Czech Technical University in Prague
Faculty of Information Technology
Department of Software Engineering

Master's thesis

Accelerating evolutionary algorithms by means of Gaussian processes

Bc. Andrej Kudinov

Supervisor: doc. Ing. RNDr. Martin Holeňa, CSc.

4th May 2015

Acknowledgements

In the first place, I would like to thank my supervisor doc. Ing. RNDr. Martin Holeňa, CSc. for his guidance and helpful approach. I would also like to thank Lukáš Bajer for helping with the frameworks and to MetaCentrum VO for providing the computing resources. Finally, I would like to thank Kateřina Kuncířová and my parents for their unrelenting support.

Declaration

In Prague on 4th May 2015 . . . .

Citation of this thesis

Kudinov, Andrej. *Accelerating evolutionary algorithms by means of Gaussian processes.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2015.

Abstrakt

Tato práce zkoumá výkon gaussovských procesů (GP) v souvislosti s metodou zvanou Covariance Matrix Adaptation Evolution Strategy (CMA-ES), state-of-the-art v oblasti evoluční optimalizace. Pro měření výkonu byly použity nichingové funkce ze soutěže CEC 2013, které jsou charakteristické vysokým počtem lokálních optim. Práce popisuje integraci CMA-ES a GP jako náhradního modelu a srovnává její výkon s metodou Model Guided Sampling Optimization.

Klíčová slova black-box optimalizace, gaussovské procesy, CMA-ES, náhradní model

Abstract

This thesis investigates the performance of Gaussian processes (GP) in the context of Covariance Matrix Adaptation Evolution Strategy (CMA-ES), the state-of-the-art evolutionary optimization method for black-box continuous optimization, using niching functions from the CEC 2013 competition, which are characterized by a high number of local optima. It describes the integration of CMA-ES and GP surrogate model and compares its performance to Model Guided Sampling Optimization.

Keywords black-box optimization, Gaussian process, CMA-ES, surrogate model

Contents

List of Figures

List of Tables

Introduction

Evolutionary computation became very successful during the past few decades in continuous black-box optimization. In such optimization, we have no prior information about the environment, like the first and second derivatives or the smoothness of the optimized function, and we can not calculate the value of that function analytically. In such cases, there is no option but to empirically evaluate the objective function and measure the obtained result.

In various real-world optimization problems, the evaluation of the objective function can be very expensive or time-consuming, e.g. protein's folding stability optimization [2], computer-assisted design [3] or job allocations in a computational grid [4], where one evaluation of the objective function can take seconds, minutes, hours or even days. In such cases, we need to keep the number of function evaluations as low as possible, without impairing the quality of expected results.

One of the most successful methods in this field is Covariance Matrix Adaptation Evolution Strategy, described in Section 2.1.1, which is quite robust with respect to a moderate noise and multi-modality of the objective function [5] in contrast to some other methods in the field of evolutionary computation. However, it requires a large number of the objective function evaluations, which is the main limitation of this method and all evolutionary algorithms (EAs) in general. This prevents using EAs on computationally expensive problems and remains as an open problem.

Gaussian processes (GPs) are frequently used in conjunction with Gaussian process regression, a.k.a. Kriging, when the objective function is modeled using training data. Then it can be used for various purposes, e.g. for the error estimation or as a surrogate model used during optimization instead of the original objective function.

The objective of this thesis is to examine the existing methods in the field of black-box optimization, propose the strategy of integration of chosen approaches in order to accelerate the computation by minimizing the number of the objective function evaluations and test the integrated methods on the set of chosen testing functions.

Chapter 1 introduces the topic of the black-box optimization and GPs, Chapter 2 describes examined methods in the field of evolutionary computation and the analysis and design of the integration of the involved methods. Chapter 3 describes the implementation and testing the integrated methods and discusses the obtained results. Chapter 3.2.3.3 concludes the thesis.

CHAPTER 1

State of the Art

The present section describes the problems in the field of black-box optimization depending on function properties and introduces theoretical fundamentals of GPs.

1.1 Black-box optimization

The continuous black-box optimization has to deal with different kinds of problems which make the process of finding the global optimum more difficult.

Multi-modality
An objective function is called *multi-modal* when it has more than one optima. Optimization algorithms in the case of such functions tend to get trapped in one of local optima and there is no guarantee that after the optimization process the found optimum is the global one. There are two most commonly used approaches of dealing with this problem, restarting the search process for many times to increase the probability of finding global optimum or using some niching techniques in order to explore local optima and hopefully find the global one.

High-dimensionality
With the increasing number of dimensions of the input space, the volume of the search space increases exponentially. Some algorithms, which are successful in the case of low-dimensional functions, can became useless for the large dimensions. This effect is called the curse of dimensionality.

Non-separability
A function is called separable when the optimum of the function can be obtained by performing n independent one-dimensional optimizations in each dimension. This property breaks the curse of dimensionality, as the complexity of the optimization process grows only linearly with n.

Functions where we can perform independent one-dimensional searches only for a subset of coordinates are called *partially-separable.*

Noise

Function f is called noisy if for the same x different values $f(x)$ can be obtained, perturbed by some random value ξ. Noisy functions are much more difficult to optimize because the information obtained from the function evaluations is less precise then in the case of noiseless function. Moreover, the first and second derivatives can be difficult to obtain and or just useless, which limits the using of gradient-based methods.

Ill-conditioning

If f is quadratic function, such as $f \to \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x}$, where $\boldsymbol{H}$ is symmetric positive-definite, is called *ill-conditioned* if the condition number of H is much larger then 1. It can cause problem in the case of some methods during the learning process of appropriate metric, as the algorithm makes too short or long search steps.

There are more function properties which can cause problems during the optimization process, like *multi-objectivity*, *deceptiveness* or *dynamism*, which are described in [6]. The performance of the compared method can be strongly affected by this properties, different methods handle the respective properties differently and there is no universal method for all of them.

1.2 Gaussian Processes

GP is a random process such that any finite sequence $X_1, \dots, X_k$ of its mutually different random variables has a multi-variate Gaussian distribution. GP is defined by its mean value and covariance matrix described by a function with relatively small number of hyper-parameters, which are usually fitted by the maximum likelihood method. The probability density of the multivariate Gaussian distribution of $\mathbf{y}_N$, conditioned on $\mathbf{X}_N$, is $p(\mathbf{y}_N|\mathbf{X}_N)$. GP is first trained with N data points from the input space $\mathbb{X}$,

$$\mathbf{X}_N = \{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{R}^D\}_{i=1}^N$$

with known input-output values $(\mathbf{x}_N, \mathbf{y}_N)$, then it is used for predicting the $(N+1)$-st point. The conditional density of the extended vector is

$$p(\mathbf{y}_{N+1}|\mathbf{X}_{N+1}) = \frac{\exp(-\frac{1}{2}\mathbf{y}_{N+1}^{\mathrm{T}}\mathbf{C}_{N+1}^{-1}\mathbf{y}_{N+1})}{\sqrt{(2\pi)^{N+1}\det(\mathbf{C}_{N+1})}}$$

where $\mathbf{C}_{N+1}$ is the covariance matrix of the $(N+1)$-dimensional Gaussian distribution. The covariance matrix can be expressed as

$$\mathbf{C}_{N+1} = \begin{pmatrix} \mathbf{C}_N & \mathbf{k} \\ \mathbf{k}^{\mathrm{T}} & \kappa \end{pmatrix}$$

where κ is is the variance of the new point itself, $\mathbf{k}$ is is the vector of covariances between the new point and training data and $\mathbf{C}_N$ is the covariance of the Gaussian distribution given the N training data points.

Covariance functions provide prior information about the objective function and express the covariance between the function values of each two data points $\mathbf{x}_i$, $\mathbf{x}_j$ as $cov(f(\mathbf{x}_i), f(\mathbf{x}_j)) = k(\mathbf{x}_i, \mathbf{x}_j)$. Consequently, the matrix of values of the covariance function for any N data points $x_1, \ldots, x_N$ needs to be positive semidefinite.

Chapter 2

Analysis and design

The following section introduces theoretical fundamentals of the approaches addressing our task. The first, Covariance Matrix Adaptation Evolution Strategy (CMA-ES), is one of the most successful methods in black-box optimization. The second method, Model Guided Sampling Optimization (MGSO), is one of the recent implementations of GPs. The third employed approach is surrogate modeling, which we will use in conjunction with the first method.

2.1 Methodology of black-box optimization

For the sake of this thesis several methods were examined:

- CMA-ES [5],
- Efficient Global Optimization (EGO) [7],
- MGSO [8],
- GP as a surrogate model [6].

The combination of two listed methods was chosen for the integration, CMA-ES with GP as a surrogate model. MGSO was chosen for comparison with the performance of the integrated methods. All employed methods are described in the following sections.

2.1.1 CMA-ES

CMA-ES [5] is one of the most successful stochastic, derivative-free methods used for continuous black-box optimization. The covariance matrix adaptation has the ability to learn and adapt the optimization process in order to increase the probability of finding a better solution.

New points are sampled normally distributed

$$\boldsymbol{x}_i \sim m + \sigma \boldsymbol{y}_i, \quad \boldsymbol{y}_i \sim \mathcal{N}_i(0, \mathbf{C}) \quad \text{for } i = 1, \dots, \lambda$$

as perturbation of m, where $m \in \mathbb{R}^n, \sigma \in \mathbb{R}_+$, $\mathbf{C} \in \mathbb{R}^{n \times n}$ and λ is the sample size. The mean m represents the favorite solution, the step-size σ controls the step length and covariance matrix $\mathbf{C}$ determines the shape of the distribution ellipsoid and represents the pairwise dependencies between the variables. The mean and the covariance matrix are updated after each search step.

Let $\boldsymbol{x}_{i:\lambda}$ the i-th ranked solution point, such that $f(\boldsymbol{x}_{1:\lambda}) \leq \cdots \leq f(\boldsymbol{x}_{\lambda:\lambda})$. The mean is

$$m \leftarrow m + \sigma \boldsymbol{y}_w, \quad \boldsymbol{y}_w = \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}$$

where $w_1 \leq \cdots \leq w_\mu < 0$ and $\sum_{i=1}^{\mu} w_i = 1$. The best μ solutions are selected from the set of new solutions and weighted intermediate recombination is applied, which rewards candidate solutions according to their fitness value.

The main objective of the covariance matrix adaptation is to increase the probability of successful steps to appear again. The covariance matrix update consists of two stages, rank-one update and rank-μ update. The rank-one update computes the evolution path $\boldsymbol{p}_c$ of successful moves of the mean, determined in a similar way as the evolution path $\boldsymbol{p}_\sigma$ of the step-size.

The rank-μ update computes the covariance matrix $\mathbf{C}^+$ as a weighted sum of covariances of successful steps of μ best individuals. In a similar way $\mathbf{C}^-$ is computed for μ worst individuals to be used for covariance matrix $\mathbf{C}$ update in CMA-ES. The update of $\mathbf{C}$ itself is a replace of previously accumulated information by a new one with corresponding weights of importance.

2.1.2 GP as a surrogate model

Surrogate modeling is a technique used in optimization in order to decrease the number of expensive function evaluations. Surrogate model, which is a regression model of suitable kind (in our case a GP), is constructed by training with known values of the objective function for some inputs first, and then it is used by the employed evolutionary optimization algorithm instead of the original objective function (in evolutionary optimization usually called fitness) during the search for the global optimum.

Every regression model approximates the original fitness function with some error. To prevent the optimization from being mislead from such an erroneous approximation, it is necessary to use the original fitness function for some subset of evaluations. That subset is determined by the evolution control (EC) strategy.

The *individual-based* EC strategy consists in determining the subset of individuals evaluated by the fitness function in each generation. First, $\lambda'' < \lambda$ points are sampled from $N(m, \sigma\mathbf{C})$, where m is the mean, σ is the step-size and **C** stands for the covariance matrix (see Section 2.1.1 for details). These points are evaluated by the original fitness function and included in training the model. Then, λ' points are sampled from the same distribution, where λ' is several to many times larger then λ. Subsequently, $\lambda - \lambda'$ points are chosen according to some criterion, e.g. fitness value, and used in the evaluation by the original fitness function [9].

The *generation-based* EC determines the frequency of whole generations evaluated by the original function. A generation is evaluated by the original fitness function and then the model is trained using obtained values. The number of consequent model-evaluated generations can be determined also dynamically, as introduced in so-called *adaptive* EC strategy [10], when the deviation between the original and the model fitness function is counted and then it is decided whether to evaluate with the original fitness or with the model.

Determining the most suitable EC parameters, however, is an open problem, which depends on the properties of the fitness function, current performance of the surrogate model and it changes during the optimization process.

2.1.3 EGO

The crucial idea of the EGO algorithm is to fit a response surface to collected data evaluated by the objective function. It tries to balance between finding the minimum of the surface and improving the approximation by sampling in the areas where the prediction error may be high. The response surface methodology is based on modeling the objective function by means of stochastic processes. The response surface is calibrated by fitting the stochastic process to data.

Modeling the objective function by the response surface has three main advantages. First, searching for the global optimum requires less objective function evaluations, as the model brings some insight into the modeled function and allows the stochastic process to make a conclusions and regulate the search process instead of moving step-by-step along some trajectory. Second, the response surface approach provides a credible stopping rule based on the expected improvement from further searching. Third, it provides fast approximation to the computer model which can be used to identify important variables [7].

2.1.4 MGSO

MGSO [8] has the ability to use regression model for prediction and error estimation in order to get a probability of obtaining a better solution. It was inspired by two previously proposed methods in the field of black-box optimization. The first method, Estimation of Distribution Algorithms [11], creates a new set of solutions for the next generation using estimated probability distribution from previously selected candidate solutions. The second approach is surrogate modeling, described in Section 2.1.2.

MGSO was proposed as an alternative method for Jones' Efficient Global Optimization (EGO) [7]. Unlike EGO, MGSO produces a whole population of suggested solutions, instead of selecting a single solution and maximizing a chosen criterion. The selection of candidate solutions is performed by sampling the probability of improvement (PoI) of the GP model, which serves as a measure of how promising the chosen point is for locating the optimum. PoI is determined by means of a chosen threshold T and the knowledge of the objective function shape modeled by the current GP model.

2.2 Integration

The section describes the integration analysis and design of the chosen approaches, CMA-ES with GP as a surrogate model (denoted hereafter S-CMA-ES). It introduces the requirements and describes some structural and behavioral aspects of the integration design.

2.2.1 Activity diagram

The list below enumerates all required activities provided by a simple user interface:

- perform optimizations,
- load results,
- get the speed-up of the compared methods compared to CMA-ES,
- perform one-tailed statistical test,
- obtain best parameter settings,
- export results.

The activity diagram is illustrated in Figure 2.1. Most actions are performed sequentially, as they require the results of all preceding activities. When all results from performed optimizations are obtained, they can be loaded and processed. The processing of the results consists of three steps. The evaluation of the speed-up values of examined methods and the successive evaluation of the best parameters for each method-function combination. Independently of these activities, we can also perform one-tailed statistical test on the specified significance level. The last activity, which depends on all preceding activities, is an export of previously processed results to some human readable format.

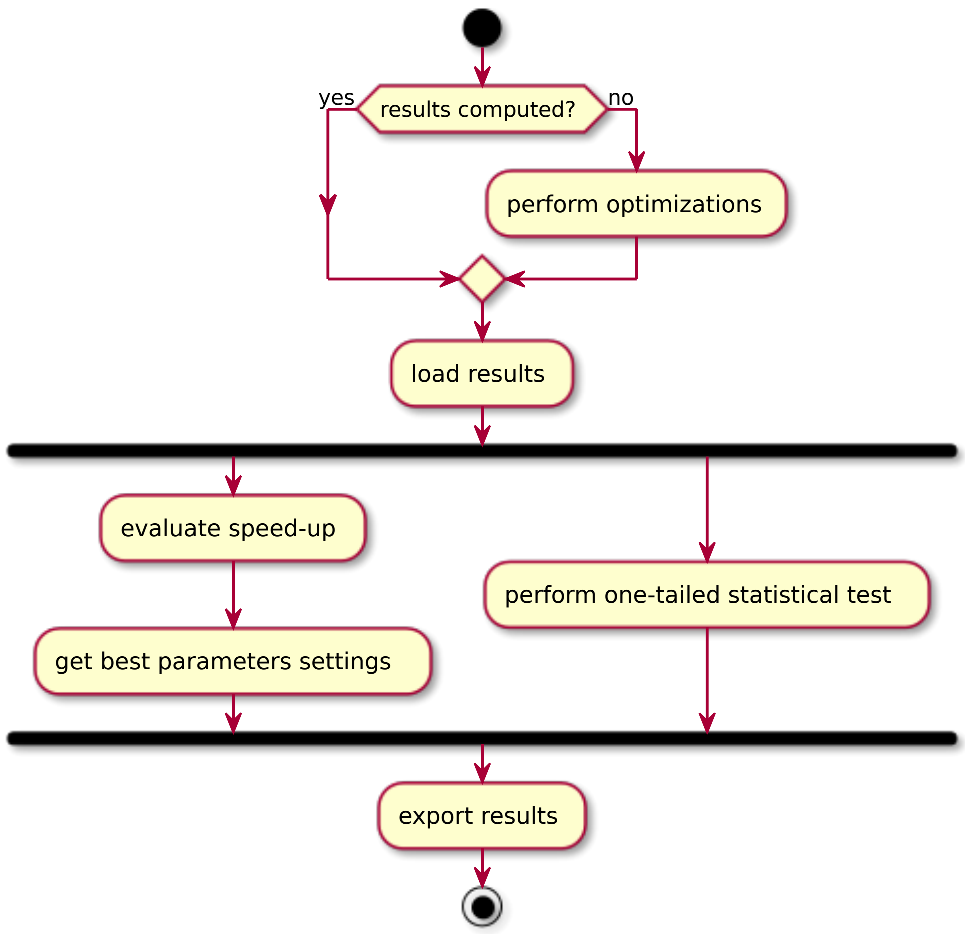


Figure 2.1: Activity diagram.

2.2.2 Components

Figure 2.2 illustrates the component diagram. In the case of MGSO, the implementation GPEDA (Gaussian Process sampling EDA algorithm) by Lukáš Bajer and Viktor Charypar [12] was used. In the case of GPs, the implementation of GMPL (Gaussian Processes for Machine Learning), based on the book written by Carl Edward Rasmussen and Hannes Nickisch [13], was used, which provides all necessary functionalityin the context of GP model and it is also used in the implementation of MGSO. The benchmark from the CEC 2013 competition, described in section 3.2, was used for performance testing.

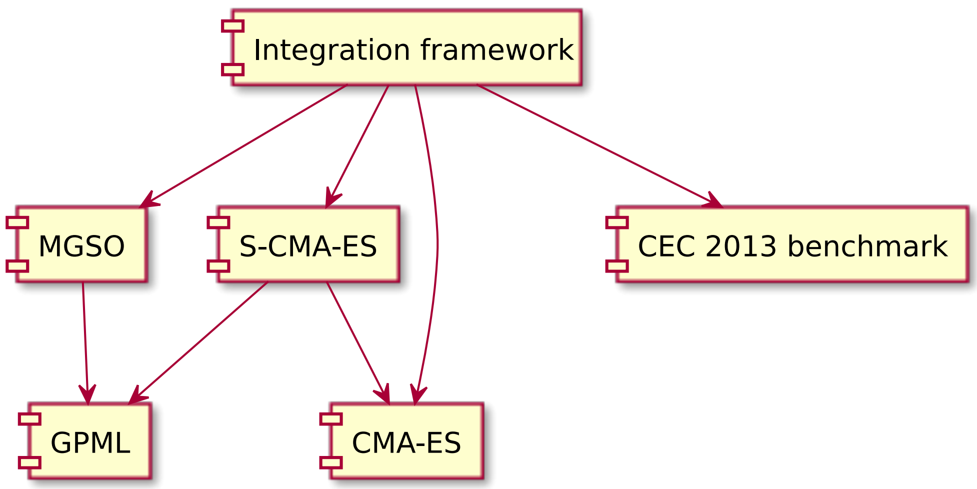


Figure 2.2: Component diagram.

2.2.3 Class diagram

The integration framework was first designed in the procedural style, decomposed into several batch files. However, later it was divided using OOP normalization into into classes designated for performing corresponding procedures, as illustrated in Figure 2.3. The advantage of such decomposition is the ability to use encapsulation, inheritance, etc., in order to reuse the code, assign responsibilities and generally improve and simplify the maintenance of the code.

2.2.3.1 Main interface

The main interface, represented by the class *PerformanceTestManager*, was implemented for users convenience. It implements the facade design pattern and allows to launch specific groups of procedures automatically to spare a user executing all procedures manually. It delegates responsibilities to six single-service classes:

- *ComputationManager*,
- *ResultDAO*,
- *SpeedUpEvaluator*,
- *StatisticsTester*,
- *BestParamsEvaluator*,
- *ResultPrinter*.

Except for the first one, all listed classes are abstract and define the required interface and can be extended by some other classes with different functionality if necessary.

ComputationManager is responsible for running all planned optimizations performed by the corresponding methods. This class is described in the next section in detail, as it deserves a closer look.

Following three classes are responsible for the result processing. The first, the *SpeedUpEvaluator*, returns a matrix containing the speed-up values acquired from the performance comparison of two methods for all parameter combinations defined in the given schedule.

The second class, the *StatisticsTester*, performs one-tailed statistical test according to the given value of significance level. For that purpose it uses the results obtained from optimization processes. The output of this class is a matrix of Boolean values (zeros and ones) which signify the rejection or acceptance of the null hypothesis.

The last class in this category, the *BestParamsEvaluator*, returns the structure containing best observed parameters determined by passed speed-up values taking into account the average value of the speed-up for the respective method-function combination.

There are two classes interacting with the file system. The first, the *ResultDAO*, provides the access to the persistent storage and is responsible for loading and saving obtained results. The *ResultPrinter* exports processed results gained from result processing to required format.

2.2.3.2 Computations

Figure 2.4 shows the class diagram in the context of planning and performing the optimizations. The class *ComputationManager* contains a specified set of predefined *Computation* instances. This class specifies, which methods are going to be used and which parameter setting are going to be examined.

The *Computation* class is responsible for fulfilling the schedule associated with the specified method, thus the instances of this class are determined by the corresponding *Schedule* and *Method* in the class constructor.

Instances of the *Schedule* are defined by sets of values corresponding to each examined parameter and thereby determine the state space of the computation. As it is illustrated in Figure 2.6 For each *Method* there is a corresponding subclass of the *Schedule* class. Because schedules can share many common properties, they can be specified in a parent schedule and accessed through the method *getProperty* in the same way as its own properties. For example, the schedules *indiSchedule* and *geneSchedule* have common properties defined in the *scmaesSchedule*, thus it can be set as a parent for both schedules so they can access its properties as their own.

The assembling of the *Options* objects is not trivial and it varies in accordance with used method. For this reason, this responsibility is assigned to the

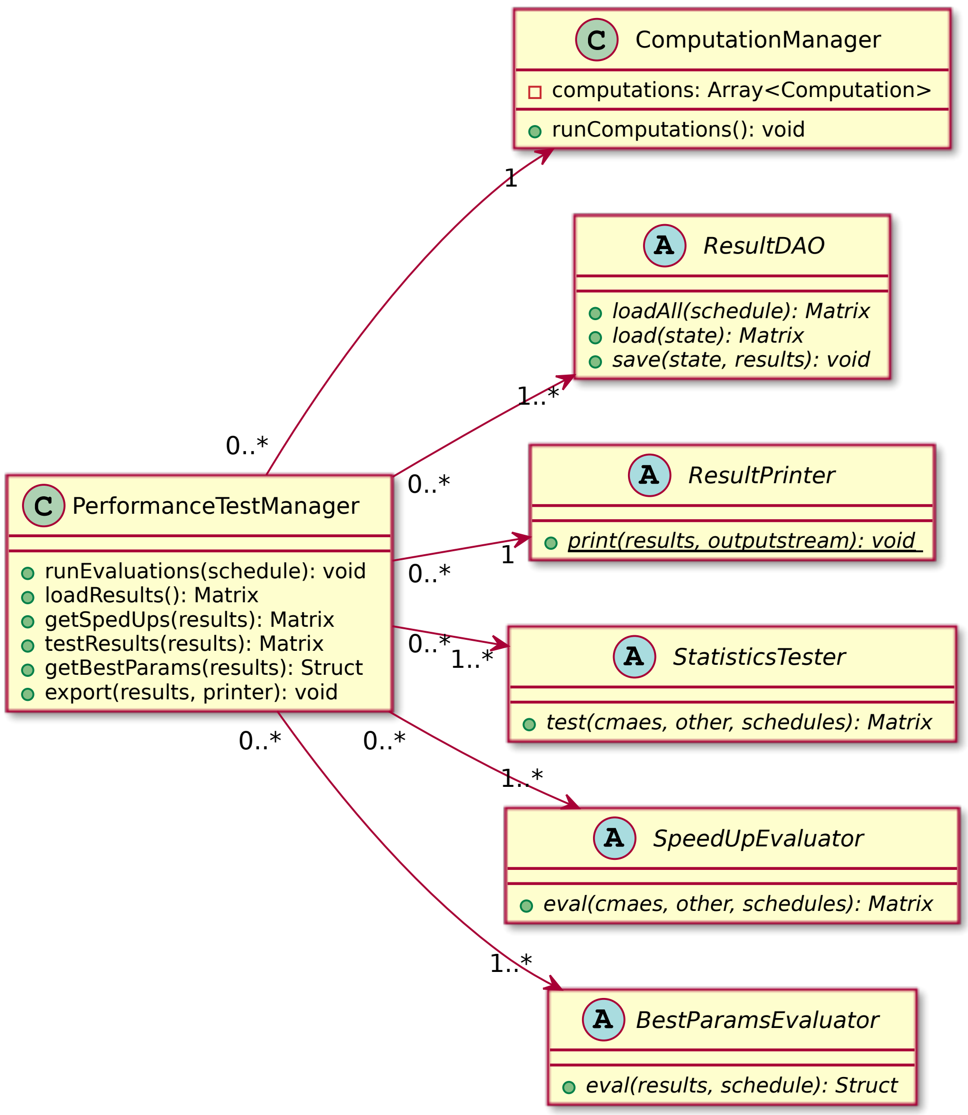

ComputationManager
computations: Array<Computation>
runComputations(): void
ResultDAO
loadAll(schedule): Matrix
load(state): Matrix
save(state, results): void
ResultPrinter
print(results, outputstream): void
StatisticsTester
test(cmaes, other, schedules): Matrix
SpeedUpEvaluator
eval(cmaes, other, schedules): Matrix
BestParamsEvaluator
eval(results, schedule): Struct
PerformanceTestManager
runEvaluations(schedule): void
loadResults(): Matrix
getSpedUps(results): Matrix
testResults(results): Matrix
getBestParams(results): Struct
export(results, printer): void
0..*
1
0..*
1..*
0..*
1
0..*
1..*
0..*
1..*
0..*
1..*

Figure 2.3: Fragment of the class diagram – main interface.

external class *OptionsFactory*, which implements the factory design pattern. It sets all parameters in *Options* according to the given *Schedule* and the current *State* of the *Computation*. The described fragment of the class diagram is illustrated in Figure 2.5.

The class *ResultDAO* is responsible for persisting obtained results. It stores the result after each *Method* run and assigns it a unique name determined from the given *State*. After each *Method* run the internal state is incremented and the whole process is repeated until the schedule is fulfilled.

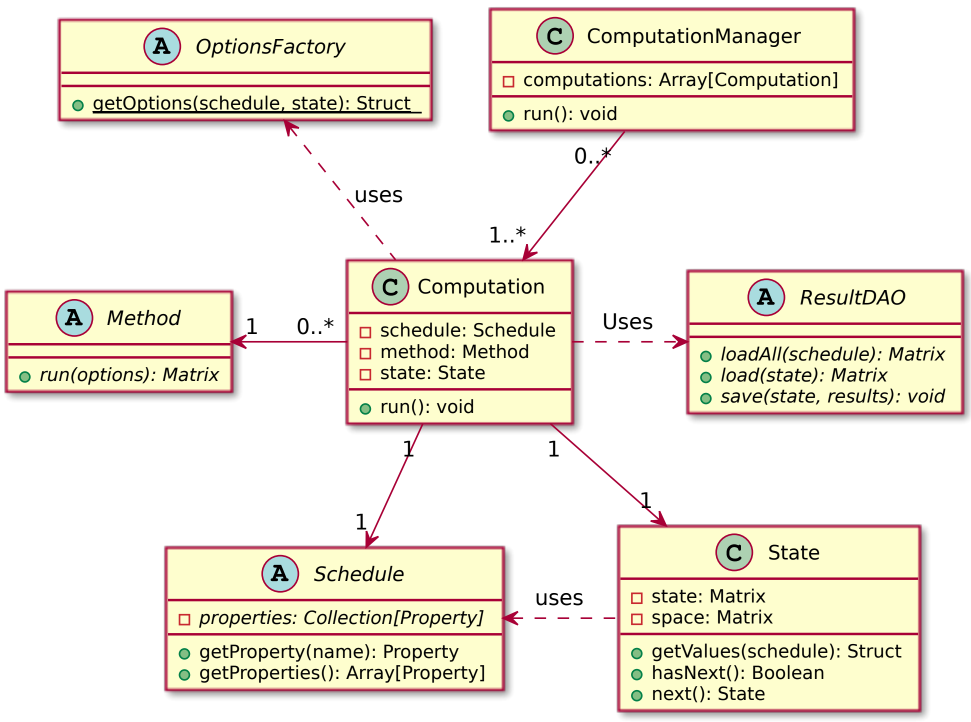


Figure 2.4: Fragment of the class diagram – computations.

2.2.3.3 Integration

Figure 2.7 shows the simplified class diagram in the context of the integration. All methods are represented by the corresponding subclasses of the class *Method*. The integration of CMA-ES and GP surrogate model is implemented in the classes *SCMAES* and *SurrogateManager*. The *SCMAES* is based on the native CMA-ES implementation, but instead of original CMA-ES sampling of new solutions, it uses the *SurrogateManager* in order to control sampling by employing a surrogate model. The way how the model is exploited is determined by used EC strategy. EC strategy options are specified in two classes, *GenerationEC* and*IndividualEC*, both inheriting from the abstract class *EvolutionControl*.

Figure 2.8 demonstrates the *SurrogateManager* with all its auxiliary classes. The purpose of the *Archive* is to save sampled results to a persistent storage and retrieve relevant samples to become a part of population used for training a model. The *ModelFactory* is responsible for creating instances of the subclasses of the class *Model*, in our case *GpModel*, but it can be extended by other models. The *SurrogateSelector* is an auxiliary class which provides the *SurrogateManager* with additional operations like selecting individuals for reevaluation.

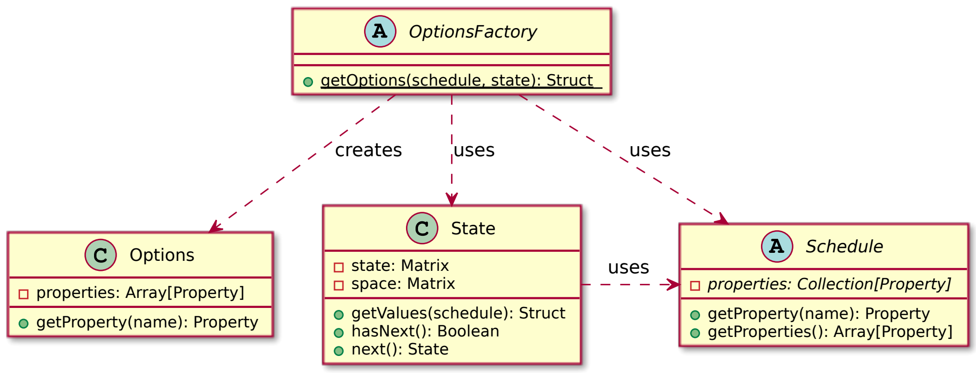

OptionsFactory
getOptions(schedule, state): Struct
creates
uses
uses
Options
properties: Array[Property]
getProperty(name): Property
State
state: Matrix
space: Matrix
getValues(schedule): Struct
hasNext(): Boolean
next(): State
uses
Schedule
properties: Collection[Property]
getProperty(name): Property
getProperties(): Array[Property]

Figure 2.5: Fragment of the class diagram – options.

2.2.4 Sequence diagram

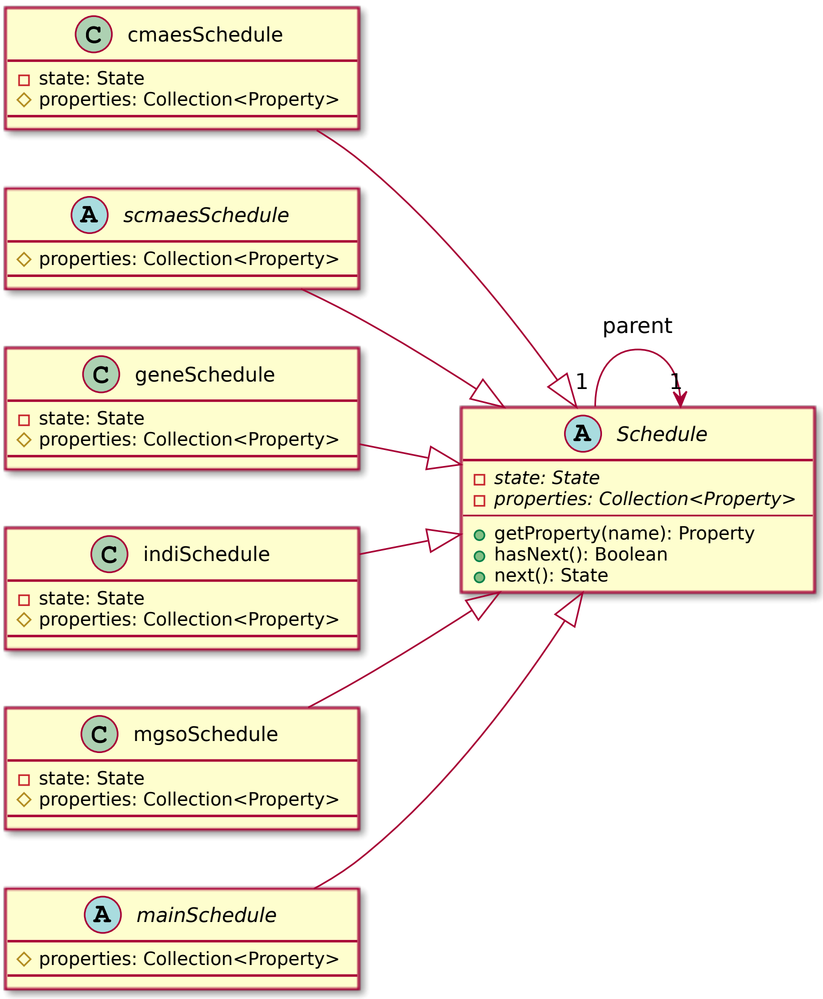
This section explains how single objects interact in the selected situations using the sequence diagrams.

2.2.4.1 Computation

As demonstrated in Figure 2.9, the *ComputationManager* runs successively all computations defined in the internal collection. When launched, each *Computation* gets the current *State* of the corresponding *Schedule* and passes both to the *OptionsFactory* in order to obtain the instance of *Options* and use it in the present computation. After the computation is finished, the result is passed to the *ResultDAO*, which saves it to a persistent storage. Finally, the subsequent state is obtained again and the whole process is repeated until the schedule is fulfilled.

2.2.4.2 Integration

Simplified diagram in Figure 2.10 demonstrates the communication of objects of the classes implementing the method integration. When the *Computation* is launched, the main control loop of the *CMAES* is executed. The algorithm behaves similarly as CMA-ES except for sampling new individuals. If some model is present, the sampling is delegated to the *SurrogateManager*, passing it EC strategy settings as an argument. The *SurrogateManager* performs sampling according to the given *EvolutionControl*, which decides when to sample new individuals using original fitness function and when to use a model instead, as explained in Section 2.1.2. Described procedure is repeated until a stopping criterion is met.



C cmaesSchedule
state: State
properties: Collection<Property>
A scmaesSchedule
properties: Collection<Property>
C geneSchedule
state: State
properties: Collection<Property>
C indiSchedule
state: State
properties: Collection<Property>
C mgsoSchedule
state: State
properties: Collection<Property>
A mainSchedule
properties: Collection<Property>
parent
1
1
A Schedule
state: State
properties: Collection<Property>
getProperty(name): Property
hasNext(): Boolean
next(): State

Figure 2.6: Fragment of the class diagram – schedules.

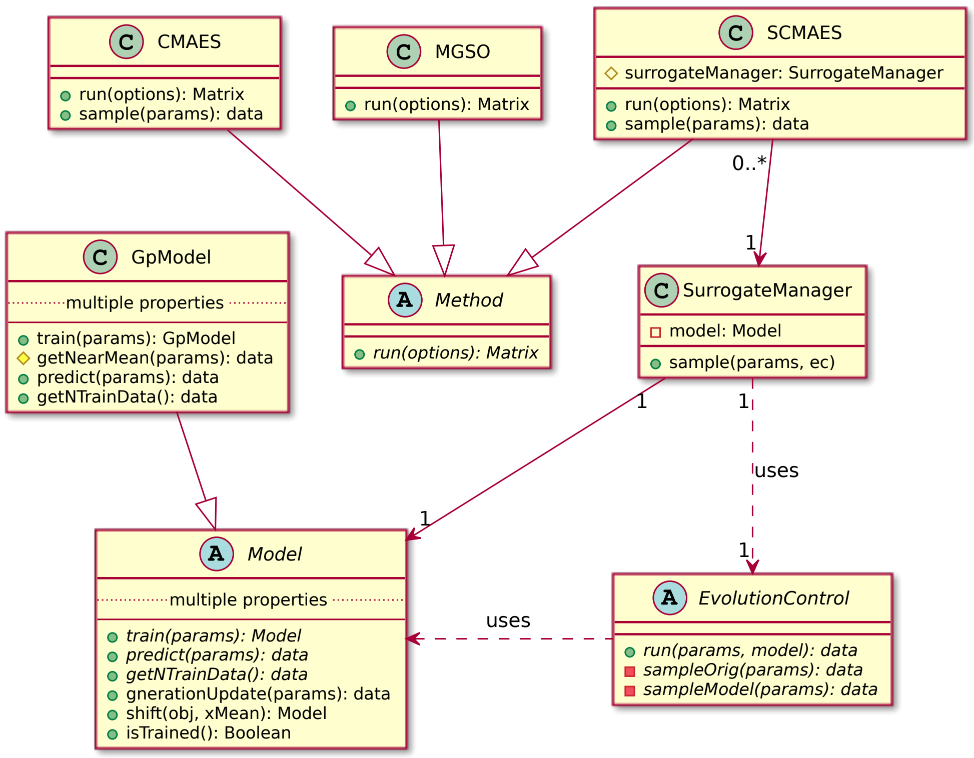


Figure 2.7: Fragment of the class diagram – methods.

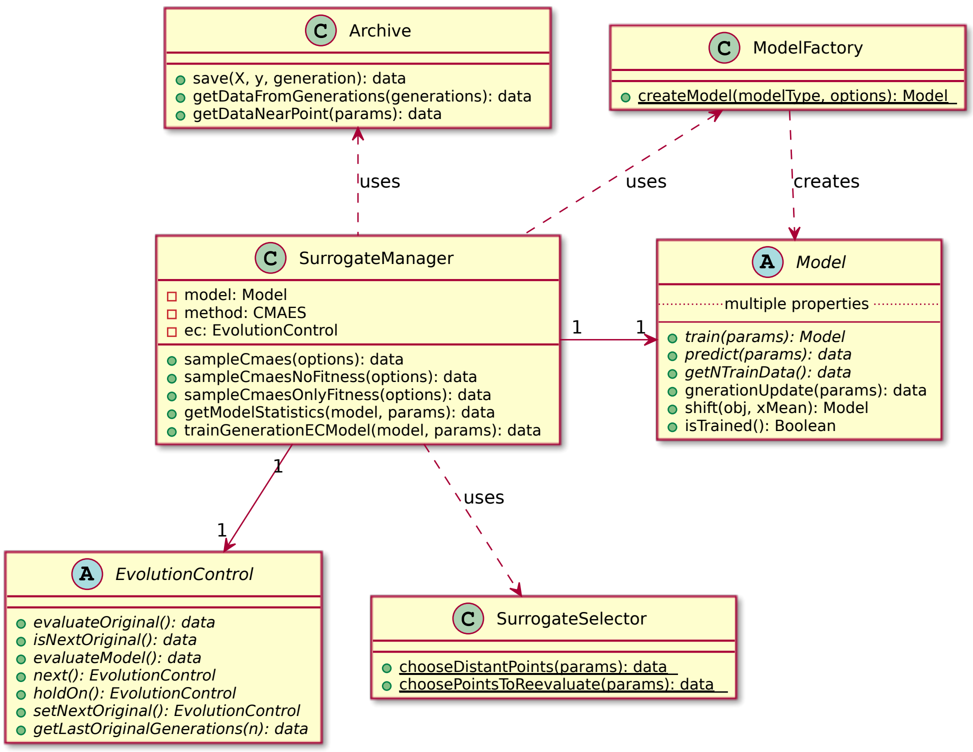

Archive
save(X, y, generation): data
getDataFromGenerations(generations): data
getDataNearPoint(params): data
ModelFactory
createModel(modelType, options): Model
uses
uses
creates
SurrogateManager
model: Model
method: CMAES
ec: EvolutionControl
sampleCmaes(options): data
sampleCmaesNoFitness(options): data
sampleCmaesOnlyFitness(options): data
getModelStatistics(model, params): data
trainGenerationECModel(model, params): data
1
1
Model
multiple properties
train(params): Model
predict(params): data
getNTrainData(): data
gnerationUpdate(params): data
shift(obj, xMean): Model
isTrained(): Boolean
1
1
uses
EvolutionControl
evaluateOriginal(): data
isNextOriginal(): data
evaluateModel(): data
next(): EvolutionControl
holdOn(): EvolutionControl
setNextOriginal(): EvolutionControl
getLastOriginalGenerations(n): data
SurrogateSelector
chooseDistantPoints(params): data
choosePointsToReevaluate(params): data

Figure 2.8: Fragment of the class diagram – method integration.

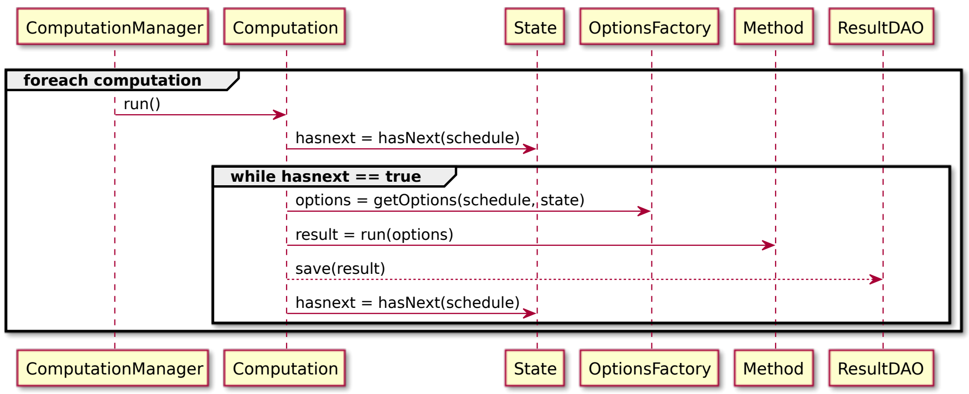


Figure 2.9: Sequence diagram – computations.

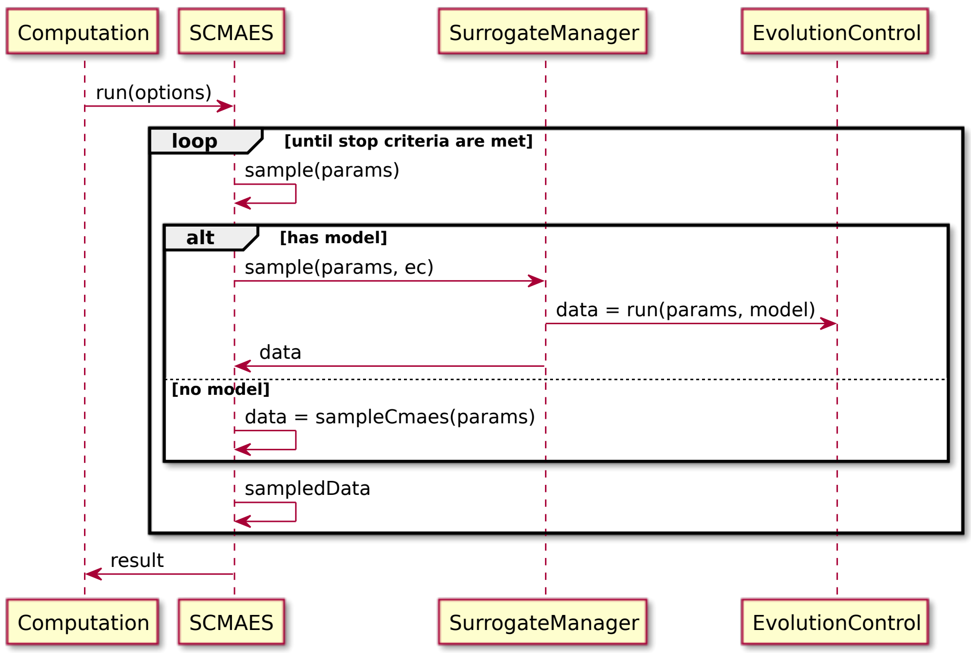


Figure 2.10: Sequence diagram – integrated methods.

Chapter 3

Realization

Section 3.1 shows some examples produced during the implementation process and Section 3.2 describes the whole process of the experimental evaluation and performance testing of the compared methods. It also compares the performance of the methods and discusses the obtained results.

3.1 Implementation

The implementation was performed using Matlab software developed by MathWorks [14] with Statistics and Machine Learning Toolbox, as it was used in the implementations of the compared methods. It provides required functionality to describe, analyze, and model data using statistics and machine learning.

Figure 3.1 shows the file structure created during implementation. Figures 3.2 and 3.3 show the implementation of the classes *SCMAES* and *Computation*, both described in Section 2.2.

3.2 Experimental Evaluation

For comparison of the performance of MGSO, CMA-ES and S-CMA-ES, the following set of 12 multi-modal fitness functions, illustrated in Figures 3.4 and 3.5, from the CEC 2013 competition [1] were used:

$f1$: **Five-Uneven-Peak Trap (1D)** – Figure 3.4a
This function has 5 local optima (i.e., peaks). However it has only two global optima.

$f2$: **Equal Maxima (1D)** – Figure 3.4b
This function has 5 global optima in the examined range, all symmetricaly located.

$f3$: **Uneven Decreasing Maxima (1D)** – Figure 3.4c
This function has only 1 global optimum and 4 local optima in the examined range.

$f4$: **Inverted Himmelblau (2D)** – Figure 3.4d
This is an inverted version of Himmelblau function. It has 4 global optima with 2 closer to each other than the other 2. There are no local optima in this function.

$f5$: **Six-Hump Camel Back (2D)** – Figure 3.4e
The function has 2 global optima as well as 2 local optima.

$f6$: **Inverted Shubert (2D, 3D)** – Figure 3.4f
This function is an inverted version of the Shubert function, where there are $n * 3^D$ global optima unevenly distributed. These global optima are divided into 3^D groups, with each group having D global optima being close to each other. In the case of 2D version, there are 18 global optima divided into 9 pairs, with optima very close to each other in each pair. By contrast, the distance between any pair is relatively greater. There are in total 760 global and local optima.

$f7$: **Vincent (2D, 3D)** – Figure 3.5a
This is an inverted version of the Vincent function. It has 6^D global optima, but in contrast to the evenly distributed global optima in $f6$, in this function the global optima have noticeably different distances between them. Moreover, there are no local optima in the Vincent function.

```
|
|__ CEC2013 ................... the benchmark from CEC 2013 competition
|__ src ......................................the directory of source codes
|   |__ computations ................................computation classes
|   |__ dao ...........................................data access classes
|   |__ main .......................................main interface classes
|   |__ methods ...........................................method classes
|   |__ processing ..............................result processing classes
|   |__ schedules .........................................schedule classes
|   |__ util ..........................additional classes, result export, etc.
|__ gpeda ...................................GPEDA component sources
|__ results .......................the results from performed experiments
|__ S-CMA-ES ............................S-CMA-ES component sources
|__ startup.m ...the startup script setting paths to all required components
```

Figure 3.1: Directory structure.

```
classdef SCMAES < Method
%% Class representing S-CMA-ES

 methods
  function res = run(this, options)
   %% Runs the optimization process and returns its
      progress as matrix of values [number of
      evaluations, distance to optimum]
   cmOptions = options.cmOptions;
   x = options.x;
   func = options.func;
   sOptions = options.sOptions;
   sigma = options.sigma;


   [xmin, f_min, counteval, stopflag, out, bestever,
      y_eval] = s_cmaes(func, x, sigma, cmOptions, '
      surrogateOptions', sOptions);

   % Subtract optimum value from reached value to obtain
      its delta
   y_eval(:, 1) = y_eval(:, 1) - options.optima;

   res = y_eval(:, [1 2]);
  end
 end
end
```

Figure 3.2: The example of the implementation of the class *SCMAES*.

$f8$: **Modified Rastrigin - All Global Optima (2D)** – Figure 3.5b
This is a modified Rastrigin function. The global optima in this function are evenly distributed.

$f9$: **Composition Function 1 (2D)** – Figure 3.5c
This function has eight global optima in the optimization box, is multimodal, shifted, non-rotated, non-symmetric, separable near the global optima, scalable and it is constructed using eight basic functions:

- $f1$ - $f2$: Griewank's function,
- $f3$ - $f4$: Weierstrass function,
- $f5$ - $f6$: Sphere function.

```
classdef Computation < handle
%% Class representing a computation

 properties (Access = private)
  schedule %% determines all parameter combinations to
     use for optimizations
  state %% holds the current state of the computation,
     the combination of the parameters
  method %% optimization method to evaluate
 end

 methods
  function this = Computation(schedule, method)
   %% Class contructor
   this.schedule = schedule;
   this.method = method;

       this.state = State(schedule);
  end

  function run(this)
   %% Runs the all optimization processes of given
      method corresponding schedule
   while this.state.hasNext()
    disp(this.state.state)
    options = OptionsFactory.getOptions(this.schedule,
       this.state);
    result = this.method.run(options);
    ResultDAO.save(result);
    this.state = this.state.next();
   end
  end
 end
end
```

Figure 3.3: The example of the implementation of the class *Computation*.

$f10$**: Composition Function 2 (2D)** – Figure 3.5d
This function has eight global optima in the optimization box, is multimodal, shifted, non-rotated, non-symmetric, separable near the global optima, scalable and is constructed using eight basic functions:

- $f1$ - $f2$: Rastrigin's function,
- $f3$ - $f4$: Weierstrass function,
- $f5$ - $f6$: Griewank's function.
- $f7$ - $f8$: Sphere function,

$f11$**: Composition Function 3 (2D, 3D, 5D, 10D)** – Figure 3.5e
This function has six global optima in the optimization box, it is multimodal, shifted, rotated, non-symmetric, non-separable, scalable and it is constructed using six basic functions:

- $f1$ - $f2$: EF8F2 function,
- $f3$ - $f4$: Weierstrass function,
- $f5$ - $f6$: Griewank's function.

$f12$**: Composition Function 4 (3D, 5D, 10D, 20D)** – Figure 3.5f
This function has eight global optima in the optimization box, is multimodal, shifted, rotated, non-symmetric, non-separable, scalable and it is constructed using eight basic functions:

- $f1$ - $f2$: Rastrigin's function,
- $f3$ - $f4$: EF8F2 function,
- $f5$ - $f6$: Weierstrass function,
- $f7$ - $f8$: Griewank's function.

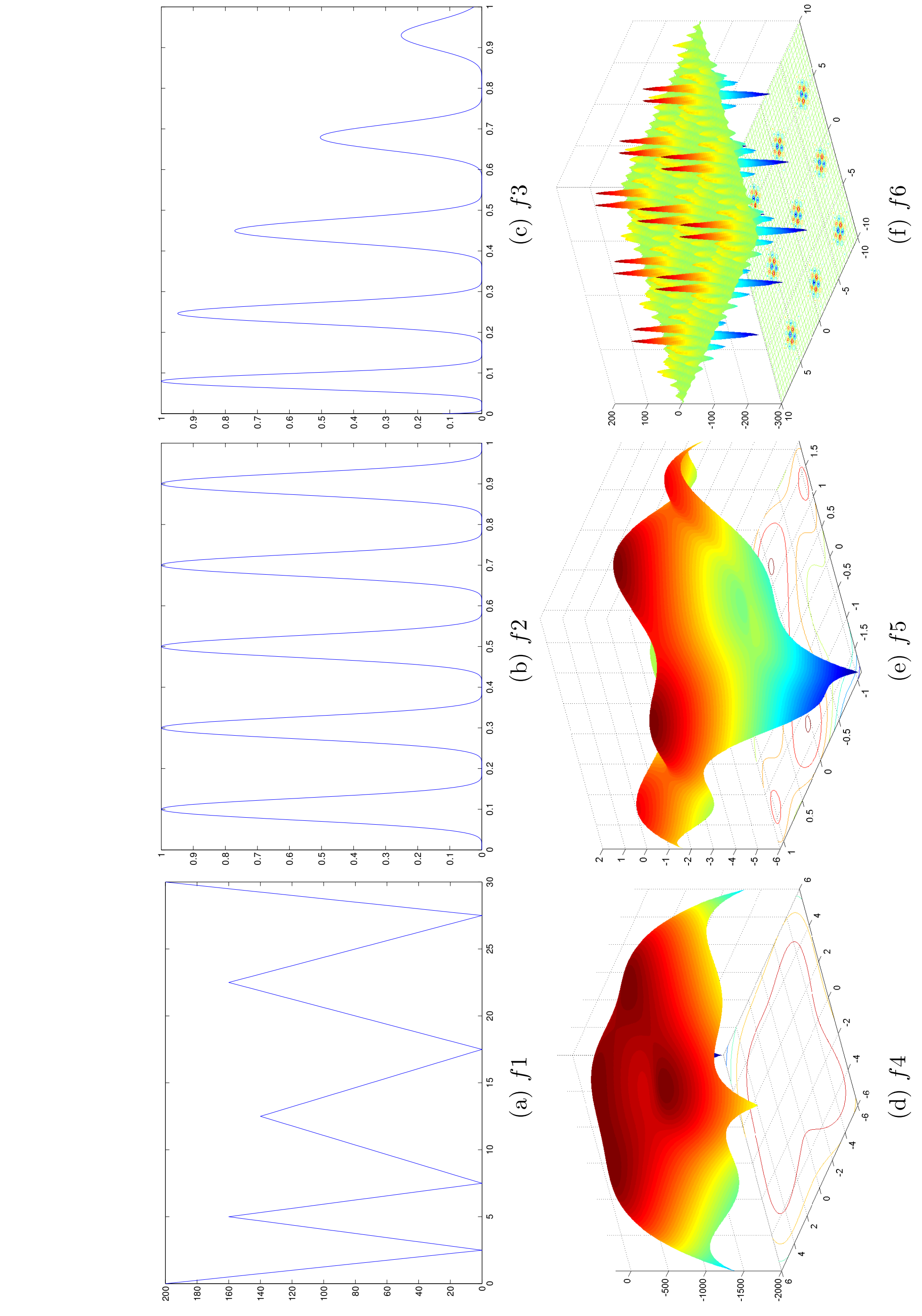
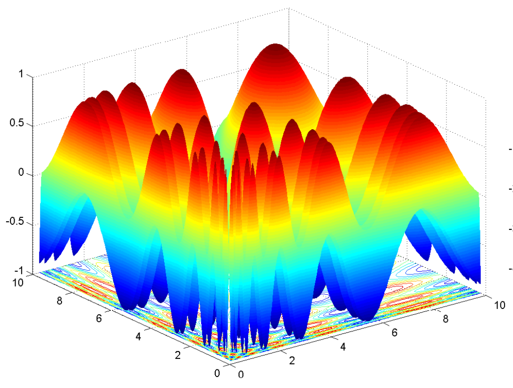
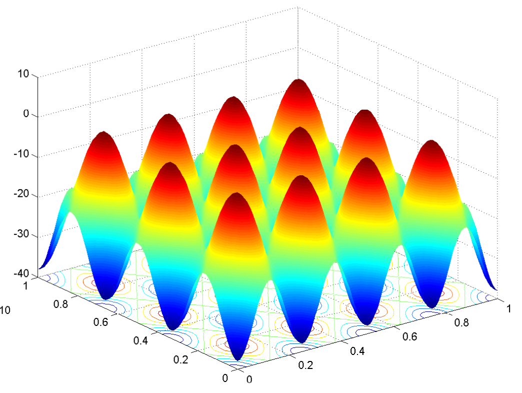
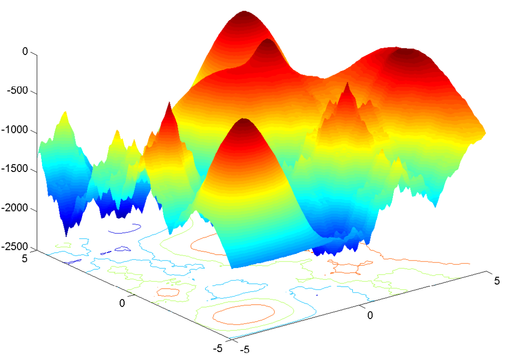
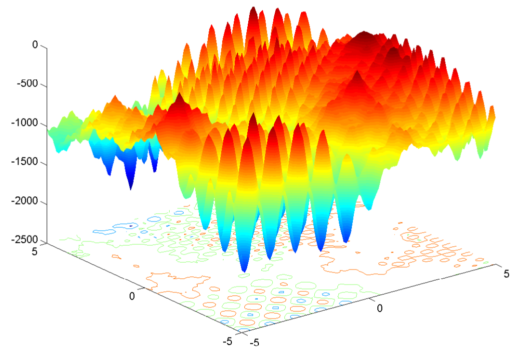


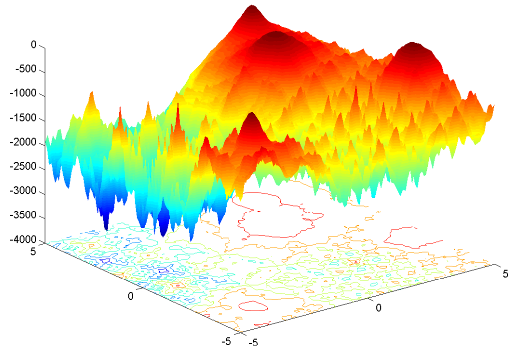
Figure 3.4: Benchmark functions $f1 - f6$ (all figures taken from [1]).


(a) $f7$


(b) $f8$


(c) $f9$


(d) $f10$


(e) $f11$

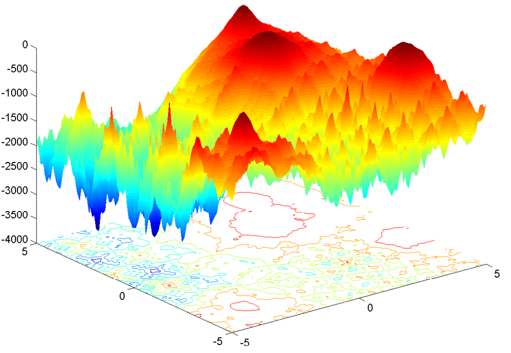

(f) $f12$

Figure 3.5: Benchmark functions $f7 - f12$ (all figures taken from [1]).

func.	variable ranges	# global optima	# local optima
$f1$	$x \in [0, 30]$	2	3
$f2$	$x \in [0, 1]$	5	0
$f3$	$x \in [0, 1]$	1	4
$f4$	$x, y \in [-6, 6]$	4	0
$f5$	$x \in [-1.9, 1.9], y \in [-1.1, 1.1]$	2	2
$f6$	$x_i \in [-10, 10]^D, i \in 1, \dots, D$	$D * 3^D$	many
$f7$	$x_i \in [0.25, 10]^D, i \in 1, \dots, D$	6^D	0
$f8$	$x_i \in [0, 1]^D, i \in 1, \dots, D$	$\sum_{i=1}^D k_i$ (note[1])	0
$f9$	$x_i \in [-5, 5]^D, i \in 1, \dots, D$	6	many
$f10$	$x_i \in [-5, 5]^D, i \in 1, \dots, D$	8	many
$f11$	$x_i \in [-5, 5]^D, i \in 1, \dots, D$	6	many
$f12$	$x_i \in [-5, 5]^D, i \in 1, \dots, D$	8	many

Table 3.1: Parameters of the tested functions: variable ranges, number of global and local optima.

All numbers of local and global optima and variable ranges can be found in Table 3.1.

In the case of some high-dimensional functions one optimization process can take a long time, and due to the large number of required evaluations, most of them were performed on the computation servers provided by MetaCentrum – Virtual Organization.

3.2.1 MGSO Performance

MGSO performance was examined using two covariance functions with parameters shown in Table 3.2. The results in Tables 3.3, 3.4, 3.5 and 3.6 show the speed-up of MGSO with respect to CMA-ES. As can be seen, the $\mathbf{K}_{\mathrm{SE}}^{\mathrm{iso}}$ covariance function performed better among these two for more than half of cases.

The highest speed-up can be seen in the case of $f2$, $f4$, 3D version of $f7$, 2D version of $f11$ and 5D $f12$. The worst results were observed in the case of $f1$, 2D version of $f10$ and 3D and 10D versions of $f12$. Figures 3.6, 3.7, 3.8, 3.9 and 3.10 show optimization progress of the respective functions.

[1]$k_i = 1$, for $i = 1 - 3, 5 - 7, 9 - 11, 13 - 15$, and $k_4 = 2$, $k_8 = 2$, $k_{12} = 3$, $k_{16} = 4$

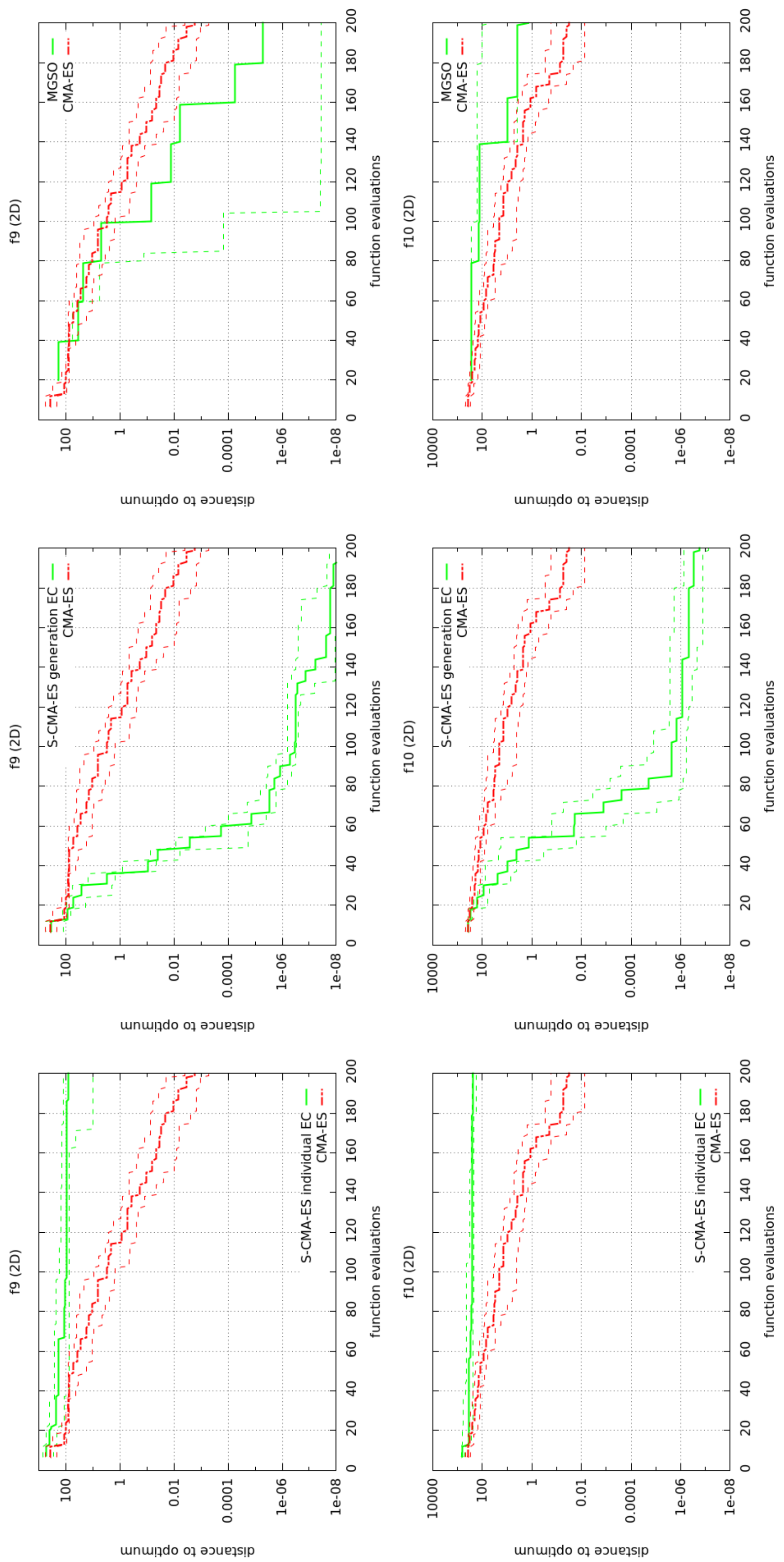


Figure 3.6: Examples of the best-fitness progress – functions $f9$ (2D) and $f10$ (2D) (see Section 3.2.3.1 for details).

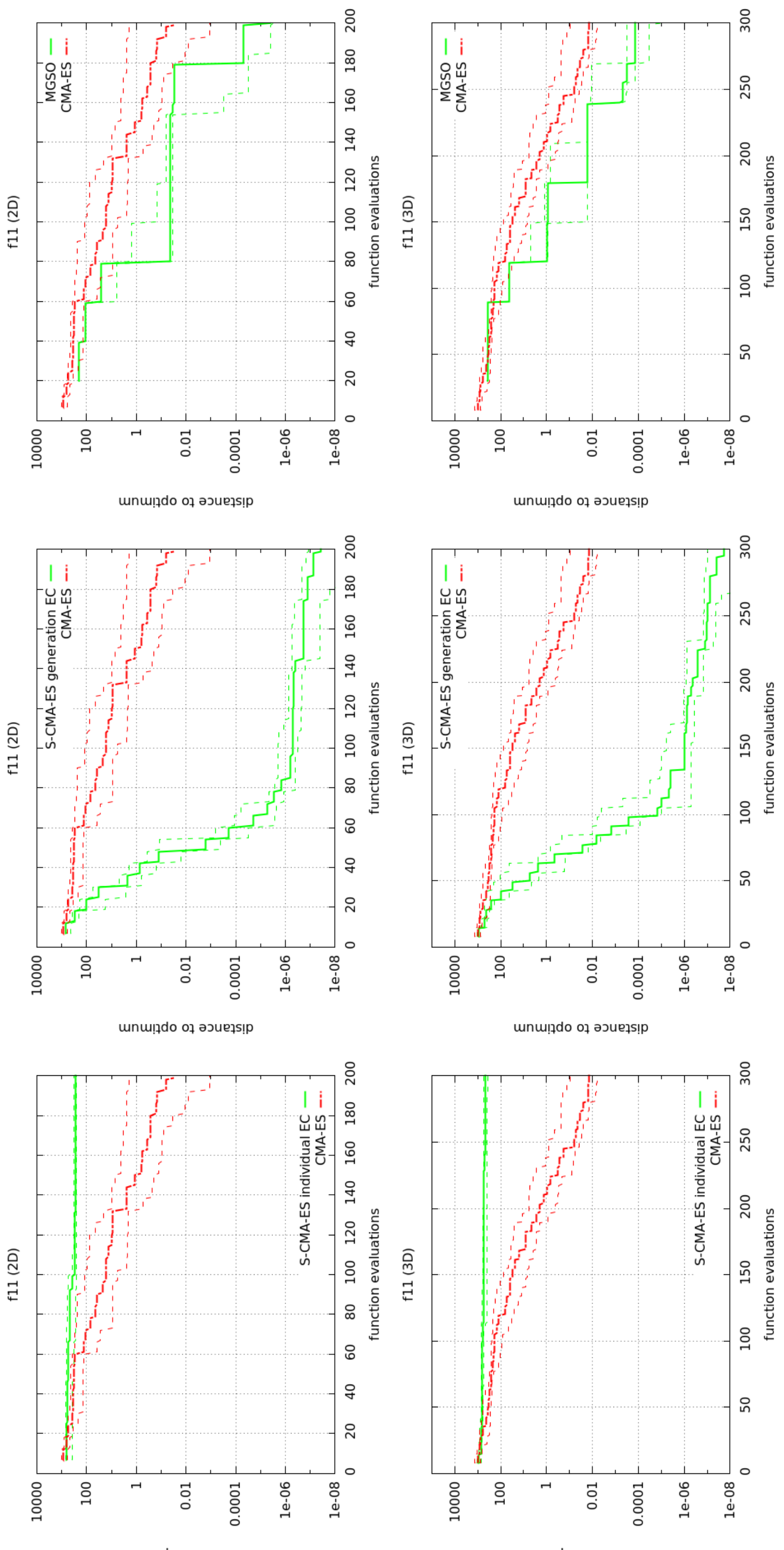


Figure 3.7: Examples of the best-fitness progress – functions $f11$ (2D) and (3D) (see Section 3.2.3.1 for details).

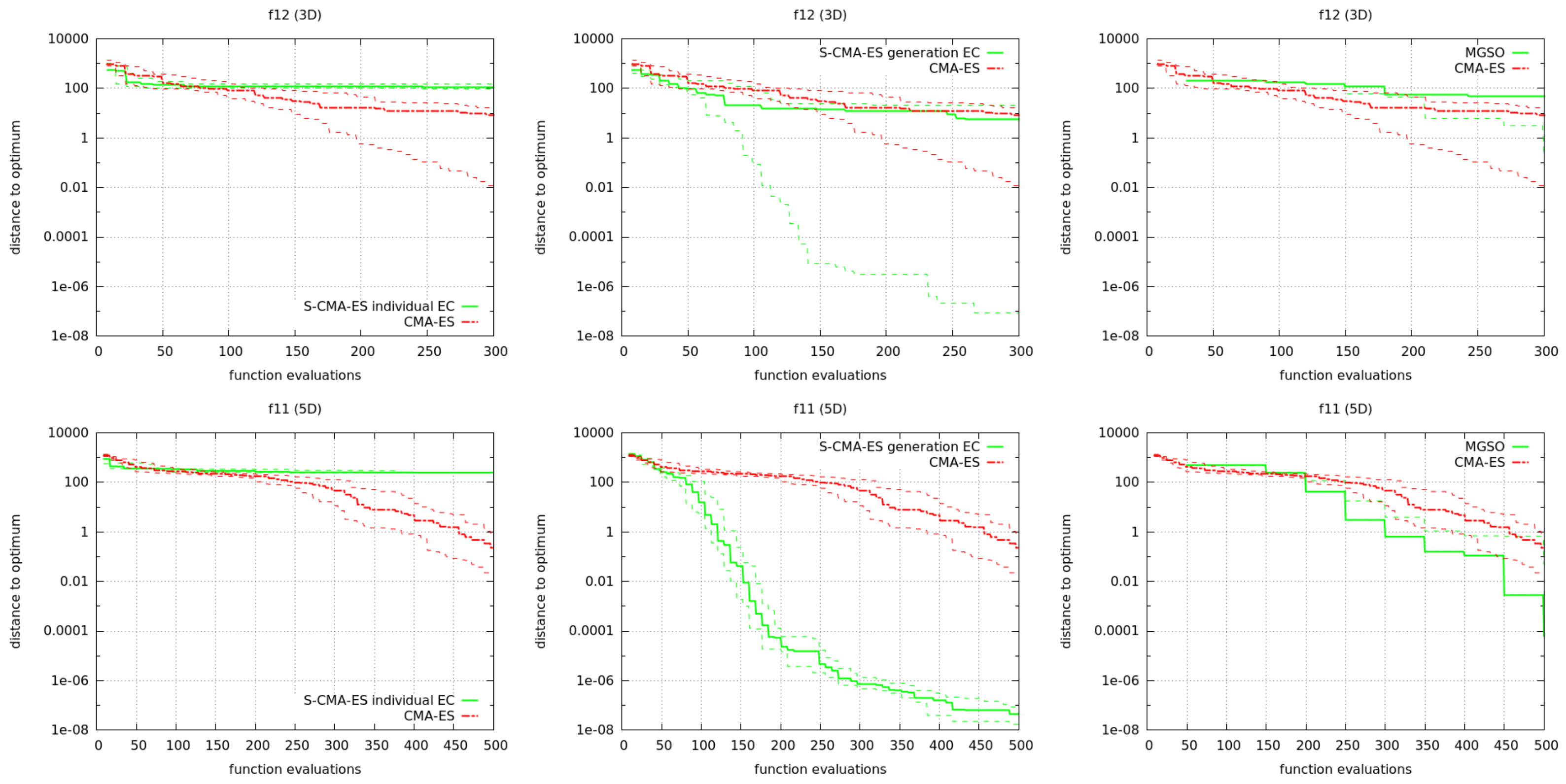


Figure 3.8: Examples of the best-fitness progress – functions $f11$ (5D) and $f12$ (3D) (see Section 3.2.3.1 for details).

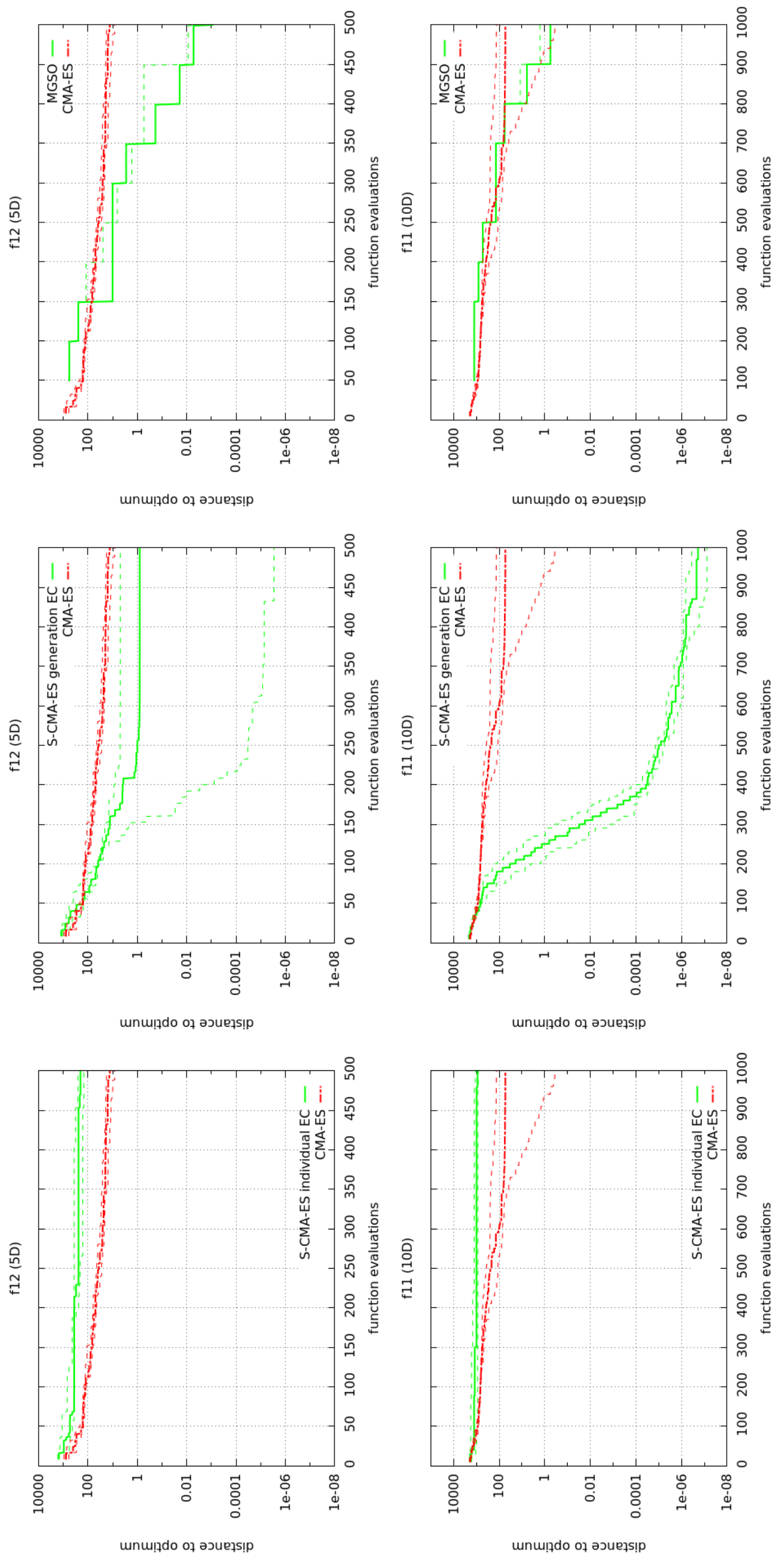


Figure 3.9: Examples of the best-fitness progress – functions f11 (10D) and f12 (5D) (see Section 3.2.3.1 for details).

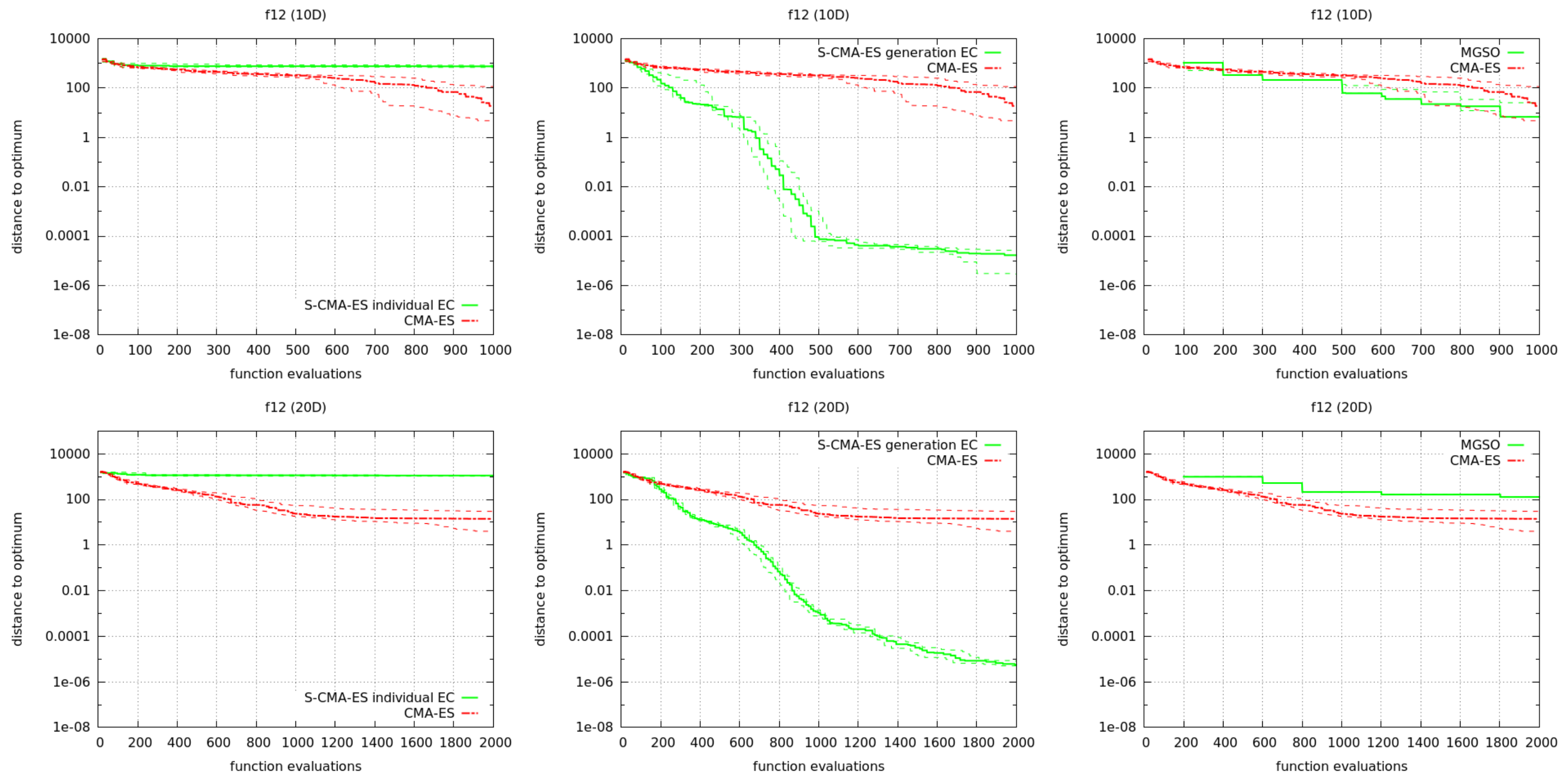


Figure 3.10: Examples of the best-fitness progress – functions f12 (10D) and f12 (20D) (see Section 3.2.3.1 for details).

S-CMA-ES	
covariance functions	$cov \in \{\mathbf{K}_{\text{Matérn}}^{\nu=\frac{5}{2}}, \mathbf{K}_{\text{exp}}, \mathbf{K}_{\text{SE}}^{\text{iso}}, \mathbf{K}_{\text{SE}}^{\text{ard}}\}$
starting values of (σ_f^2, ℓ)	$(0.1, 10)$ for $\mathbf{K}_{\text{SE}}^{\text{iso}}$ $(0.05 \times \boldsymbol{J}_{1,D}, 0.1)$ for $\mathbf{K}_{\text{SE}}^{\text{ard}}$ $(0.5, 2)$ otherwise
starting values of σ_n^2	0.01

MGSO	
covariance functions	$cov \in \{\mathbf{K}_{\text{SE}}^{\text{iso}}, \mathbf{K}_{\text{SE}}^{\text{ard}}\}$
starting values of (σ_f^2, ℓ)	$(0.1, 10)$ for $\mathbf{K}_{\text{SE}}^{\text{iso}}$ $(0.05 \times (\boldsymbol{J}_{1,D}), 0.1)$ for $\mathbf{K}_{\text{SE}}^{\text{ard}}$
starting values of σ_n^2	0.01

Table 3.2: Model parameter settings for S-CMA-ES and MGSO performance testing (see Section 3.2.3.1 for details).

3.2.2 S-CMA-ES Performance

The speed-up results are shown in Tables 3.3, 3.4, 3.5 and 3.6. In performed evaluations, four covariance functions in the GP surrogate model were used, two types of the squared exponential covariance function, the isotropic version $\mathbf{K}_{\text{SE}}^{\text{iso}}$ and the version using automatic relevance determination $\mathbf{K}_{\text{SE}}^{\text{ard}}$, and two types of the Matérn covariance function, $\mathbf{K}_{\text{Matérn}}^{\nu=\frac{1}{2}}$ ($\mathbf{K}_{\text{exp}}$), which is better known as exponential covariance function, and $\mathbf{K}_{\text{Matérn}}^{\nu=\frac{5}{2}}$, their definitions can be found in [15]. The covariance functions parameters are shown in Table 3.2.

In the performed experiments, different configurations of the chosen EC strategies, described in Section 2.1.2, were examined, *generation-based* and *individual-based*. The result are discussed in the following sections.

3.2.2.1 Generation-based EC strategy

Apart from covariance function selection, *Generation-based* EC strategy was determined by two other parameters, the number of model-evaluated generations and the multiplication factor of CMA-ES' step size $\sigma^{(g)}$. In the implementation, the first parameter was fixed on values 1, 2, 4 and 8 consequent model-evaluated generations and the second parameter was varied among the values 1 and 2.

In the case of the *generation-based* EC, the overall best settings with respect to the median values are $(\mathbf{K}_{\text{SE}}^{\text{iso}}, 8, 1)$ – 8 consequent model-evaluated generations with unmodified step size in combination with $\mathbf{K}_{\text{SE}}^{\text{iso}}$ covariance function. The overall best *generation-based* EC settings showed to be also the best *generation-based* EC settings of the respective functions, except for 3D version of $f12$, where S-CMA-ES performed better using larger step size.

Using different covariance functions didn't bring much better results than the overall best covariance function.

3.2.2.2 Individual-based EC strategy

Apart from covariance function selection, three other parameters were examined in the case of *individual-based* EC strategy. In the implementation, the first parameter $\alpha \in [0, 1]$ determines the amount of points $\lambda'' \in [0, \alpha\lambda]$, where λ is the size of the original population. Those points are sampled first and then evaluated and used for training the model. The second parameter $\beta \in \{1, \ldots, \infty\}$ is a multiplicator determining the size of extended population $\beta(\lambda - \lambda'')$ without the pre-sampled individuals. Extended population is required by the model for choosing promising points for re-evaluation by the original fitness function. The third parameter $\gamma \in [0, 1]$ determines the amount of points $\gamma(\lambda - \lambda'')$ with the best model-fitness chosen from the extended population to be re-evaluated by the original fitness function and become a part of the final population. The complement to λ points is gathered from the rest of the extended population by dividing it into $\beta(\lambda - \lambda'')(1 - \gamma)$ clusters and selecting the best point from each cluster.

In performed evaluations the parameter α was fixed on values 0, 0.0625, 0.125, and 0.25, the parameter β was varied among the values 5 and 10 and γ was fixed on values 0, 0.1 and 0.2. Achieved results show the best overall settings $(\mathbf{K}_{\mathrm{SE}}^{\mathrm{ard}}, 0, 5, 0.1)$ – $\mathbf{K}_{\mathrm{SE}}^{\mathrm{ard}}$ covariance matrix, no pre-sampling before training the model, 5 as the multiplicator determining the size of extended population and 0.1 as a multiplicator determining the amount of best points chosen from extended population.

The best results using described parameters where achieved in the case of functions $f3$ and 2D and 3D version of $f6$. However, the overall performance of the *individual-based* EC strategy lags far behind the *generation-based* EC strategy, MGSO and even CMA-ES itself. The results show that in most cases the optimization process is unable to approach the optimum as close as other methods.

3.2.2.3 Summary

In the case of *generations-based* EC strategy, employing a GP as a surrogate model demonstrated performance improvement for most tested functions. The highest speed-up was achieved on both 2D and 3D versions of $f7$, 2D and 5D versions of $f11$ and 20-D version of $f12$. Figures 3.6, 3.7, 3.8, 3.9 and 3.10 show the optimization progress using the best observed settings for the respective functions. The highest speed-up can be seen in the later phase of the optimization process.

The lowest speed-up factors were observed in the case of the functions $f1$, $f5$ and both 2D and 3D versions of the function $f6$. These functions are characterized by unevenly distributed global optima and relatively large distances between global and local optima, where the optimization process tends to get trapped.

f1 (1–D)						
Δf_{opt}	S-CMA-ES - generation EC str.		S-CMA-ES - individual EC str.		MGSO	
1e1	00.90	01.00	01.00		00.17	00.14
1e0	00.90	01.00	01.00		00.06	00.14
1e-1	00.90	01.00	01.00		00.06	00.08
1e-2	00.90	01.00	01.00		-	00.05
1e-3	00.90	01.00	01.00		-	00.05
1e-4	00.90	01.00	01.00		-	-
1e-5	00.90	01.00	01.00		-	-
1e-6	00.90	01.00	01.00		-	-
1e-7	00.90	01.00	01.00		-	-
1e-8	00.90	01.00	01.00		-	-
param:	$(\mathbf{K}^{\text{iso}}_{SE}, 8, 1)$	$(\mathbf{K}^{\text{ard}}_{\text{SE}}, 8, 1)$	$(\mathbf{K}^{\text{ard}}_{\text{SE}}, 0, 5, 0.1)$		$\mathbf{K}^{\text{iso}}_{\text{SE}}$	$\mathbf{K}^{\text{ard}}_{\text{SE}}$

f2 (1–D)						
Δf_{opt}	S-CMA-ES - generation EC str.		S-CMA-ES - individual EC str.		MGSO	
1e-1	01.00	**01.44**	00.76	00.13	01.30	
1e-2	**01.41**	**02.54**	00.30	00.10	**03.30**	
1e-3	**01.85**	**02.54**	00.12	00.86	01.08	
1e-4	**03.11**	**03.75**	-	01.73	**01.73**	
1e-5	**04.04**	**04.49**	-	-	**02.25**	
1e-6	**04.34**	**04.83**	-	-	**02.37**	
1e-7	**12.95**	**13.57**	-	-	**08.05**	
1e-8	12.21	15.23	-	-	16.78	
param:	$(\mathbf{K}^{\text{iso}}_{SE}, 8, 1)$	$(\mathbf{K}^{\text{ard}}_{\text{SE}}, 8, 1)$	$(\mathbf{K}^{\text{ard}}_{\text{SE}}, 0, 5, 0.1)$	$(\mathbf{K}^{\upsilon=\frac{5}{2}}_{\text{Matérn}}, 2^{-2}, 10, 0.2)$	$\mathbf{K}^{\text{iso}}_{\text{SE}}$	

f3 (1–D)						
Δf_{opt}	S-CMA-ES - generation EC str.		S-CMA-ES - individual EC str.		MGSO	
1e-1	**02.44**		00.52	00.45	01.83	
1e-2	05.71		01.44	01.51	02.13	
1e-3	21.36		07.12	17.80	09.49	
1e-4	18.41		07.12	-	08.63	
1e-5	20.07		07.76	-	09.41	
1e-6	20.07		07.76	-	09.41	
1e-7	*		*	*	*	
1e-8	*		*	*	*	
param:	$(\mathbf{K}^{\text{iso}}_{SE}, 8, 1)$		$(\mathbf{K}^{\text{ard}}_{\text{SE}}, 0, 5, 0.1)$	$(\mathbf{K}_{\text{exp}}, 0, 5, 0.2)$	$\mathbf{K}^{\text{iso}}_{\text{SE}}$	

f4 (2–D)						
Δf_{opt}	S-CMA-ES - generation EC str.		S-CMA-ES - individual EC str.		MGSO	
1e2	00.54	01.00	00.16	00.11	00.35	00.35
1e1	01.24	**01.63**	00.17	00.44	**01.55**	01.55
1e0	**02.59**	**03.52**	-	01.26	**02.20**	04.00
1e-1	**02.74**	**03.47**	-	-	**01.97**	02.62
1e-2	**02.99**	**03.66**	-	-	**02.44**	03.25
1e-3	**03.91**	**05.00**	-	-	**03.58**	04.57
1e-4	**04.45**	**05.22**	-	-	**04.52**	04.68
1e-5	**13.44**	**14.83**	-	-	00.67	13.23
1e-6	**17.63**	**20.10**	-	-	-	18.12
1e-7	+	+	*	*	*	+
1e-8	+	+	*	*	*	+
param:	$(\mathbf{K}^{\text{iso}}_{SE}, 8, 1)$	$(\mathbf{K}_{\text{exp}}, 8, 1)$	$(\mathbf{K}^{\text{ard}}_{\text{SE}}, 0, 5, 0.1)$	$(\mathbf{K}_{\text{exp}}, 2^{-2}, 5, 0.1)$	$\mathbf{K}^{\text{iso}}_{\text{SE}}$	$\mathbf{K}^{\text{ard}}_{\text{SE}}$

f5 (2–D)						
Δf_{opt}	S-CMA-ES - generation EC str.		S-CMA-ES - individual EC str.		MGSO	
1e0	01.00		00.12	00.09	00.65	00.65
1e-1	**01.72**		00.05	00.08	00.80	00.80
1e-2	**03.28**		-	00.70	**01.52**	01.52
1e-3	**02.95**		-	-	**01.51**	01.92
1e-4	**03.27**		-	-	**01.96**	02.48
1e-5	**03.74**		-	-	**02.55**	03.23
1e-6	*		*	*	*	*
1e-7	*		*	*	*	*
1e-8	*		*	*	*	*
param:	$(\mathbf{K}^{\text{iso}}_{SE}, 8, 1)$		$(\mathbf{K}^{\text{ard}}_{\text{SE}}, 0, 5, 0.1)$	$(\mathbf{K}^{\upsilon=\frac{5}{2}}_{\text{Matérn}}, 0, 10, 0.2)$	$\mathbf{K}^{\text{iso}}_{\text{SE}}$	$\mathbf{K}^{\text{ard}}_{\text{SE}}$

f6 (2–D)						
Δf_{opt}	S-CMA-ES - generation EC str.		S-CMA-ES - individual EC str.		MGSO	
1e2	**01.18**	01.18	-	01.74	**03.04**	02.03
1e1	04.97	04.20	-	-	00.36	02.21
1e0	07.16	08.40	-	-	-	01.80
1e-1	+	+	*	*	*	+
1e-2	+	+	*	*	*	+
1e-3	+	+	*	*	*	+
1e-4	+	+	*	*	*	+
1e-5	*	*	*	*	*	*
1e-6	*	*	*	*	*	*
1e-7	*	*	*	*	*	*
1e-8	*	*	*	*	*	*
param:	$(\mathbf{K}^{\text{iso}}_{SE}, 8, 1)$	$(\mathbf{K}^{\text{ard}}_{\text{SE}}, 4, 1)$	$(\mathbf{K}^{\text{ard}}_{\text{SE}}, 0, 5, 0.1)$	$(\mathbf{K}^{\text{ard}}_{\text{SE}}, 0, 10, 0)$	$\mathbf{K}^{\text{iso}}_{\text{SE}}$	$\mathbf{K}^{\text{ard}}_{\text{SE}}$

Table 3.3: Speed-up of S-CMA-ES using *individual*- and *generation-based* EC strategies and MGSO, compared to CMA-ES without a surrogate model – functions $f1 - f6$ (see Section 3.2.3.3 for details).

f7 (2–D)						
Δf_{opt}	S-CMA-ES - generation EC str.		S-CMA-ES - individual EC str.		MGSO	
1e-2	**03.28**	**02.93**	00.05	00.43	00.14	01.02
1e-3	**04.42**	**04.81**	-	-	00.27	00.54
1e-4	**05.63**	**06.09**	-	-	-	00.52
1e-5	**07.91**	**07.91**	-	-	-	01.45
1e-6	**16.24**	**16.24**	-	-	-	03.39
1e-7	65.25	68.62	-	-	-	11.29
1e-8	+	+	*	*	*	+
param:	($\mathbf{K}^{\text{iso}}_{SE}$, 8, 1)	($\mathbf{K}_{\text{exp}}$, 8, 1)	($\mathbf{K}^{\text{ard}}_{\text{SE}}$, 0, 5, 0.1)	($\mathbf{K}^{\text{ard}}_{\text{SE}}$, 2^{-4}, 10, 0)	$\mathbf{K}^{\text{iso}}_{\text{SE}}$	$\mathbf{K}^{\text{ard}}_{\text{SE}}$

f6 (3–D)						
Δf_{opt}	S-CMA-ES - generation EC str.		S-CMA-ES - individual EC str.		MGSO	
1e4	01.00		01.00	01.00	00.27	
1e3	17.55		-	07.42	04.53	
1e2	07.72		-	-	04.06	
1e1	05.96		-	-	-	
1e0	+		*	*	*	
1e-1	+		*	*	*	
1e-2	+		*	*	*	
1e-3	+		*	*	*	
1e-4	+		*	*	*	
1e-5	+		*	*	*	
1e-6	+		*	*	*	
1e-7	+		*	*	*	
param:	($\mathbf{K}^{\text{iso}}_{SE}$, 8, 1)		($\mathbf{K}^{\text{ard}}_{\text{SE}}$, 0, 5, 0.1)	($\mathbf{K}^{\text{ard}}_{\text{SE}}$, 0, 10, 0.1)	$\mathbf{K}^{\text{iso}}_{\text{SE}}$	

f7 (3–D)						
Δf_{opt}	S-CMA-ES - generation EC str.		S-CMA-ES - individual EC str.		MGSO	
1e-1	**03.15**		00.03	01.28	01.14	
1e-2	**05.26**		-	-	**01.56**	
1e-3	**10.36**		-	-	**03.16**	
1e-4	**13.33**		-	-	**04.95**	
1e-5	64.47		-	-	22.83	
1e-6	+		*	*	+	
1e-7	+		*	*	+	
1e-8	+		*	*	+	
param:	($\mathbf{K}^{\text{iso}}_{SE}$, 8, 1)		($\mathbf{K}^{\text{ard}}_{\text{SE}}$, 0, 5, 0.1)	($\mathbf{K}^{v=\frac{5}{2}}_{\text{Matérn}}$, 0, 10, 0)	$\mathbf{K}^{\text{iso}}_{\text{SE}}$	

f8 (2–D)						
Δf_{opt}	S-CMA-ES - generation EC str.		S-CMA-ES - individual EC str.		MGSO	
1e0	**02.68**	**02.68**	-	00.96	**01.68**	01.68
1e-1	**03.91**	**03.91**	-	-	**02.22**	02.22
1e-2	**04.46**	**04.79**	-	-	**03.20**	03.20
1e-3	**07.15**	**07.65**	-	-	**03.29**	03.29
1e-4	**12.95**	**13.74**	-	-	**05.61**	06.73
1e-5	**22.40**	**24.85**	-	-	**11.39**	13.53
1e-6	+	+	*	*	+	+
1e-7	+	+	*	*	+	+
1e-8	+	+	*	*	+	+
param:	($\mathbf{K}^{\text{iso}}_{SE}$, 8, 1)	($\mathbf{K}^{v=\frac{5}{2}}_{\text{Matérn}}$, 8, 1)	($\mathbf{K}^{\text{ard}}_{\text{SE}}$, 0, 5, 0.1)	($\mathbf{K}_{\text{exp}}$, 2^{-4}, 5, 0)	$\mathbf{K}^{\text{iso}}_{\text{SE}}$	$\mathbf{K}^{\text{ard}}_{\text{SE}}$

f9 (2–D)						
Δf_{opt}	S-CMA-ES - generation EC str.		S-CMA-ES - individual EC str.		MGSO	
1e1	**03.40**	**02.74**	-	00.10	01.06	
1e0	**03.38**	**03.11**	-	00.07	**01.15**	
1e-1	**03.82**	**03.82**	-	-	**01.53**	
1e-2	**04.35**	**04.35**	-	-	**01.52**	
1e-3	**07.80**	**07.80**	-	-	**02.68**	
1e-4	**22.40**	**23.56**	-	-	**08.54**	
1e-5	+	+	*	*	+	
1e-6	+	+	*	*	+	
1e-7	+	+	*	*	+	
1e-8	+	+	*	*	*	
param:	($\mathbf{K}^{\text{iso}}_{SE}$, 8, 1)	($\mathbf{K}^{\text{ard}}_{\text{SE}}$, 8, 1)	($\mathbf{K}^{\text{ard}}_{\text{SE}}$, 0, 5, 0.1)	($\mathbf{K}^{\text{ard}}_{\text{SE}}$, 2^{-4}, 5, 0)	$\mathbf{K}^{\text{iso}}_{\text{SE}}$	

f10 (2–D)						
Δf_{opt}	S-CMA-ES - generation EC str.		S-CMA-ES - individual EC str.		MGSO	
1e2	**01.98**	**01.76**	00.03	00.26	00.39	
1e1	**02.29**	**02.62**	-	-	00.46	
1e0	**03.01**	**03.21**	-	-	00.07	
1e-1	**03.71**	**04.00**	-	-	-	
1e-2	**06.15**	**07.02**	-	-	-	
1e-3	18.28	21.88	-	-	-	
1e-4	+	+	*	*	*	
1e-5	+	+	*	*	*	
1e-6	+	+	*	*	*	
1e-7	+	+	*	*	*	
1e-8	+	+	*	*	*	
param:	($\mathbf{K}^{\text{iso}}_{SE}$, 8, 1)	($\mathbf{K}^{v=\frac{5}{2}}_{\text{Matérn}}$, 8, 1)	($\mathbf{K}^{\text{ard}}_{\text{SE}}$, 0, 5, 0.1)	($\mathbf{K}^{\text{ard}}_{\text{SE}}$, 0, 10, 0)	$\mathbf{K}^{\text{iso}}_{\text{SE}}$	

Table 3.4: Speed-up of S-CMA-ES using *individual*- and *generation-based* EC strategies and MGSO, compared to CMA-ES without a surrogate model – functions $f7 - f12$ (see Section 3.2.3.3 for details).

f11 (2–D)					
Δf_{opt}	S-CMA-ES - generation EC str.		S-CMA-ES - individual EC str.		MGSO
1e2	**03.92**	**03.38**	00.04	01.06	**01.24**
1e1	**03.69**	**03.69**	-	-	**01.43**
1e0	**04.63**	**05.01**	-	-	**02.32**
1e-1	**05.19**	**05.19**	-	-	**03.18**
1e-2	**08.70**	**09.23**	-	-	**02.51**
1e-3	**12.06**	**14.04**	-	-	**03.72**
1e-4	59.40	65.25	-	-	18.46
1e-5	+	+	*	*	+
1e-6	+	+	*	*	*
1e-7	+	+	*	*	*
1e-8	+	+	*	*	*
param:	$(\mathbf{K}_{SE}^{\text{iso}}, 8, 1)$	$(\mathbf{K}_{\text{SE}}^{\text{ard}}, 8, 1)$	$(\mathbf{K}_{\text{SE}}^{\text{ard}}, 0, 5, 0.1)$	$(\mathbf{K}_{SE}^{\text{iso}}, 2^{-4}, 10, 0)$	$\mathbf{K}_{\text{SE}}^{\text{iso}}$

f11 (3–D)					
Δf_{opt}	S-CMA-ES - generation EC str.		S-CMA-ES - individual EC str.		MGSO
1e3	00.53	01.00	01.00	01.00	00.27
1e2	**02.51**	**03.04**	-	-	**01.33**
1e1	**03.04**	**03.38**	-	-	**01.41**
1e0	**03.12**	**03.47**	-	-	**01.85**
1e-1	**03.90**	**04.33**	-	-	**01.71**
1e-2	**05.95**	**07.21**	-	-	**02.19**
1e-3	27.83	32.15	-	-	11.06
1e-4	+	+	*	*	+
1e-5	+	+	*	*	+
1e-6	+	+	*	*	+
1e-7	+	+	*	*	*
1e-8	+	+	*	*	*
param:	$(\mathbf{K}_{SE}^{\text{iso}}, 8, 1)$	$(\mathbf{K}_{\text{exp}}, 8, 1)$	$(\mathbf{K}_{\text{SE}}^{\text{ard}}, 0, 5, 0.1)$	$(\mathbf{K}_{\text{SE}}^{\text{ard}}, 0, 5, 0)$	$\mathbf{K}_{\text{SE}}^{\text{iso}}$

f12 (3–D)						
Δf_{opt}	S-CMA-ES - generation EC str.		S-CMA-ES - individual EC str.		MGSO	
1e2	**02.05**	01.72	00.37	00.27	00.91	00.46
1e1	**01.42**	02.27	-	00.67	00.31	00.63
1e0	-	**02.28**	-	-	00.48	00.65
1e-1	-	**02.76**	-	-	-	-
1e-2	-	**03.78**	-	-	-	-
1e-3	-	19.56	-	-	-	-
1e-4	*	+	*	*	*	*
1e-5	*	+	*	*	*	*
1e-6	*	+	*	*	*	*
1e-7	*	+	*	*	*	*
1e-8	*	+	*	*	*	*
param:	$(\mathbf{K}_{SE}^{\text{iso}}, 8, 1)$	$(\mathbf{K}_{\text{SE}}^{\text{ard}}, 8, 2)$	$(\mathbf{K}_{\text{SE}}^{\text{ard}}, 0, 5, 0.1)$	$(\mathbf{K}_{\text{Matérn}}^{v=\frac{5}{2}}, 0, 10, 0.1)$	$\mathbf{K}_{\text{SE}}^{\text{iso}}$	$\mathbf{K}_{\text{SE}}^{\text{ard}}$

f11 (5–D)			
Δf_{opt}	S-CMA-ES - generation EC str.	S-CMA-ES - individual EC str.	MGSO
1e2	**03.02**	-	**01.36**
1e1	**03.24**	-	**01.51**
1e0	**04.37**	-	**01.96**
1e-1	**09.79**	-	03.12
1e-2	27.12	-	07.51
1e-3	47.82	-	13.17
1e-4	+	*	+
1e-5	+	*	*
1e-6	+	*	*
1e-7	+	*	*
1e-8	+	*	*
param:	$(\mathbf{K}_{SE}^{\text{iso}}, 8, 1)$	$(\mathbf{K}_{\text{SE}}^{\text{ard}}, 0, 5, 0.1)$	$\mathbf{K}_{\text{SE}}^{\text{iso}}$

Table 3.5: Speed-up of S-CMA-ES using *individual*- and *generation-based* EC strategies and MGSO, compared to CMA-ES without a surrogate model – functions $f13 - f16$ (see Section 3.2.3.3 for details).

f12 (5–D)					
Δf_{opt}	S-CMA-ES - generation EC str.		S-CMA-ES - individual EC str.		MGSO
1e2	**01.64**	**01.70**	-	00.04	00.78
1e1	**05.60**	**06.61**	-	-	**04.73**
1e0	19.24	33.01	-	-	28.86
1e-1	+	+	*	*	+
1e-2	+	+	*	*	+
1e-3	+	+	*	*	+
1e-4	+	+	*	*	*
1e-5	+	+	*	*	*
1e-6	+	+	*	*	*
1e-7	+	*	*	*	*
1e-8	*	*	*	*	*
param:	$(\mathbf{K}_{SE}^{\text{iso}}, 8, 1)$	$(\mathbf{K}_{\text{SE}}^{\text{ard}}, 4, 1)$	$(\mathbf{K}_{\text{SE}}^{\text{ard}}, 0, 5, 0.1)$	$(\mathbf{K}_{SE}^{\text{iso}}, 2^{-2}, 5, 0)$	$\mathbf{K}_{\text{SE}}^{\text{iso}}$

f11 (10–D)					
Δf_{opt}	S-CMA-ES - generation EC str.		S-CMA-ES - individual EC str.		MGSO
1e3	00.89	01.14	00.17	00.64	00.27
1e2	**03.65**	**04.11**	-	-	01.05
1e1	**08.51**	**09.35**	-	-	02.61
1e0	**09.78**	**10.59**	-	-	02.10
1e-1	**20.02**	**21.50**	-	-	-
1e-2	+	+	*	*	*
1e-3	+	+	*	*	*
1e-4	+	+	*	*	*
1e-5	+	+	*	*	*
1e-6	+	+	*	*	*
1e-7	*	+	*	*	*
1e-8	*	*	*	*	*
param:	$(\mathbf{K}_{SE}^{\text{iso}}, 8, 1)$	$(\mathbf{K}_{\text{Matérn}}^{v=\frac{5}{2}}, 8, 1)$	$(\mathbf{K}_{\text{SE}}^{\text{ard}}, 0, 5, 0.1)$	$(\mathbf{K}_{SE}^{\text{iso}}, 2^{-4}, 5, 0)$	$\mathbf{K}_{\text{SE}}^{\text{iso}}$

f12 (10–D)				
Δf_{opt}	S-CMA-ES - generation EC str.	S-CMA-ES - individual EC str.	MGSO	
1e2	**08.98**	-	**01.88**	02.02
1e1	**06.83**	-	-	01.05
1e0	**17.61**	-	-	-
1e-1	+	*	*	*
1e-2	+	*	*	*
1e-3	+	*	*	*
1e-4	+	*	*	*
1e-5	+	*	*	*
1e-6	*	*	*	*
1e-7	*	*	*	*
1e-8	*	*	*	*
param:	$(\mathbf{K}_{SE}^{\text{iso}}, 8, 1)$	$(\mathbf{K}_{\text{SE}}^{\text{ard}}, 0, 5, 0.1)$	$\mathbf{K}_{\text{SE}}^{\text{iso}}$	$\mathbf{K}_{\text{SE}}^{\text{ard}}$

f12 (20–D)				
Δf_{opt}	S-CMA-ES - generation EC str.	S-CMA-ES - individual EC str.		MGSO
1e3	01.39	00.05	00.18	00.42
1e2	**02.73**	-	-	00.03
1e1	**07.24**	-	-	-
1e0	59.06	-	-	-
1e-1	+	*	*	*
1e-2	+	*	*	*
1e-3	+	*	*	*
1e-4	+	*	*	*
1e-5	+	*	*	*
1e-6	*	*	*	*
1e-7	*	*	*	*
1e-8	*	*	*	*
param:	$(\mathbf{K}_{SE}^{\text{iso}}, 8, 1)$	$(\mathbf{K}_{\text{SE}}^{\text{ard}}, 0, 5, 0.1)$	$(\mathbf{K}_{SE}^{\text{iso}}, 0, 10, 0)$	$\mathbf{K}_{\text{SE}}^{\text{iso}}$

Table 3.6: Speed-up of S-CMA-ES using *individual-* and *generation-based* EC strategies and MGSO, compared to CMA-ES without a surrogate model – functions $f17 - f20$ (see Section 3.2.3.3 for details).

3.2.3 Results and their assessment

This section describes all outputs of the experimental evaluation and performance testing.

3.2.3.1 Best-fitness progress diagrams

Figures 3.6, 3.7, 3.8, 3.9 and 3.10 show the examples of the best-fitness progress with the best observed settings (see Table 3.3 for details). Medians and the first and third quartiles of the best fitness reached are shown; medians and quartiles measured for MGSO and S-CMA-ES on 15 and 10 independent runs (for both EC strategies), respectively.

3.2.3.2 Parameter table

Table 3.2 shows used parameters in our evaluations using the methods MGSO and S-CMA-ES. The symbols $\mathbf{K}_{\mathrm{SE}}^{\mathrm{iso}}$, $\mathbf{K}_{\mathrm{SE}}^{\mathrm{ard}}$, $\mathbf{K}_{\mathrm{exp}}$, $\mathbf{K}_{\mathrm{Matérn}}^{\nu=\frac{5}{2}}$, denote, respectively, the isotropic squared exponential, squared exponential with automatic relevance determination, exponential and Matérn with parameter $\nu = \frac{5}{2}$ covariance functions. $\boldsymbol{J}_{1,D}$ denotes the vector of ones of length equal to the dimension D of the input space.

3.2.3.3 Result table

Table 3.3 shows the speed-up of S-CMA-ES and MGSO, compared to CMA-ES without a surrogate model. For the respective targets (distances to the true optimum Δf_{opt}), speed-up of the expected running time (ERT) is shown. ERT is the number of function evaluations needed to reach the target divided by the ratio of the targets which reached the target. Stopping criteria: the distance 10^{-8} to the true optimum and $100 * D$ original fitness function evaluations.

The first column in each box corresponds to the overall best covariance function and EC settings of the S-CMA-ES: (covariance function, $\#(\text{model generations}), \sigma_{\mathrm{sample}}) = (\mathbf{K}_{SE}^{iso}, 8, 1)$. The second column corresponds to the best covariance function and EC settings in terms of the average speed-up for the respective function-dimension combination, if there was any better than the overall best observed settings. The last two columns in each box show the speed-up of the MGSO with both employed covariance functions.

Signs "-" instead of the speed-up values mean that, unlike the CMA-ES, no run of the considered method (S-CMA-ES or MGSO) was able to reach that target. Signs "+" mean that, unlike the employed method, no CMA-ES run (out of 20) was able to reach the target. Signs "*" mean that neither the considered method nor CMA-ES were able to reach the target. Speed-ups written in bold mark cases where the S-CMA-ES' or MGSO's median of the ERT is significantly lower than the median of the CMA-ES according to the one-sided Wilcoxon's test on the significance level $\alpha = 0.05$.

Conclusion

In this thesis, two optimization approaches based on Gaussian processes were tested on the set of niching functions from the CEC 2013 competition [1], and were compared to the state-of-the-art evolutionary approach in black-box optimization, CMA-ES. One of them is Model Guided Sampling Optimization [8], the other approach, S-CMA-ES, consists in using GP as a surrogate model for CMA-ES.

For this purpose, the framework integrating the methods was designed and implemented. Afterwards, it was used for performance testing of the integrated methods on benchmark functions and for comparison of the speed-up of both methods using different settings with CMA-ES with no surrogate model.

In the case of S-CMA-ES, two evolution control (EC) strategies were used, the *individual-* and *generation-based* EC strategies. Although, S-CMA-ES using *generation-based* EC strategy outperformed MGSO, both methods showed the performance improvement in most cases. On the other hand, the *individual-based* EC strategy brought the worst results in comparison to other methods. We also observed, that S-CMA-ES performs better using *generation-based* EC setting with more consequent model-evaluated generations (8 in our case), unmodified step size and the isotropic squared exponential covariance (SE) function. MGSO was tested only with two covariance functions, the isotropic SE function and SE function using automatic relevance determination, better results were achieved again for the former.

Bibliography

[1] Li, X.; Engelbrecht, A.; Epitropakis, M. G. Benchmark Functions for CEC'2013 Special Session and Competition on Niching Methods for Multimodal Function Optimization'. 2013. Available from: `http://goanna.cs.rmit.edu.au/~xiaodong/cec13-niching/competition/`

[2] Chaput, J. C.; Szostak, J. W. Evolutionary optimization of a nonbiological ATP binding protein for improved folding stability. *Chemistry & Biology*, volume 11, no. 6, June 2004: pp. 865–874.

[3] Arian Nik, M.; Fayazbakhsh, K.; Pasini, D.; et al. A comparative study of metamodeling methods for the design optimization of variable stiffness composites. *Composite Structures*, volume 107, 2014: pp. 494–501, ISSN 0263-8223.

[4] Tesauro, G.; Jong, N. K.; Das, R.; et al. On the use of hybrid reinforcement learning for autonomic resource allocation. *Cluster Computing*, volume 10, no. 3, 2007: pp. 287–299. Available from: `http://dblp.uni-trier.de/db/journals/cluster/cluster10.html#TesauroJDB07`

[5] Hansen, N. The CMA evolution strategy: a comparing review. In *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, edited by J. Lozano; P. Larranaga; I. Inza; E. Bengoetxea, Springer, 2006, pp. 75–102.

[6] Loshchilov, I. *Surrogate-Assisted Evolutionary Algorithms.* Theses, Université Paris Sud - Paris XI ; Institut national de recherche en informatique et en automatique - INRIA, Jan. 2013. Available from: `https://tel.archives-ouvertes.fr/tel-00823882`

[7] Jones, D. R.; Schonlau, M.; Welch, W. J. Efficient Global Optimization of Expensive Black-Box Functions. *J. of Global Optimization*,

volume 13, no. 4, Dec. 1998: pp. 455–492, ISSN 0925-5001, doi: 10.1023/A:1008306431147. Available from: `http://dx.doi.org/10.1023/A:1008306431147`

[8] Bajer, L.; Charypar, V.; Holeňa, M. Model guided sampling optimization with Gaussian processes for expensive black-box optimization. In *Blum, C. (ed.)*, GECCO Companion '13: New York: ACM, 2013, pp. 1715–1716.

[9] Bajer, L.; Holeňa, M. Two Gaussian Approaches to Black-Box Optomization. *CoRR*, volume abs/1411.7806, 2014. Available from: `http://arxiv.org/abs/1411.7806`

[10] Loshchilov, I.; Schoenauer, M.; Sebag, M. Self-adaptive surrogate-assisted covariance matrix adaptation evolution strategy. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, GECCO '12, New York, NY, USA: ACM, 2012, ISBN 978-1-4503-1177-9, pp. 321–328.

[11] Larrañaga, P.; Lozano, J. *Estimation of Distribution Algorithms: A new tool for evolutionary computation.* Kluwer Academic Pub, 2002.

[12] Bajer, L.; Charypar, V. *Gaussian Process sampling EDA algorithm.* 2013.

[13] Rasmussen, C.; Williams, C. *Gaussian Processes for Machine Learning.* 2013.

[14] MATLAB. *Release 2013b.* Natick, Massachusetts: The MathWorks, Inc., 2015.

[15] Rasmussen, C.; Williams, C. *Gaussian Processes for Machine Learning.* Adaptive Computation and Machine Learning, Cambridge, MA, USA: MIT Press, Jan. 2006, 248 pp. Available from: `http://mitpress.mit.edu/026218253X`

Appendix A

Acronyms

CMA-ES	Covariance Matrix Adaptation Evolution Strategy
EA	Evolutionary algorithm
EC	Evolution control
EGO	Efficient Global Optimization
GP	Gaussian process
MGSO	Model Guided Sampling Optimization
PoI	Probability of improvement

Appendix B

Contents of enclosed CD

```
│ src ........................................ the directory of source codes
│ ├── framework ................... the directory of framework source codes
│ └── thesis .............. the directory of LaTeX source codes of the thesis
├── text ........................................... the thesis text directory
│ ├── DP_Andrej_Kudinov_2015.eps.......... the thesis text in EPS format
│ └── DP_Andrej_Kudinov_2015.pdf ......... the thesis text in PDF format
└── README.txt ........................ the file with CD contents description
```