histogram are two standard approaches to estimate the probability density of data. We create the histogram and KDE for both model error and a normal function with the same mean and standard deviation. Then, we visually compare them. Figure 3.8 compares the error of the model with a normal random variable. Results arguably support the idea of simulating the surrogate model using the objective function with additive Gaussian noise.

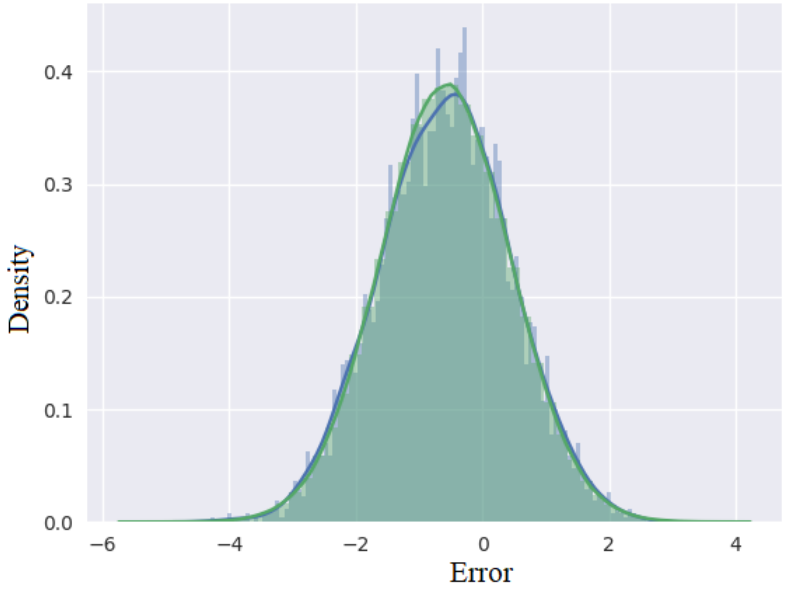

Figure 3.8: Modeling error. The blue histogram and curve represent the results of histogram and KDE of error, green histogram and curve are for a normal function with the same variance and mean.

3.6 Adaptation of Hyperparameters

3.6.1 Length-Scale

In addition to the step-size, hyperparameters of the modeling algorithm also affect the noise strength of the model. Figure 3.9 illustrates the effects of the length-scale on the noise strength and the bias of the error. This data was gathered from changing the length-scale of the Gaussian Process from 10 to 100 on the algorithm 4 with the

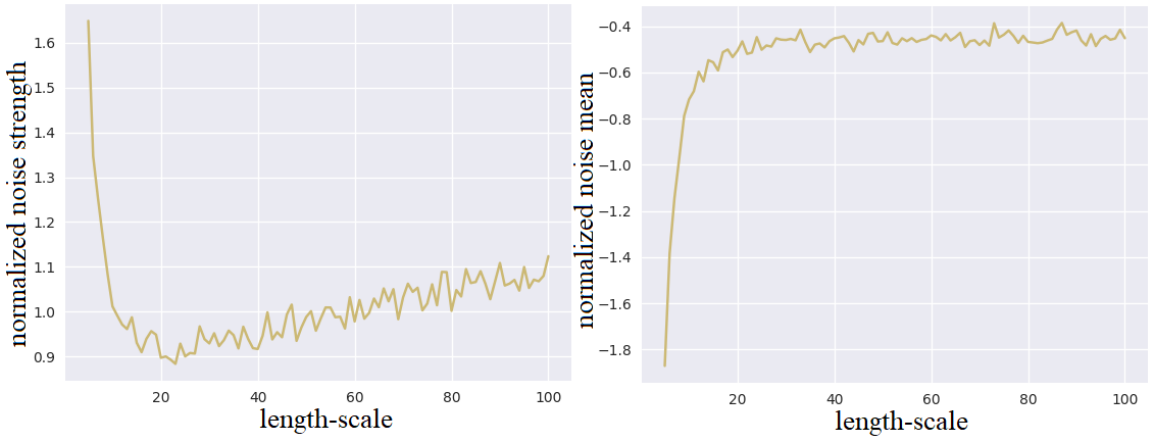

Figure 3.9: Modeling error variance and mean, as a function of length-scale.

same specifications that were mentioned in this section. Length-scale, as the main hyper-parameter of the GP with a squared exponential kernel, determines the speed of fluctuations in the model function. The most common approach to adapt the length-scale is Maximum Likelihood Estimate (MLE). However, as was mentioned in the earlier chapters, the search space of the Likelihood function, Equation 2.23, is multi-modal and necessitates a heavy computational cost. In this chapter, we use some features of the optimization problem and evolution strategy such as the step-size and number of dimensions of the optimization problem to improve the process of tuning the length-scale.

First, we use Algorithm 4 to find the best values of the length-scale. In this approach in each iteration of the outer loop of the algorithm and by changing the value of the length-scale, we find the value of the length-scale that minimizes the error of the surrogate model. The length-scale is measured on 850 points with the value of 1.1^{α}, where α changes from -350 to 500. The modeling algorithm is a GPR with the previous specifications. This is not a real length-scale adaptation algorithm, but an approach to study the relationship between length-scale and step-size, while avoiding the local optima for the length-scale. The optimization problem is a quadratic sphere with ten dimensions. To create a bigger perspective of the performance of the algorithm, we set the algorithm to start from a random point with zero mean and the variance of 10^{15}. Figure 3.10 compares the values of the optimum length-scale and step-size. From this figure, it can be concluded that there is a direct relation between the values

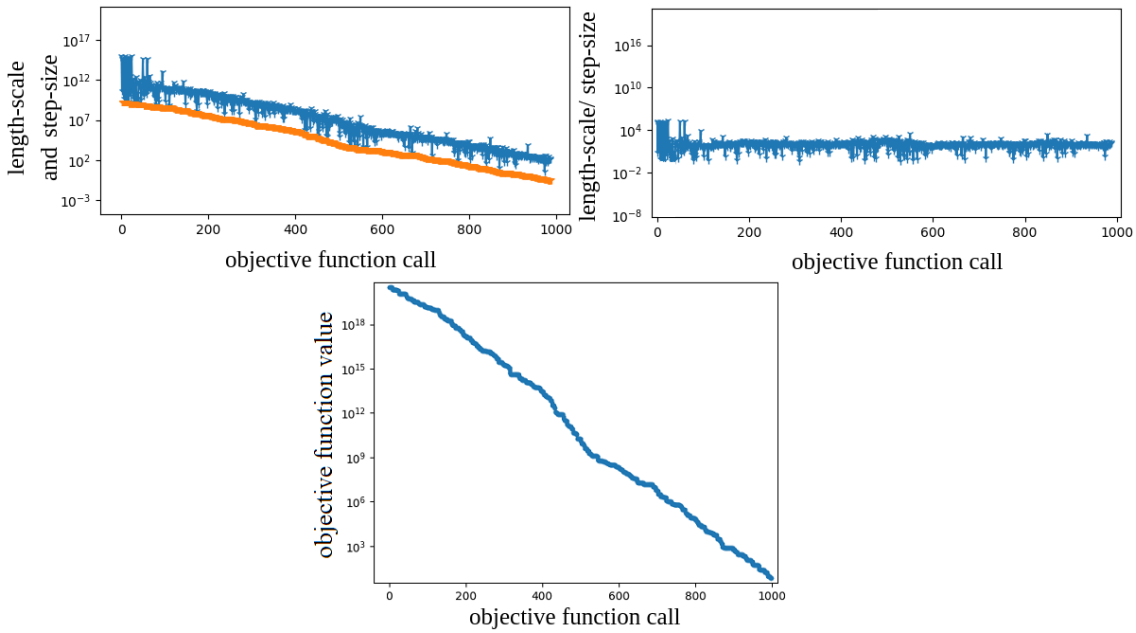

Figure 3.10: Length-scale (blue), step-size (orange), objective function value (blue) and relative length-scale (blue) as a function of objective function call, when the value of normalized step-size is 1.2.

of step-size and length-scales.

In the next step, we use the MLE to find the values of the length-scale. As was mentioned in chapter 2, we use a quasi-Newton method known as BFGS for the optimization of the length-scale. The number of restarts of the algorithm, starting point and range of the search are the parameters of the BFGS. In Figure 3.10 it can be seen that the value of the length-scale is fluctuating between $10^{-1}\sigma$ and $10^{6}\sigma$. We can use this information for choosing the range and starting point of the BFGS.

We use algorithm 4 and a benchmark containing quadratic spheres with 4, 8, 10, 16 and 32 dimensions. We change the inner loop of the algorithm to generate and evaluate new offspring until it finds an offspring which is more promising than its parent based on the model evaluation. Then we evaluate the point using the real objective function. In this way, we can find the true positive rate of the algorithm. In this experiment, we use the normalized step-size of 1.2, which provides the true positive rate of approximately 30% on a ten-dimensional quadratic sphere. Based on Figure 2.2 on a quadratic sphere the optimal value for success rate is approximately 30%. The number of the training points are four times number of the dimensions.

Other features of the algorithm are the same as the previous experiment.

Figure 3.11 shows that the modeling algorithm has the true positive rate of approximately 80% for four dimensions and smaller value of true positive rate for bigger number of dimensions. Figure 3.11 shows that if we select random offspring using a weak model, the true positive rate of the algorithm is about 30%, which is equal to the success rate that we picked for the algorithm. This means that as long as the true positive rate of the modeling algorithm is above 30%, model improves the performance of the algorithm. We experimented with BFGS with number of restarts from one to 500. Based on our experiments for the type of test function that we are using and 40 training data, the number of restarts does not affect the true positive rate of the algorithm.

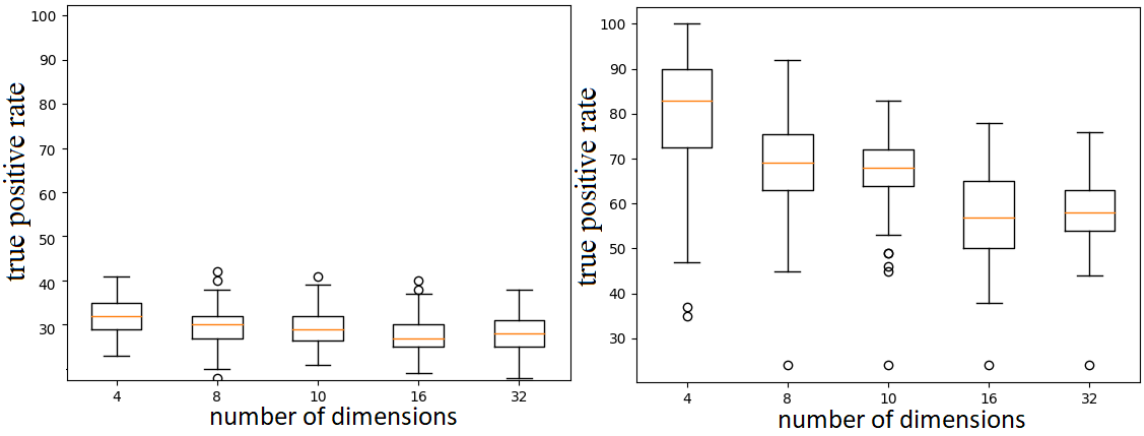

Figure 3.11: True positive rate as a function of number of dimensions, when the model selects the offspring randomly (figure on the left) which provides a true positive rate equal to the success probability of the algorithm, and the true positive rate when we use a relatively well-trained surrogate model (figure on the right).

In the next step, we use the values of length-scale provided by the maximum likelihood estimate to find the relationship between step-size, number of dimensions and the length-scale. Figure 3.12 compares the values of length-scale divided by step-size, with the value of a function $(3.5\sigma\sqrt{n})$. The error bars show the range of values that the length-scale takes when we use MLE for adapting the length-scale. The figure shows that choosing $3.5\sigma\sqrt{n}$ for Gaussian process provides a near optimal length-scale (based on MLE) for the GP. Therefore, in our experiment on SA-$(1+1)$-ES we use this value $(3.5\sigma\sqrt{n}\ )$ for the length-scale.

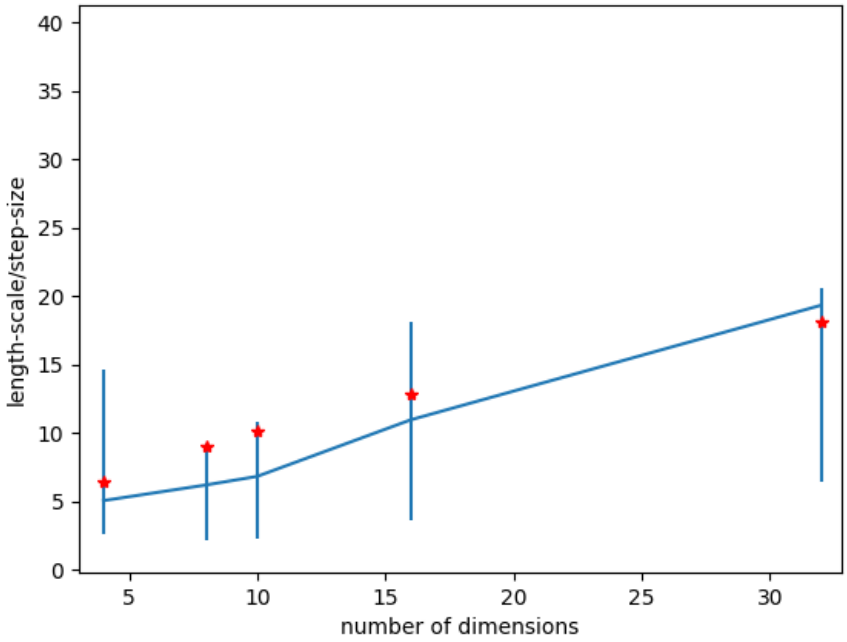

Figure 3.12: Length scale as a function of number of dimensions using the maximum likelihood estimate with 150 restarts (blue line) and $3.5\sigma\sqrt{n}$ (red stars).

3.6.2 Training Set

In a surrogate assisted ES after each objective function call, the algorithm stores the function value and space location of the evaluated offspring. This information can be used to develop a surrogate model. However, using all of the evaluated points can substantially increase the CPU cost of the algorithm. Additionally, the more points we have, the harder it gets for the MLE to find the right value of the hyperparameters of the algorithm. We can use the local information around the parent to create a local model that can estimate the function value of the offspring that are being selected in the vicinity of their parents. Therefore, we need to find a strategy to identify the number of points k that should be used for training. In addition to the size of the training set, we need to find the right strategy to select the training set. Selecting the last k evaluated points, and selecting the k closest points to the minimum point are the two most popular approaches for choosing the training set. Based on our experiments, we argue that for the simple test problems considered here the two strategies usually end up choosing the same training data. That is because as SA-ES gets closer to

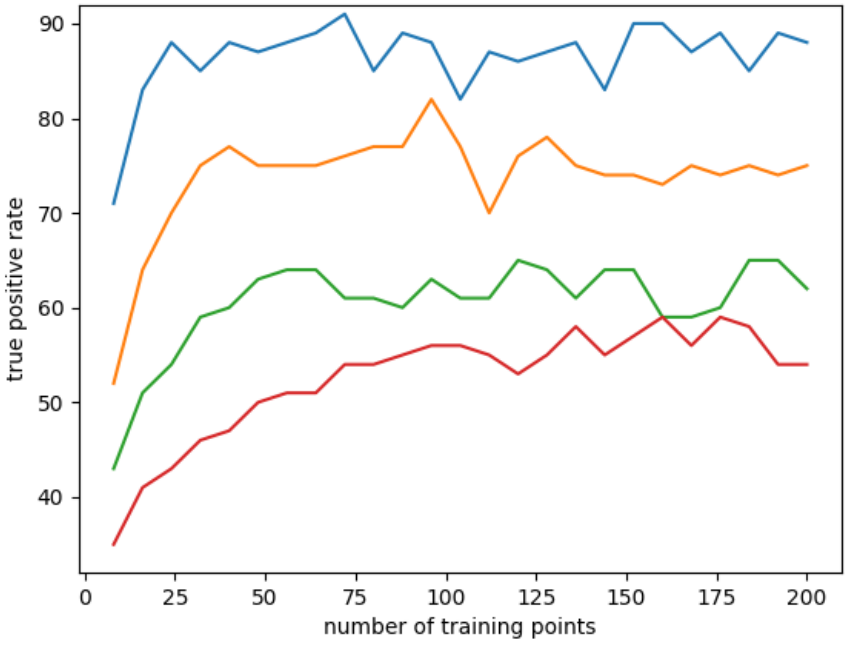

Figure 3.13: True positive rate as a function of number of training points for different number of dimensions. For the experiments, we used spheres with four (blue), eight (orange), 16 (green) and 32 dimensions (red).

the optimum solution, step-size decreases and therefore in each generation the newest offspring are selected with smallest mutation from the parent.

Next, we use the GPR with the specifications mentioned above to study the effects of the size of the training set on the performance of the algorithm. For this purpose, we use the notation of the true positive rate to evaluate the accuracy of the model in Algorithm 4. For each iteration of the outer loop of the Algorithm 4 we generate 1000 random offspring. We evaluate the random points using both the surrogate model and the objective function. Then, for different values of the size of the training set and the number of dimensions (4, 8, 10, 16 and 32) we measure the true positive rate of the surrogate model. Figure 3.13 shows the true positive rate of the quadratic spheres as a function of the number of training points. It can be seen that for a smaller number of training points the value of true positive rate is quite small. In other words, there is not enough information for creating the model. As we increase the number of training points the true positive rate gets better, until a certain point.

For the values larger than four times number of dimensions, the value of true positive rate do not get any better. Therefore, in this research we use the most recent $4n$ solutions as the training set for our modeling algorithm.

Chapter 4

Surrogate-Assisted (1+1) Evolution Strategy and Experiments

In the previous chapter, we implemented a simulated version of a surrogate assisted $(1 + 1)$-ES on a quadratic sphere. We used those simulations to study different aspects of SA-ESs. We introduced some quality measures such as fitness gain and true positive rate. Then, we implemented a series of experiments, using a simple $(1 + 1)$-ES, to approximate the optimum values of the hyperparameters of Gaussian process regression. In this chapter, we implement and test a real surrogate assisted $(1 + 1)$-ES with a real step-size adaptation mechanism, designed to work with the surrogate model. We also use Gaussian process regression for creating the surrogate model. We use the values that we derived in the previous chapter for the hyperparameters of the GPR and parameters of the step-size adaptation mechanism.

First, we define the test functions that we use for the evaluation of the algorithm. Then, we implement and test the parameters of the new step-size adaptation mechanism. Moreover, we use the notation of evaluation rate to reduce the computational cost of the algorithm. For the length-scale of GPR, we use both $3.5\sigma\sqrt{n}$ and the maximum likelihood estimate.

4.1 Test Functions

We use a Schwefel's function, a Quartic function and three types of sphere function. These test functions do not represent real-world problems. However, they are simple and easy to analyze. We can use them to study different aspects of the optimization algorithms. It is likely that if an optimization algorithm fails on these problems, it cannot solve real-world problems.

Table 4.1 contains the mathematical definition of each test function and some of the parameters of the test. $\sigma^{(0)}$ is the initial value of the step-size. $\mathbf{x}^{(0)}$ indicates that all tests start from a random point with a mean of zero and a variance of one. The

global minimum of all of these test functions is zero. Based on this notation, f_{stop} defines the stop criteria of the optimization. The number of objective function calls that the algorithm needs to reach f_{stop} is considered as a measure for the efficiency of optimization algorithms.

In the previous chapters, we discussed the importance of the quadratic sphere. Linear and cubic sphere have many of the characteristics of the quadratic sphere, and we expect the ES to show a similar behavior on all three spheres. But GP may be more suitable for one sphere than another. Evaluation of the performance of the surrogate assisted optimization algorithms on these three cases can develop a broader understanding of the behavior of these algorithms on spherical test functions.

Schwefel's Problem 1.2 [48] is a convex quadratic function that has been used in many benchmarks for optimization. Schwefel [48] estimated that the condition number of the Hessian of this function, when $n = 10$, is 175.1. The condition number of Hessian is used to study the curvature of the test functions. It can be calculated by dividing the highest eigenvalue of the Hessian matrix by the smallest eigenvalue in the Hessian matrix. Typically, the smaller the condition number of the Hessian matrix, the easier would be for the optimization algorithm to find the optimal solution. Büche et al. [13] asserted that the function's contours are hyper-ellipsoids which are rotated relative to the coordinate axes of the search space. Compared to the spheres, it might be more difficult for our modeling algorithm with an isotropic kernel to model this type of test function.

The Quartic function is a modification of the function $\sum_{i=1}^{n-1}[\beta(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$ where $\beta = 1$ [5]. This test function, when $n = 10$, has a condition number of Hessian equal to 49 at the optimizer. It can be considered as a simpler version of Rosenbrock function. Rosenbrock is a well-known optimization problem $\beta = 100$ which has a high value of condition number of Hessian at the optimizer (more than 3,500, when $n = 10$). As a result, it is very difficult to solve this problem with the isotropic mutation. Therefore, we use the problem with $\beta = 1$ for evaluation of our SA-$(1 + 1)$-ES.

Figure 4.1 shows the number of objective function calls that a simple $(1 + 1)$-ES, starting from the location of $\mathbf{x}^{(0)}$ and step-size of $\sigma^{(0)}$ needs to reach f_{stop}. The values are the result of 100 runs of the algorithm. Table 4.2 shows the median value for the

number of objective function call. By comparing the number of objective function calls that the surrogate-assisted ES needs to reach f_{stop} with the values in Table 4.2, we find to what extent surrogate models can improve our evolution strategy.

Name	Function	f_{stop}	$x^{(0)}$	$\sigma^{(0)}$
Linear Sphere	$f_{linearsphere} = (\sum_{i=1}^{n} x_i^2)^{\frac{1}{2}}$	10^{-8}	$N(0, I)$	1
Quadratic Sphere	$f_{quadraticsphere} = \sum_{i=1}^{n} x_i^2$	10^{-8}	$N(0, I)$	1
Cubic Sphere	$f_{cubicsphere} = (\sum_{i=1}^{n} x_i^2)^{\frac{3}{2}}$	10^{-8}	$N(0, I)$	1
Schwefel's function	$f_{schwefel's function} = \sum_{j=1}^{n} (\sum_{i=1}^{j} x_i)^2$	10^{-8}	$N(0, I)$	1
Quartic function	$f_{Quartic} = \sum_{i=1}^{n-1} [(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$	10^{-8}	$N(0, I)$	1

Table 4.1: Test functions and parameters of the test

Dimension	4	8	10	16	32
linear sphere	557.5	1038.0	1270.0	1982.0	3887.5
Quadratic sphere	290.5	535.5	673.0	1068.0	2084.0
Cubic sphere	211.0	383.0	472.0	764.5	1517.5
Schwefel's	397.0	1558.5	2367.0	6294.0	26167.0
Quartic function	1016.0	3164.0	4335.0	7539.0	15970.5

Table 4.2: Median number of objective calls $(1 + 1)$-ES

4.2 Step-Size Adaptation

Rechenberg in [43] showed that the success rate of an evolution strategy could be used to adapt the step-size. Using the analysis from the past chapters, we design a new step size adaptation mechanism with three coefficients for SA-$(1 + 1)$ ES (Algorithm 5) that adapts the step-size based on true positive rate and evaluation rate of the algorithm.

Evaluation rate is the success rate of a model evaluation, and true positive is the success rate of a objective function evaluation. Since, the cost of the model evaluation is usually not considerable compared to the cost of an objective function call, in this section we only use the true positive rate for adapting the step-size. In chapter 3 and based on Figures 3.5 and 3.6, we concluded that for a quadratic sphere when $n \to \infty$ the optimum value of true positive rate varies based on the value of noise-signal-ratio. We also assumed based on the experiments on a quadratic sphere in Figure 3.7 that the noise-to-signal ratio for suitable values of step-size is usually either higher than

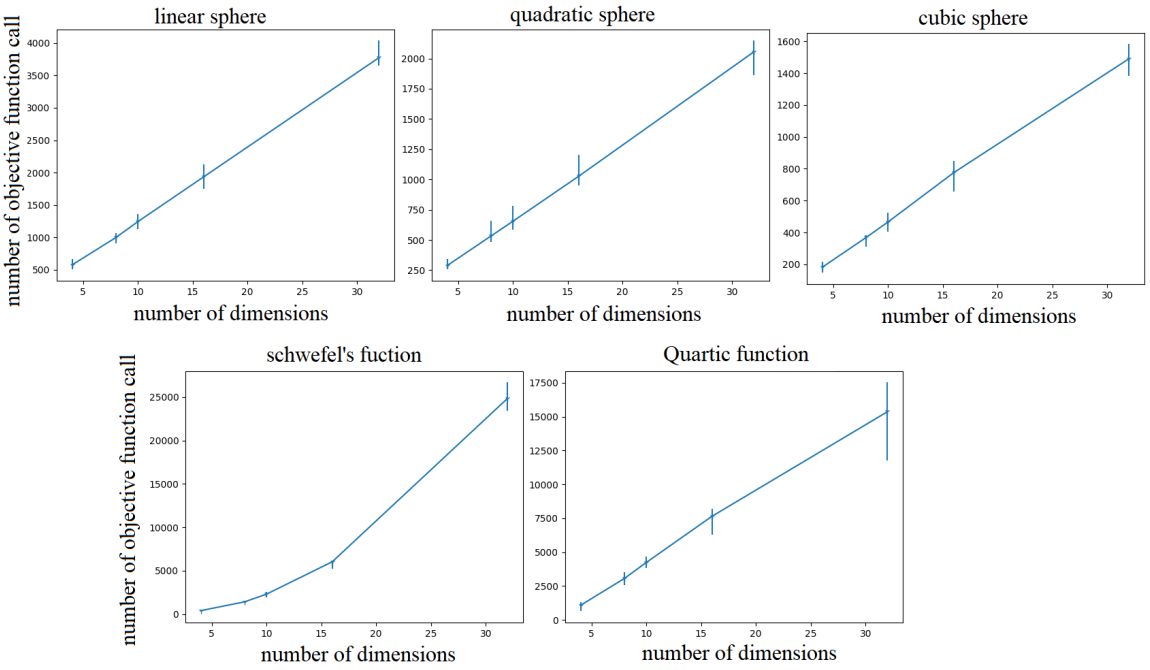

Figure 4.1: Number of objective function calls as a function of number of dimensions. Lines display the median and error bars show the range of the results of 101 runs of the algorithm.

one or some value close to one. Therefore, the optimum value for true positive rate is approximately 30%. 30% true positive rate also guarantees that on the test function mentioned above no matter how inaccurate the models are, the fitness gain of the algorithm is never worse than the $(1 + 1)$-ES. If we use values of true positive rate smaller than 20% and the model works inaccurately then the performance of the surrogate assisted (1+1)-ES might not be even as good as the simple (1+1)-ES with 1/5th rule step-size adaptation mechanism.

We design a SA-$(1 + 1)$-ES (Algorithm 5), where two coefficients c_2 and c_3 are used for adapting the step-size (in this section we do not use one of the coefficients $c_1 = 0$). Coefficient c_2 is used in line 16 where the offspring selected by the model evaluation is not successful based on the objective function. Second coefficient c_3 (line 14) is used to increase the step-size if an offspring is better than the parent based on the original function evaluation. Finding a balance between the values of these two coefficients has a crucial effect on the quality of the step-size adaptation mechanism. The inner loop of the algorithm continues until it finds an offspring that is superior to the parent based on the model evaluation. For other parameters such as D we use

Algorithm 5 Surrogate assisted $(1+1)$ ES

initialize $\mathbf{x} \in \mathbb{R}^n, c_1 \geq 0, c_2 > 0, c_3 > 0, P \leftarrow \emptyset$
while not happy **do**
  do
    $\mathbf{x}_1 \leftarrow \mathbf{x} + \sigma N(0, I)$
    Evaluate $\mathbf{x}_1$ using the model, yielding $f_\epsilon(\mathbf{x}_1)$
    if $f_\epsilon(\mathbf{x}_1) > f(\mathbf{x})$ **then**
      $\sigma \leftarrow \sigma e^{(-c_1/D)}$
    end if
  while $f_\epsilon(\mathbf{x}_1) > f(\mathbf{x})$
  $P \leftarrow P \cup [\mathbf{x}_1, f(\mathbf{x}_1)]$
  Update the model f_ϵ
  if $f(\mathbf{x}_1) < f(\mathbf{x})$ **then**
    $\mathbf{x} \leftarrow \mathbf{x}_1$
    $\sigma \leftarrow \sigma e^{(c_3/D)}$
  else
    $\sigma \leftarrow \sigma e^{(-c_2/D)}$
  end if
end while

the same values as what we used for Algorithm 1.

As was mentioned in chapter 3 we use $p_{truepositive}$ as the probability of an offspring, selected by the model, being better than its parent based on the original function evaluation. Therefore, in Equation 4.1 we can expect the step-size to remain unchanged if

$$-p_{eval}(1 - p_{truepositive})c_2 + p_{eval}(p_{truepositive})c_3 = 0 \tag{4.1}$$

which can be simplified as

$$-(1 - p_{truepositive})c_2 + (p_{truepositive})c_3 = 0 \tag{4.2}$$

$$\frac{c_2}{p_{truepositive}} = \frac{c_3}{1 - p_{truepositive}} \tag{4.3}$$

Kern et al. [31] asserted that for $(1+1)$-ES if we use $1 - 0.2$ for the coefficient of having a successful selection and 0.2 for the coefficient of having a failure (offspring not being better than the parent), the algorithm reaches the success rate of 20%. We can use a similar approach to reach the true positive rate of h by setting c_3 to be $1-h$ and $c_2 = h$. These coefficients also keep the balance in the Equation 4.3, which means

the true positive rate can be expected to reach the value of h when these coefficients are applied.

4.3 Performance with Respect to the True Positive Rate

In this section, we evaluate the performance of Algorithm 5 for different values of the coefficients. The goal in this section is to find the best value of true positive rate. We use the Gaussian process regression technique with normalization based on the minimum of the training data. In the last chapter, we showed that for modeling a quadratic sphere function, it is desirable to use $3.5\sigma\sqrt{n}$ for the value of length-scale. The training points of the model are the $4n$ last points that were evaluated by the objective function. All of the test function are in ten dimensions. The initialization and ending criteria of the optimization are based on the information provided in Table 4.1.

Figure 4.2 compares the speed-up of Algorithm 5 on different test functions. *Speed-up* is the number of objective function calls of the $(1 + 1)$-ES, based on the median results in Table 4.2, divide by the number of objective function calls of the SA-ES. The figure is based on the median of the results of ten runs of Algorithm 5. Error bars show the range of the values of speed-up for each value of true positive rate. We see in this figure that for simple problems like quadratic sphere, the speed-up is slightly better for the true positive rate of 40% compared to the case where the true positive rate is 30%. However, for more ill-conditioned problems such as the Quartic function and Schwefel's function the true positive rate of 30% provides the best performance. Therefore, we use the values of $h = 0.3$ and $1 - h = 0.7$ for, respectively, c_2 and c_3.

4.4 Performance with Respect to the Evaluation Rate

The accuracy of the model changes based on the location of the offspring. Points that are closer to the training data are approximated more accurately compared to the points that are far from the training data. Therefore, the offspring that are generated by smaller values of step-size are more accurate than the ones generated by bigger values of step-size. Although we use the true positive rate to have the model maintain a certain value of accuracy for the offspring that are generated by the SA-ES, the

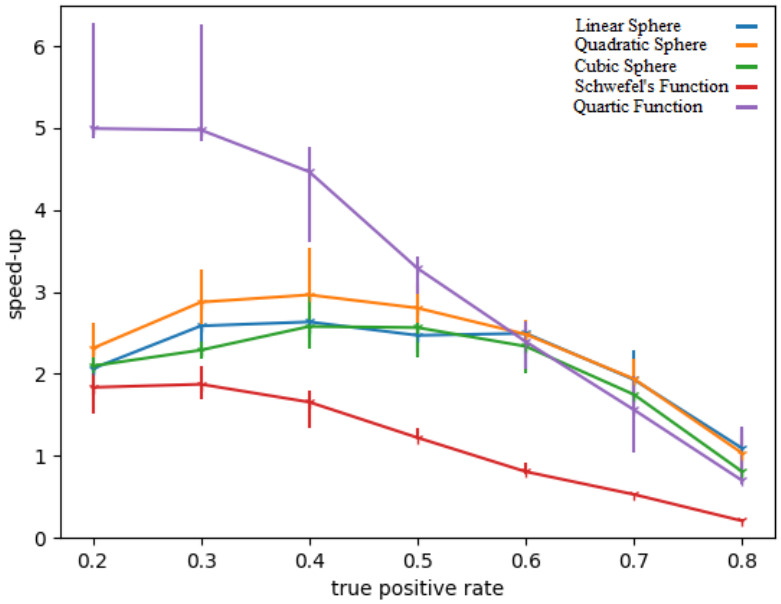

Figure 4.2: Speed-up as a function of true positive rate. Lines display the median result of 10 runs of the algorithm. Error bars show the range of the values of speed-up.

optimization algorithm might get into a situation where the accuracy of the model is too low for the value of step-size and therefore reject most of the generated points. In this situation, the model might not be able to find an offspring that is better than the parent for a large number of model evaluations. To prevent this issue, the step-size needs to decrease until the model evaluation is successful. We implement this strategy in Algorithm 5, where we use the coefficient c_1 to control the evaluation rate.

We evaluate the values of speed-up, true positive rate and evaluation rate for different values of c_1 from zero to 10^{-1}. The true positive rate is determined by counting the number of objective function calls that are successful divided by the total number of objective function calls of the algorithm. Evaluation rate is measured by generating 100 offspring and finding how many of those are superior to their parent based on the model evaluation. These 100 points do not affect the optimization procedure; they are only used to find the precise evaluation rate of the algorithm at each iteration of the outer loop.

Based on our experiments the average number of model evaluations decreases by a factor of more than 10 when c_1 changes from zero to 10^{-1} on the quadratic sphere.

However, $c_1 > 0$ might negatively impact the speed-up of the algorithm on some test function. For instance the speed-up of the quartic function decreased from 4.5 to 2.1 when c_1 changes from zero to 10^{-1}. Based on our experiments changing c_1 from zero to 10^{-3} does not affect the values of speed-up and true positive rate on our test functions, but has a considerable impact on the evaluation rate. Therefore, by setting $c_1 = 10^{-3}$ we can decrease the number of model evaluations without decreasing the speed-up of the algorithm. In the next experiments we use 10^{-3} for the value of c_1.

4.5 Performance with Respect to the Number of Dimensions

In this section, we investigate the quality of the SA-$(1 + 1)$-ES as a function of the number of dimensions. Figures 4.4 and 4.3 display the result of using Algorithm 5 on the benchmark problems. For most of the problems in the benchmark the value of speed-up decreases with respect to the number of dimensions. This coincides with results in Figure 3.11 where the value of the true positive rate of the models decreased when we increased the number of dimensions. Figures 4.4 and 4.3 display that the values of the true positive rate is mostly constant when we change the number of dimensions from 4 to 32, however, the speed-up changes significantly.

The values of speed-up vary based on the type of test function and the number of dimensions. For the linear sphere, the algorithm shows a speed-up by a factor of two to three. For the four-dimensional quadratic sphere, the algorithm is able to reach much higher speed-up, up to 5.5 and for the 32-dimensional quadratic sphere the speed-up by a factor of two is within the algorithm's reach. For the cubic sphere, speed-up changes in a smaller range between two and three. In Schwefel's function the surrogate model provides a speed-up of 3.5 for lower dimensions, but as the number of dimensions increases the model becomes almost useless. Speed-up of the Quartic function reaches a factor of up to 8.5 for four dimensions and decreases to speed-up of the factor of two for 32 dimensions. The plots also provide the value of objective function as a function of number of objective function calls which shows that the algorithm is capable of having linear convergence.

Figure 4.3 also compares the value of speed-up of the Kramer's approach ($t = 1$) with the values of speed-up of our SA-$(1 + 1)$-ES. Kramer's approach is a surrogate assisted ES with 1/5th rule step-size adaptation mechanism. Our SA-$(1 + 1)$-ES

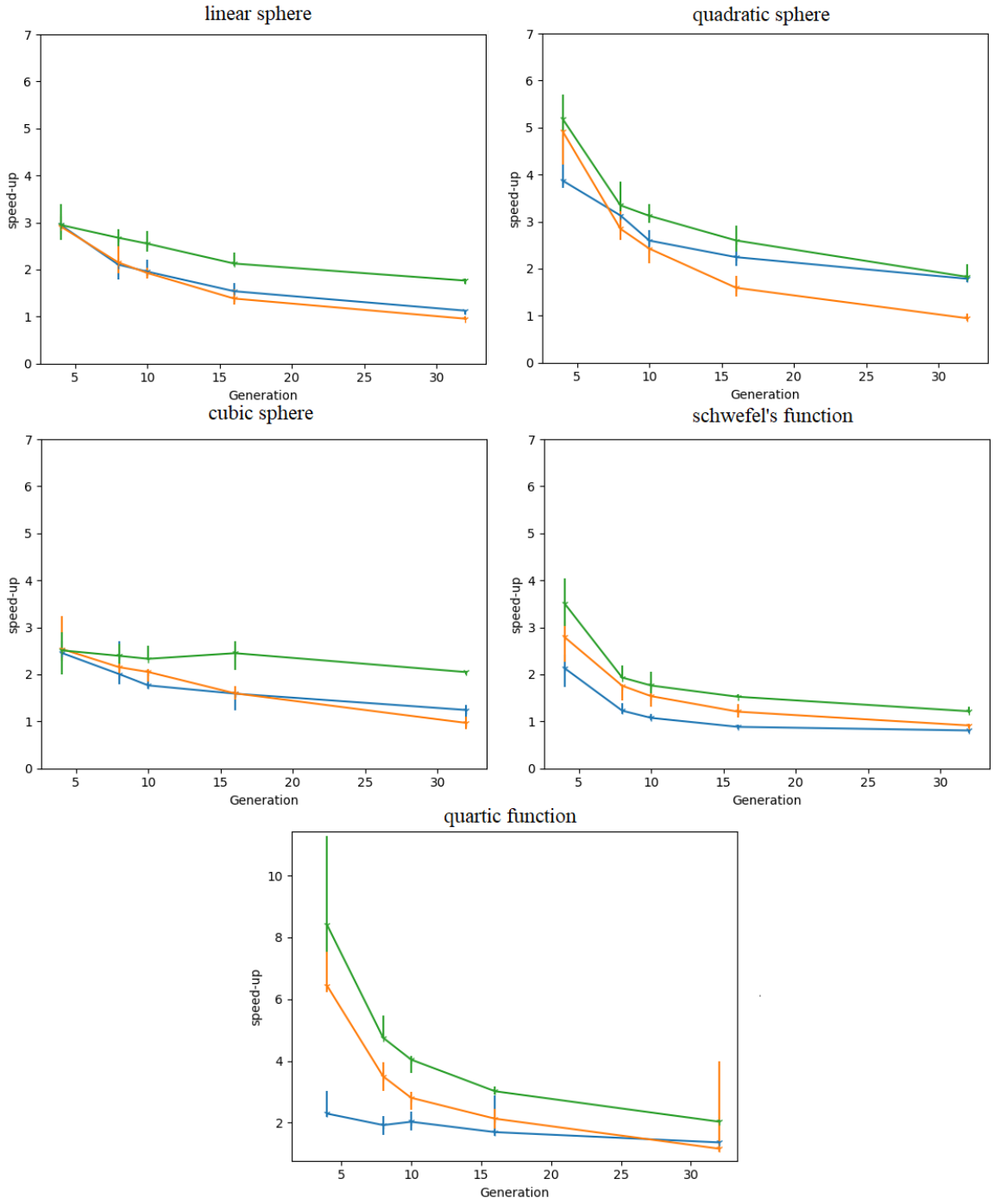

Figure 4.3: Speed-up as a function of number of dimensions. Figure displays the speed-up for Kramer's approach (blue), SA-(1+1)-ES when length-scale= $3.5\sigma\sqrt{N}$ (green) and SA-(1+1)-ES when length-scale is tuned using MLE (orange).

outperforms Kramer's approach by a factor of two to three on the Quartic function. The performance of our approach is either better or equal in all other cases as well.

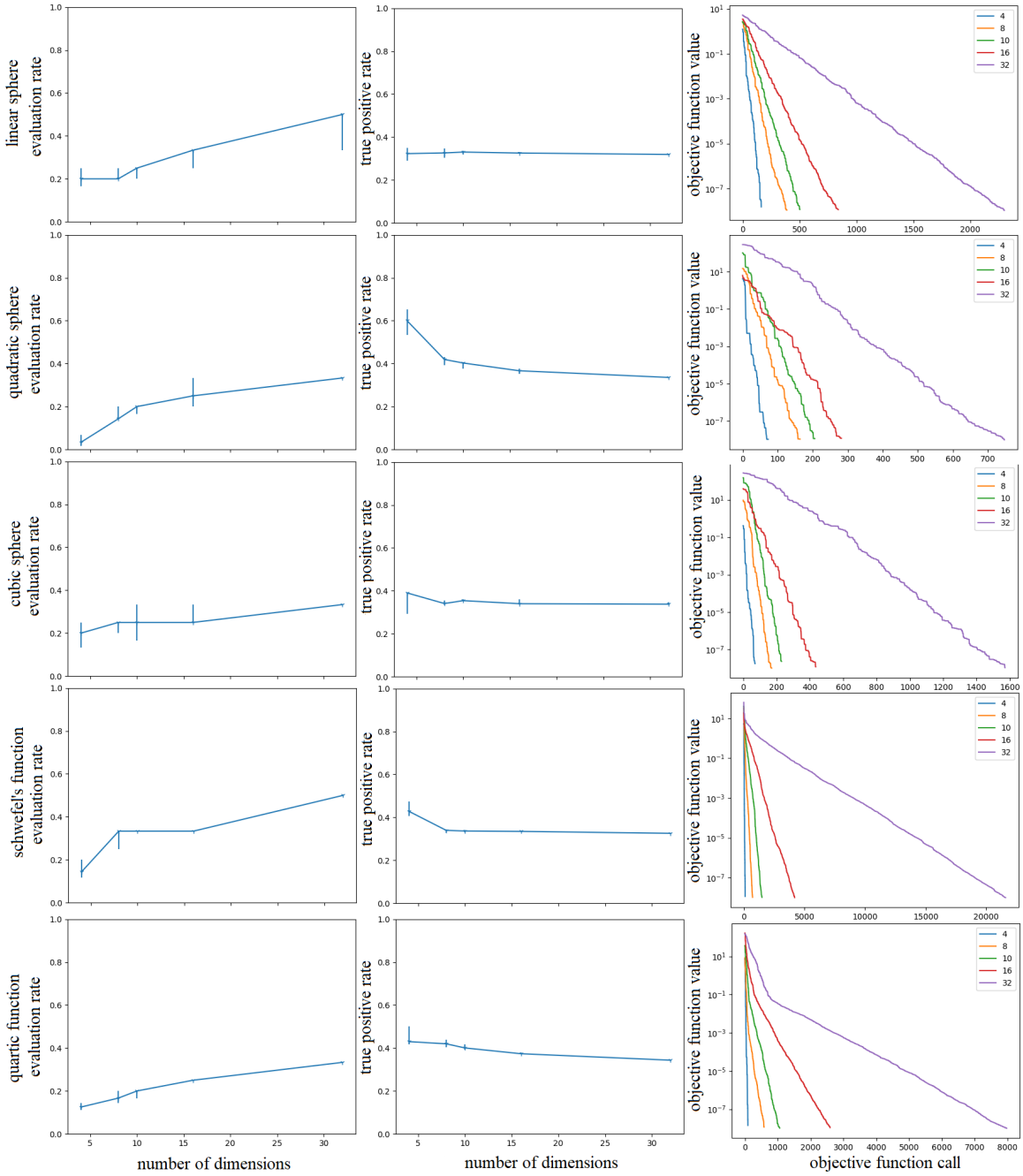

Figure 4.4: Objective function value as a function of number of objective function call. True positive rate and evaluation rate as a function of number of dimensions. The value of length-scale in the experiments is $= 3.5\sigma\sqrt{N}$.

.

On problems with a high number of dimensions (32) Kramer's approach performance is slightly worse than the $(1 + 1)$-ES. On Schwefel's function with 32 dimensions, the Kramer's algorithm sometimes was not able to reach the stop criteria and the

step-size decreased to zero during the optimization process.

4.6 Performance with Adaptive Length-Scale

So far we used the value of $3.5\sigma\sqrt{n}$ for the length-scale. This value was found for quadratic sphere and might not work on other test functions. In this section, we use the maximum likelihood estimate for adapting the length-scale. As mentioned before, we use the BFGS algorithm for finding the optimum length-scale of the MLE. The BFGS uses one restart in a range that was determined in chapter 3.

Figure 4.3 compares the speed-up of the SA-$(1+1)$-ES when we use the MLE to the case where we use $3.5\sigma\sqrt{n}$ for the length-scale. It can be seen that the speed-up of the algorithm with MLE for the test functions with smaller number of dimensions very close to the speed-up of SA-$(1+1)$-ES without using MLE. However, for larger number of dimensions the function $3.5\sigma\sqrt{n}$ provides better values for the length scale of the model.

Finally, Figure 4.5 shows the true positive rate and the evaluation rate of the SA-$(1+1)$-ES with MLE as a function of the number of dimensions. As expected the value of the true positive rate stays at 30%, and the evaluation rate is similar to the SA-ES without the MLE.

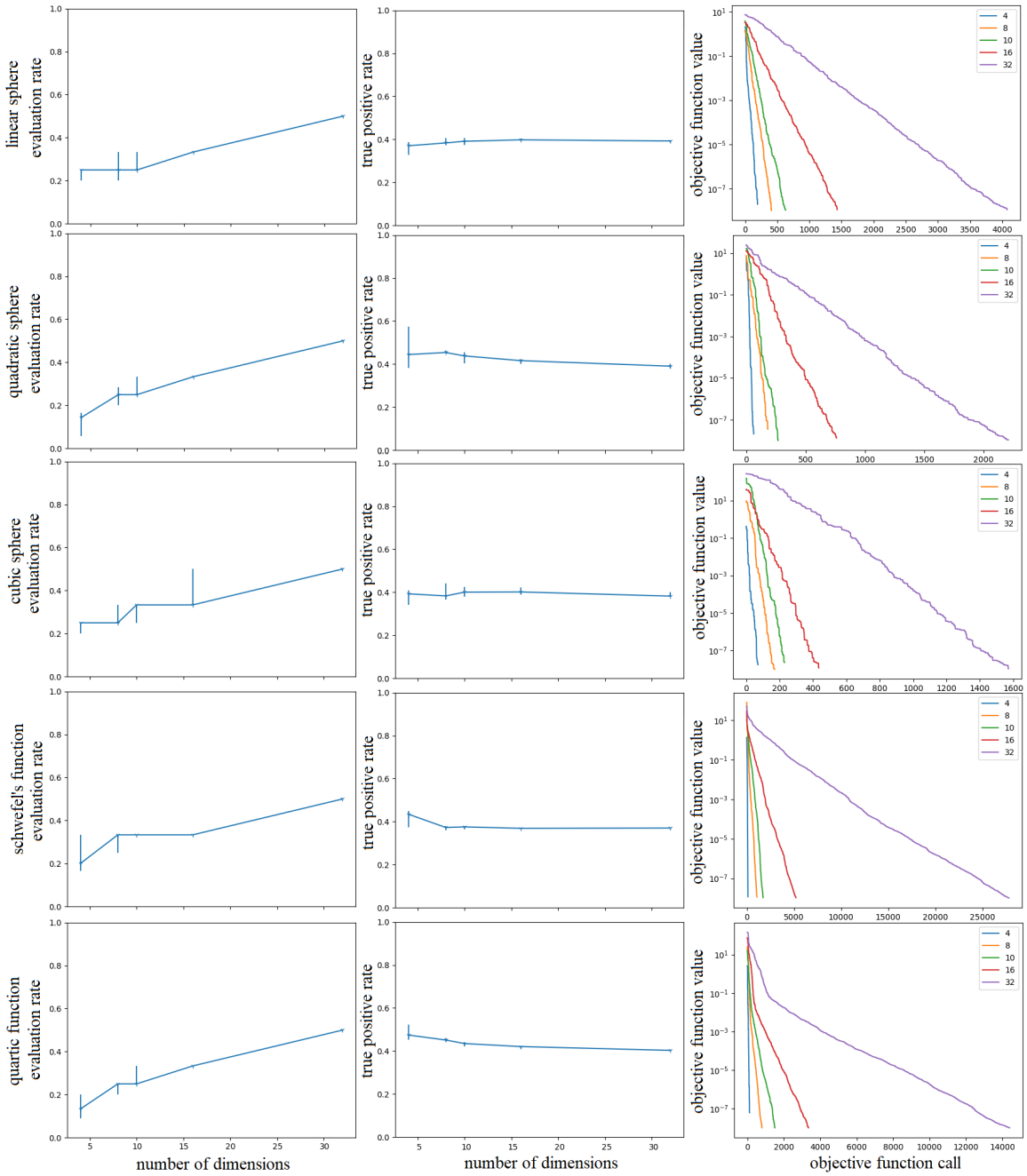

Figure 4.5: Objective function value as a function of number of objective function call. True positive rate and evaluation rate as a function of number of dimensions. The length-scale is adapted using the MLE.

Chapter 5

Conclusion

5.1 Summary

Surrogate modeling is a well-known approach that can decrease the cost of optimization by reducing the number of objective function evaluations. Although this field has been widely studied in the past 30 years, there are still some unanswered questions worth investigating. In particular for surrogate assisted evolution strategies, arguably the most important questions are how much these algorithms can improve the performance of evolution strategies and how changing the parameters of the optimization problem like the number of dimensions, and different types of difficulties such as noise (introduced in chapter 2) can affect the performance of these algorithms. Another question worth answering is the compatibility of the current step-size adaptation mechanisms used for surrogate-assisted evolution strategies, as they were all initially designed for evolution strategies without the surrogate model, and it is unclear whether they can provide optimal step-sizes once surrogate modeling is implemented.

The related work employs multiple heuristics, which makes these algorithms very difficult to analyze. To address these issues, we implemented a simple surrogate-assisted $(1 + 1)$ evolution strategy. In this study, we consider simple optimization problems. We believe that our findings made by analyzing these simple problems can be further generalized to more complex problems, especially with the help of other techniques such as Covariance Matrix Adaptation (CMA).

In chapter 3, we simulated the performance of the surrogate model. We replaced the surrogate model with noisy objective function. Using the theoretical analysis that exists in support of $(1 + 1)$-ES, we were able to a certain extent imitate the performance of a surrogate-assisted $(1 + 1)$-ES on a quadratic sphere. This analysis also helped us to suggest a step-size adaptation mechanism based on the quality of the model. To evaluate the quality of the algorithm, we introduced expected fitness

gain, true positive rate, and evaluation rate. Expected fitness gain is the average progress toward the optimum based on the objective function value. True positive rate evaluates the quality of the model. Evaluation rate is the probability of an offspring being superior to its parent based on the model evaluation. This measure affects the number of model evaluations of the algorithm. In this research we want to optimize both types of cost: the number of objective function calls and the number of model evaluations while giving more value to the number of objective function calls. Based on the performance of the simulated algorithms we observed that the value of true positive rate reaches approximately 30% when the model is very inaccurate. We also observed that if we set the true positive of the algorithm to 30%, the algorithm provides near optimal performance in most cases. In chapter 3, we also determined the values of some of the parameters of the algorithm such as length-scale and number of training data.

In chapter 4, we used a Gaussian process regression algorithm as the surrogate modeling technique. We used simple test functions to adjust the parameters of the new step-size adaptation mechanism. Defining the cost as the number of objective function evaluations that an optimization algorithm needs to reach the optimal solution within a certain accuracy, the surrogate-assisted $(1 + 1)$-ES shows a decrease in cost by a factor of up to 6 in spherical problems and up to 8 for Quartic function with small number of dimensions ($n = 4$), compared to the $(1 + 1)$-ES without the assistance of the surrogate model. However, as the number of dimensions increases the value of speed-up decreases.

Our approach shows better performance compared to Kramer's MM-ES, when $t = 1$, on most of the test functions. We argue that our step-size adaptation mechanism, which works based on maintaining the true positive rate of the algorithm at 30%, was successful at improving the performance of the optimization on the test functions in our benchmark. We were also able to reduce the number of model evaluations of the algorithm by introducing a coefficient for increasing the evaluation rate.

Moreover, we used the quadratic sphere to show that the optimum value of the length-scale changes with respect to the step-size and number of dimensions of the test problem. For the quadratic sphere with 4 to 32 number of dimensions, we estimated that the value of length-scale should be set to $3.5\sigma\sqrt{n}$. It is fair to say that this value

might be suitable only for a quadratic sphere with a small number of dimensions. In chapter 4, the performance of using a GPR with adaptive length-scale and a GPR with a length-scale of $3.5\sigma\sqrt{n}$ almost matched when we used the last $4n$ evaluated points for the training of the model. We can use this notation to further improve the process of hyper-parameter adaption of GP in evolution strategies.

5.2 Future Work

This research considered the most simple test cases to simulate and design a surrogate-assisted $(1+1)$-ES. Several directions of future research we would like to mention in this chapter consist of analyzing the behavior of this class of algorithm on problems with more difficulties.

- **Noisy optimization problems** This class of optimization problems is considered important test functions as they represent many real-world problems where there is no access to the exact value of the objective function. Using the analysis in chapter 3 and by adding the effects of overvaluation of the parents (as defined in [1]) to the calculation, we can develop a simulation of the performance of a SA-$(1+1)$-ES on a noisy quadratic sphere. These simulations can further help us to improve the design of surrogate-assisted evolution strategies for noisy optimization.

- **Ill-conditioned optimization problems** The idea behind this study was to understand surrogate-assisted ESs on simple functions, in the hope that it can be generalized for solving more difficult problems, especially with the help of complementary strategies such as covariance matrix adaption. To generalize the results of this study, we recommend a surrogate-assisted $(1+1)$-CMA-ES with the same evolution control used in this study, should be designed and evaluated. The parameters of the mutation operator of the CMA-ES might need to be adjusted once the surrogate modeling is added to the algorithm.

- **Parameter adaptation of the modeling algorithm** In this research, we found a direct relationship between the values of length-scale and the values of step-size and number of dimensions. This approach can be further developed

to use the information from past iterations of the algorithm to tune the best values for the hyperparameters of the algorithm.

Bibliography

[1] Dirk V Arnold. *Noisy Optimization with Evolution Strategies*, volume 8. Springer Science & Business Media, 2002.

[2] Dirk V Arnold and Hans-Georg Beyer. Local performance of the (1 + 1)-ES in a noisy environment. *IEEE Transactions on Evolutionary Computation*, 6(1):30–41, Feb 2002.

[3] Dirk V Arnold and Hans-Georg Beyer. A general noise model and its effects on evolution strategy performance. *IEEE Transactions on Evolutionary Computation*, 10(4):380–391, 2006.

[4] Anne Auger and Nikolaus Hansen. A restart CMA evolution strategy with increasing population size. In *2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1769–1776 Vol. 2, Sept 2005.

[5] Anne Auger, Nikolaus Hansen, JM Perez Zerpa, Raymond Ros, and Marc Schoenauer. Experimental comparisons of derivative free optimization algorithms. In *International Symposium on Experimental Algorithms*, pages 3–15. Springer, 2009.

[6] Samineh Bagheri. *Efficient Surrogate Assisted Optimization for Constrained Black-Box Problems*. PhD thesis, Leiden University, 2015.

[7] Samineh Bagheri, Wolfgang Konen, Michael Emmerich, and Thomas Bäck. Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets. *Applied Soft Computing*, 61:377 – 393, 2017.

[8] Lukáš Bajer. *Model-Based Evolutionary Optimization Methods*. PhD thesis, Charles University, 2018.

[9] Lukáš Bajer and Martin Holeňa. Surrogate model for continuous and discrete genetic optimization based on RBF networks. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 251–258. Springer, 2010.

[10] Lukáš Bajer, Zbyněk Pitra, and Martin Holeňa. Benchmarking gaussian processes and random forests surrogate models on the BBOB noiseless testbed. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 1143–1150. ACM, 2015.

[11] Lukáš Bajer, Zbyněk Pitra, and Martin Holeňa. Investigation of gaussian processes and random forests as surrogate models for evolutionary black-box optimization. In *Proceedings of the Companion Publication of the 2015 Annual*

Conference on Genetic and Evolutionary Computation, pages 1351–1352. ACM, 2015.

[12] Hans-Georg Beyer. *Ein Evolutionsverfahren zur mathematischen Modellierung stationärer Zustände in dynamischen Systemen*. Hochschule für Architektur und Bauwesen, 1989.

[13] Dirk Büche, Nicol N Schraudolph, and Petros Koumoutsakos. Accelerating evolutionary algorithms with gaussian process fitness function models. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(2):183–194, 2005.

[14] J. Dennis and R. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Society for Industrial and Applied Mathematics, 1996.

[15] Morris L. Eaton. *Multivariate Statistics: A Vector Space Approach*. Lecture Notes-Monograph Series. John Wiley and Sons Inc, 1983.

[16] Michael Emmerich, Alexios Giotis, Mutlu Özdemir, Thomas Bäck, and Kyriakos Giannakoglou. Metamodel—assisted evolution strategies. In *International Conference on Parallel Problem Solving from Nature*, pages 361–370. Springer, 2002.

[17] Alexander Forrester, Andras Sobester, and Andy Keane. *Engineering Design via Surrogate Modelling: A Practical Guide*. John Wiley & Sons, 2008.

[18] John J Grefenstette and J Michael Fitzpatrick. Genetic search with approximate function evaluations. In *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, pages 112–120, 1985.

[19] Nikolaus Hansen. References to CMA-ES applications (2009). *http://www.lri.fr/hansen/cmaapplications.pdf*.

[20] Nikolaus Hansen. Benchmarking a bi-population CMA-ES on the BBOB-2009 function testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference*, pages 2389–2396. ACM, 2009.

[21] Nikolaus Hansen, Dirk V Arnold, and Anne Auger. Evolution strategies. In *Springer Handbook of Computational Intelligence*, pages 871–898. Springer, 2015.

[22] Nikolaus Hansen, Anne Auger, Raymond Ros, Steffen Finck, and Petr Pošík. Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 1689–1696. ACM, 2010.

[23] Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. [research report] RR-6829, Institut National de Recherche en Informatique et en Automatique INRIA. 2009.

[24] Nikolaus Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 312–317, May 1996.

[25] Martin Holeňa, David Linke, Uwe Rodemerck, and Lukáš Bajer. Neural networks as surrogate models for measurements in optimization algorithms. In *International Conference on Analytical and Stochastic Modeling Techniques and Applications*, pages 351–366. Springer, 2010.

[26] Yaochu Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12, 2005.

[27] Yaochu Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61–70, 2011.

[28] Don Jones. Large-scale multi-disciplinary mass optimization in the auto industry. Presented at the Modeling and Optimization: Theory and Applications Conference (MOPTA), 2008.

[29] Arash Kayhani and Dirk V Arnold. Design of a surrogate model assisted (1 + 1)-ES. In *International Conference on Parallel Problem Solving from Nature*, pages 16–28. Springer, 2018.

[30] Stefan Kern, Nikolaus Hansen, and Petros Koumoutsakos. Local meta-models for optimization using evolution strategies. In *Parallel Problem Solving from Nature-PPSN IX*, pages 939–948. Springer, 2006.

[31] Stefan Kern, Sibylle D Müller, Nikolaus Hansen, Dirk Büche, Jiri Ocenasek, and Petros Koumoutsakos. Learning probability distributions in continuous evolutionary algorithms–a comparative review. *Natural Computing*, 3(1):77–112, 2004.

[32] Oliver Kramer. *Machine Learning for Evolution Strategies*. Springer, 2016.

[33] Dudy Lim, Yew-Soon Ong, Yaochu Jin, and Bernhard Sendhoff. A study on metamodeling techniques, ensembles, and multi-surrogates in evolutionary computation. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, pages 1288–1295. ACM, 2007.

[34] Ilya Loshchilov. *Surrogate-Assisted Evolutionary Algorithms*. PhD thesis, Université Paris Sud-Paris XI; Institut National de Recherche en Informatique et en Automatique INRIA, 2013.

[35] Ilya Loshchilov, Marc Schoenauer, and Michèle Sebag. A mono surrogate for multiobjective optimization. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, pages 471–478. ACM, 2010.

[36] Ilya Loshchilov, Marc Schoenauer, and Michèle Sebag. Black-box optimization benchmarking of IPOP-saACM-ES and BIPOP-saACM-ES on the BBOB-2012 noiseless testbed. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 175–182. ACM, 2012.

[37] Ilya Loshchilov, Marc Schoenauer, and Michele Sebag. Self-adaptive surrogate-assisted covariance matrix adaptation evolution strategy. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, pages 321–328. ACM, 2012.

[38] Ilya Loshchilov, Marc Schoenauer, and Michèle Sebag. Bi-population CMA-ES algorithms with surrogate models and line searches. In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 1177–1184. ACM, 2013.

[39] David JC MacKay. *Gaussian Processes - A Replacement for Supervised Neural Networks?* Tutorial Lecture Notes for NIPS, 1997.

[40] Zbigniew Michalewicz and Marc Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 4(1):1–32, 1996.

[41] Nikita Orekhov. Using gaussian processes as surrogate models for the CMA evolution strategy. Master's thesis, Czech Technical University in Prague, 2016.

[42] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian Process for Machine Learning*. MIT press, 2006.

[43] Ingo Rechenberg. Evolutionsstrategie: Optimierung technischer Systeme nach den Prinzipien der biologischen Evolution. *Frommann-Holzboog*, 1973.

[44] Rommel G Regis. Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. *Engineering Optimization*, 46(2):218–243, 2014.

[45] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533, 1986.

[46] Michael D Schmidt and Hod Lipson. Coevolution of fitness predictors. *IEEE Transactions on Evolutionary Computation*, 12(6):736–749, 2008.

[47] Hans-Paul Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie. Mit einer vergleichenden Einführung in die Hill-Climbing- und Zufallsstrategie*. Birkhäuser, 1977.

[48] Hans-Paul Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, Inc., 1981.

[49] Holger Ulmer, Felix Streichert, and Andreas Zell. Model-assisted steady-state evolution strategies. In *Genetic and Evolutionary Computation Conference*, pages 610–621. Springer, 2003.

[50] Jürgen Wakunda and Andreas Zell. A new selection scheme for steady-state evolution strategies. In *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*, pages 794–801. Morgan Kaufmann Publishers Inc., 2000.